MITIGATING TASK DRIFT IN LONG-HORIZON GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

While coding agents have advanced rapidly, progress in GUI agent harnesses has lagged, especially for long-horizon tasks. In such tasks, a GUI agent can carry an incorrect or outdated completion judgment into later decisions and overlook unmet user requirements. We define task drift as this persistent mismatch between the agent's believed progress and the actual satisfaction of the original task requirements. The root cause is that progress judgments detach from the original requirements and lack requirement-specific environmental evidence. To address this, we introduce a framework that separates persistent user requirements from revisable plans and progress judgments, and grounds each judgment in concrete environmental evidence. Its controller distinguishes missing evidence from evidence of an unmet requirement: the former triggers inspection, the latter triggers correction, and invalidated support triggers revalidation only for affected requirements. On OSWorld, our framework improves the task score from 73.3 to 82.1 with Qwen3.7-Plus as the backbone.

1 INTRODUCTION

GUI agents that operate in real desktop environments—reading screens, clicking, typing, and manipulating files across applications—are increasingly deployed on long-horizon tasks whose success conditions unfold over dozens to hundreds of steps (Xie et al., 2024). At this horizon, no agent can reconstruct the full state of the world at every step. Instead, it must maintain beliefs about what has already been accomplished, and these beliefs function as control inputs to all subsequent decisions: which action to take next, which conditions to re-check, and, most critically, when to terminate. A completion judgment, once formed, is therefore not a private annotation within the context of the agent; it is a premise on which all subsequent decisions are conditioned. Although this failure mode is well documented: cross-domain failure attribution identifies undetected environment change and false goal assumptions among the dominant long-horizon failure categories (Wang et al., 2026a). What is missing is not a diagnosis but a mechanism that acts on this diagnosis. In addition, a judgment may also be *correct when formed yet outdated*: later actions overwrite the file, reset the form, or navigate away from the page on which the belief was grounded, while the belief itself survives unchanged in the context of the agent. In both cases, the stale belief continues to authorize decisions—the agent skips re-verification, plans around work it assumes to be complete, and ultimately reports success—while the underlying requirement is in fact unmet. We term this persistent mismatch between the believed progress of an agent and the actual satisfaction of the original task requirements *task drift*. Drift is especially damaging in long-horizon settings precisely because it compounds silently: every decision downstream of a drifted judgment is locally rational given the belief and globally wrong given the world, and the first moment at which the agent could notice the discrepancy is often the moment it has already declared the task finished.

Existing agent architectures lack the machinery to prevent drift. ReAct-style loops and their descendants interleave reasoning and action but accumulate self-generated progress notes in an undifferentiated context, where finishing a plan becomes indistinguishable from satisfying a requirement (Yao et al., 2022). Reflection and self-critique methods ask the same model that produced a belief to evaluate it, so a confidently wrong judgment is more often confirmed than corrected (Shinn et al., 2023).

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.

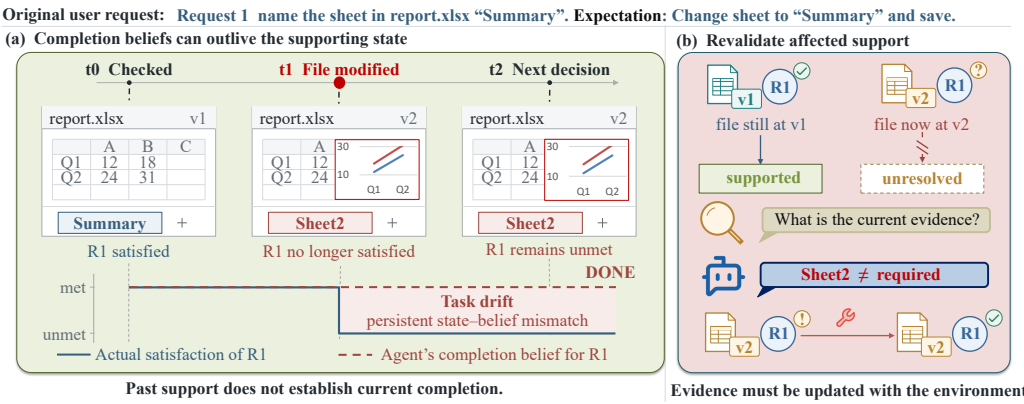

Figure 1: **Task drift and dependency-scoped revalidation.** A GUI agent executes a user request in a live desktop environment and must decide on its own when each requirement of the request is satisfied. Here, the request contains a single checkable obligation R1: name the sheet in `report.xlsx` as `Summary`. **(a)** At t_0, the agent checks R1 against file version v_1 and records it as satisfied. At t_1, a subsequent action modifies the file: version v_2 renames the sheet to `Sheet2`, and R1 is in fact no longer satisfied. The completion belief formed at t_0 nevertheless survives unchanged and licenses a DONE claim at t_2—a persistent mismatch between believed progress and actual requirement satisfaction that we call *task drift*. Prior support does not establish current completion. **(b)** Our framework binds every judgment to the world state it read. When the file version changes, only the support that depends on that file is revoked: R1 reopens as *unresolved*, the controller requests current evidence, the measured discrepancy (`Sheet2` $\neq$ `Summary`) guides a targeted repair, and R1 is re-supported on the corrected v_2. Evidence is updated along with the environment.

Outcome verifiers probe the environment progressively but return a trajectory-level reward (Cui et al., 2026): they cannot localize which requirement failed, and they run after the episode, when the budget for repair has been spent. Finally, systems that do react to environmental change tend to invalidate globally—discarding all prior judgments on any state shift—which forces expensive wholesale re-derivation and erodes the incentive of the agent to perform verification. Therefore, what is missing is a representation in which completion criteria remain anchored to the original request rather than to the evolving plan of the agent, in which every progress judgment is bound to the environmental evidence that supports it together with the state dependencies of that evidence, and in which revision is scoped to exactly the judgments whose support has been invalidated.

To address this gap, We introduce a framework that governs progress updates and execution decisions with requirement-specific environmental evidence. The framework wraps an GUI actor and intervenes at the two points where drift is created: when progress beliefs are updated, and when execution decisions—continue, inspect, repair, terminate—are made. It rests on three components. First, the framework *separates persistent user requirements from revisable plans and progress judgments*: the original instruction is compiled once into a requirement contract anchored to the exact source text, while plans can never add, discharge, or redefine a requirement. Second, *requirement–evidence associations* bind each obligation to evidence that could settle it, such as deterministic artifact checks or structured interface observations; obligations without a faithful check remain explicitly unresolved. Third, an *evidence-governed controller* uses this ledger to decide completion: unmet requirements trigger correction, missing evidence triggers inspection, and state changes reopen only affected obligations. Completion is granted only when every obligation is supported; otherwise, the claim is held within a bounded budget and reported as unresolved rather than success.

We evaluate the framework on OSWorld, a benchmark of real computer-use tasks scored by execution-based checkers in live virtual machines. With Qwen3.7-Plus(Bai et al., 2025; Team, 2026) as the backbone, the framework improves the task score from 73.3 to 82.1 over a strong hierarchical GUI-agent baseline operating under identical step, and time. The gain is achieved without modifying the actor or the benchmark scorers, indicating that a substantial fraction of long-horizon failures are recoverable once completion judgments are governed by requirement-specific evidence.

Our contributions are threefold.

- We define *task drift*—the persistent mismatch between the believed progress of an agent and the actual satisfaction of the original requirements—and identify its two generators: judgments incorrect at formation and judgments outdated by subsequent state change.

- We introduce a framework that (i) separates persistent requirements from revisable plans and progress judgments, (ii) binds each judgment to requirement–evidence associations that record the state dependencies and provenance of the evidence, and (iii) governs execution with a controller that directs inspection for missing evidence, corrective action for measured discrepancies, and dependency-scoped revalidation when prior support is invalidated.
- We demonstrate on OSWorld that this governance raises the task score of a Qwen3.7-Plus agent from 73.3 to 82.1, and that unresolved requirements are reported honestly rather than absorbed into silent failure.

2 Related Work

We organize prior work along the two generators of task drift identified in §1: how progress judgments are formed (§2.2) and how they are maintained (§2.3), after situating the setting (§2.1).

2.1 GUI agents and self-declared completion

GUI agents have progressed from single-step command following (Zheng et al., 2024) to hierarchical systems that plan, ground, and act in real desktop and web environments (Gonzalez-Pumariega et al., 2025; Qin et al., 2025; Agashe et al., 2025; Wang et al., 2025), and are evaluated on execution-scored benchmarks (Xie et al., 2024; Rawles et al., 2025; Yuan et al., 2026; Bonatti et al., 2024). These benchmarks share the structural property that motivates our work: task success is defined by external world state and verified post hoc, yet the decision to stop and report completion is made by the agent itself, from partial observation, in the middle of execution. Believed and actual progress measurably diverge at exactly this point. Horizon-dependent failure compounds along trajectories rather than at single steps (Wang et al., 2026a); most long-horizon failures in production are silent, the agent reporting success while the world contradicts the report (Wu, 2026). The mechanism presupposes only a persistent world, partial observability, and self-declared completion—all three hold in OSWorld, our evaluation domain—and longer horizons amplify drift without creating it.

2.2 Forming progress judgments: from self-verification to external verifiers

A first line of work forms judgments within the actor. ReAct-style loops (Yao et al., 2022) interleave reasoning and action while accumulating self-generated progress notes; Reflexion, Self-Refine, and CRITIC (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024) add self-critique, with or without tools. This line inherits a documented limit: without external feedback, LLMs do not reliably locate their own errors (Huang et al., 2024), so a confidently wrong completion judgment is more often confirmed than corrected—and because the context mixes plans with requirements, finishing the plan becomes indistinguishable from satisfying the request.

A second line moves the judgment outside the actor. LLM-as-judge and process reward models score responses or steps (Zheng et al., 2023; Lightman et al., 2024), but they grade text, not world state. More recent verifiers face the environment directly: web-agent verifiers detect false or premature termination (Pan et al., 2024; Chae et al., 2026); VeriGUI predicts the expected screen effect of an action and checks the next screenshot against it (Zhang et al., 2026b); VAGEN-B makes the verifier itself a tool-augmented GUI agent, with verification far outpacing solving (83.1% versus 55.9%) (Cui et al., 2026); and work on agentic abstention asks when agents should stop, finding that recognizing when one is stuck does not improve with scale (Luo et al., 2026)—explicit stop machinery must be designed, not expected to emerge.

Three limitations cut across both lines. Judgments are formed against the agent's own account of the task—a plan, a summary, a claimed outcome—rather than a persistent requirement structure anchored to the original request. Judgments are emitted as scalars or verdicts whose evidentiary basis is discarded, so nothing can later determine whether that basis still holds. And both lines collapse two semantically different situations—the absence of evidence and evidence that a requirement is unmet—into a single negative signal, eliciting correction where inspection was appropriate. Our framework addresses all three: contracts anchor completion criteria to the original request, requirement–evidence associations retain each judgment's evidentiary basis, and a controller separates absence of evidence from evidence of violation.

2.3 MAINTAINING PROGRESS JUDGMENTS: TRUTH MAINTENANCE, MEMORY, AND EXECUTION GOVERNANCE

Long-horizon execution accumulates task drift: the agent's checks gradually diverge from the original request, without any single step being wrong. Existing work addresses the resulting judgment-maintenance problem along three lines, but all three share one strategy—adding information—and one blind spot: they indicate that something is wrong, not where, and our experiments show that merely flagging an error yields no benefit.

Truth maintenance, AGM revision, and requirements traceability maintain judgments under revision (Doyle, 1979; Gotel & Finkelstein, 1994), but desktop justifications are reads of mutable state whose truth decays with the world. Memory and compaction systems manage what a long execution retains (Omri et al., 2026; Li et al., 2026); Governance Decay (Chen, 2026) shows compaction can silently erase constraints (0% to 30%), yet validity against the current world is not their invariant. Execution-governance work—durable execution, transactional effects, checkpoint/restore, failure attribution, recovery training (Balakrishnan et al., 2026; Chen et al., 2026; Bu et al., 2026; Lin et al., 2026; Zhang et al., 2026a)—presupposes a failure signal, and task drift defeats it precisely because it is ledger-consistent: the records contain no error to detect, and the mismatch exists only between records and world. Crab's finding is closest: restoring only the conversation yields 8% correctness, versus 100% when the environment is restored with it (Wu et al., 2026); yet checkpoint/restore answers *where to resume*, not *what still holds*.

What is missing is localization: identifying which state has diverged from the world. Our framework supplies it by binding each state to its world files—contracts anchor completion criteria to the original request, requirement–evidence associations tie each judgment to the world state that licensed it, and the controller maps satisfied/refuted/unresolved to release/correction/inspection, restraining task drift at both generators.

3 METHOD

A natural reading of prior verification work is that better judgments arise from providing the model with more information: more context, more screenshots, and more feedback. Our controlled analysis (§4) suggests that this premise is misplaced in GUI settings. A multimodal model observing a desktop cannot reliably determine whether a state satisfies the request, and it is even less able to identify where a failure lies once one exists; enlarging the volume of information it receives leaves both limitations untouched. Precise world information differs in kind rather than in degree. Because it binds each judgment to a measured state of the environment, it supplies the premise that GUI agents otherwise lack: the location of the discrepancy. The gains we report follow from converting an undifferentiated signal that something is wrong into a named location of error, and the framework is built around exactly this conversion.

We organize the framework around the mechanism that each diagnosed failure demands. §3.1 fixes the execution model, the two points at which governance intervenes, and the requirement contract that anchors every verdict to the original request. §3.2 defines the evidence binding that allows a judgment to name *where* it is wrong. §3.2 describes the controller that routes localized verdicts, maintains judgments under change, and enforces the budgets that make honest termination well-defined. Throughout, the actor remains unmodified and every model judgment remains a fallible assessment; authority comes from the structure of the records, not from any single verdict.

3.1 EXECUTION MODEL AND THE REQUIREMENT CONTRACT

An episode is a loop: the actor observes a screenshot, emits an action, and the environment executes it, until the actor emits DONE or FAIL. The framework wraps this loop and intervenes at precisely the two points where drift arises. The first intervention occurs *per action*: at each step, the framework may acquire bounded host-executed observations of world state, and the admissibility gate of §3.2 determines whether each observation enters the ledger, whereas narration produced by the actor—plans, progress notes, and self-reported completions—never does. This is the inner loop of Fig. 2 a. The second intervention occurs *per completion claim*: when the actor emits DONE, the controller holds the claim instead of releasing it to the environment, and a bounded review re-judges

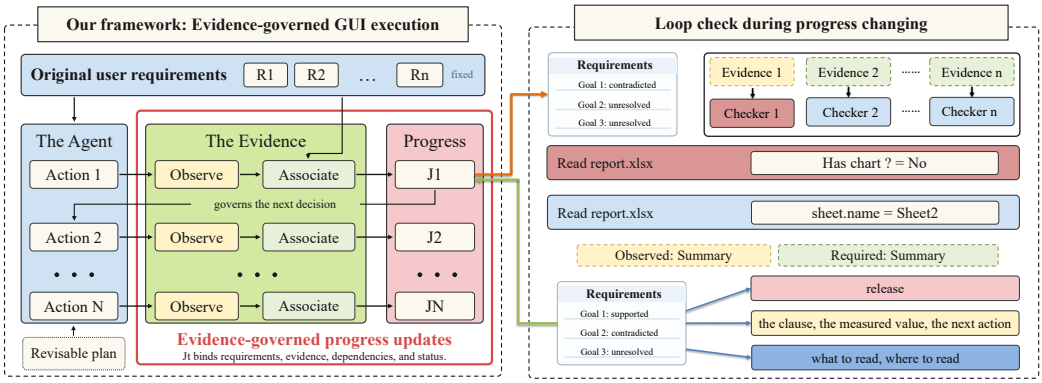

Figure 2: **Evidence-governed execution.** The framework wraps an unmodified GUI actor and intervenes at the two points where drift is created. **(a)** Per action: host-executed observations of world state pass through an admissibility gate before entering the ledger, whereas narration produced by the actor never does. **(b)** Per completion claim: a DONE claim is held while a bounded review routes each unsettled requirement to corrective feedback, inspection, or release, and only judgments whose recorded dependencies still match the world retain their support.

every unsettled requirement against admitted evidence and routes each requirement independently—corrective feedback for measured contradictions, inspection targets for missing evidence, and release when every requirement is supported. This is the outer loop of Fig. 2 b.

The first generator of drift is a verdict that targets the wrong object: the agent checks its plan, its summary, or its memory of the request rather than the request itself. The contract removes that degree of freedom. At the start of an episode, the framework compiles the original instruction into a *requirement ledger*, with one row per sentence of the instruction, preserved verbatim. Each row R_i records:

- **source_quote**: the verbatim clause. The ledger rejects any judgment that restates it, so that completion criteria remain anchored to the original request rather than to a paraphrase.
- **interpretation**: a revisable reading of the clause. Editing this field increments a revision counter and reopens the row, so a corrected reading never inherits evidence from the previous reading.
- **conclusion**: *supported*, *contradicted*, or *unresolved*. Unresolved is a first-class state, not a failed verdict.
- **evidence_refs** and **measured_statement**: the observations that the judgment cites and what they actually showed. The ledger rejects any non-unresolved conclusion that lacks either.
- **dependencies**: the world-state versions on which the conclusion rests. This field makes a judgment *localizable* (§3.2) and *invalidatable* (§3.2).
- **role**: *outcome* or *context*. Clauses that describe the starting situation rather than a required outcome may cite initial observations; the ledger does not require them to cite current-state evidence.

3.2 Evidence binding: making errors locatable

Our controlled analysis (§4) shows that providing the same judge with more evidence does not reduce false support: the bottleneck is not observability but whether a judgment can name *where* it is wrong, and a judgment can name its location only if it records the source of its reading. This subsection defines the two halves of that binding: what counts as evidence, and what a judgment must carry to remain tied to the world.

Acquisition. Evidence enters the ledger only as *observations* produced by host-executed reads over a fixed, predicate-free schema: file properties (existence, size, digest, bytes, inventory, text, JSON pointers, PDF text and metadata, image pixels, archive members), browser fields (URL, title, text, form values), and screen pixels, together with *action receipts* recording that a named action ran and what it returned—a receipt proves that the action ran, never that its intended effect succeeded. Each observation includes its *dependencies*, the versions of the world objects it read, and a *scope* indicating coverage and provenance.

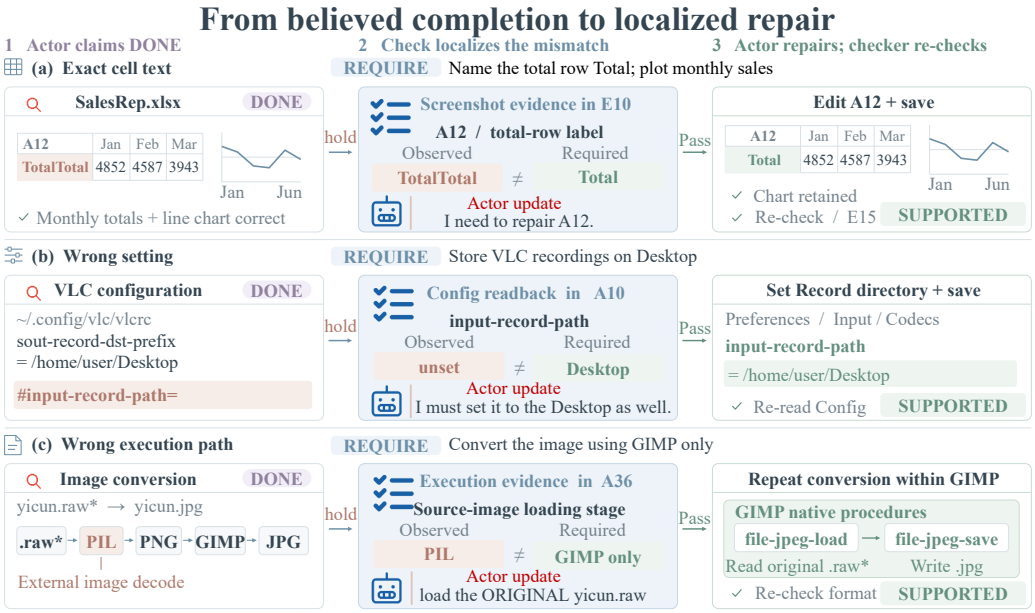

Figure 3: **The evidence for evidence governance.**From believed completion to localized repair. The figure illustrates three historical GUI trajectories, showing how the actor initially claims completion, the checker localizes the mismatch based on current world evidence, and the actor performs localized repairs followed by checker re-verification. The examples include (a) exact cell text, where TotalTotal in SalesRep.xlsx is corrected to Total; (b) wrong setting, where the VLC record path is configured to Desktop; and (c) wrong execution path, where the image conversion is repeated within GIMP. In all cases, the checker supports the repaired state, and the final official score is 1.0.

Admissibility. The gate admits an observation for a conclusive verdict only when all of the following hold: the observation carries no error, partial, or unavailable flag; its coverage is complete; every dependency still matches the current version map of the host; it was acquired within the current round and after any invalidation event; and, for image evidence, the judge received the actual pixels, since a hash or filename does not qualify. Two further rules ensure provenance integrity: initial-source observations establish baselines and context clauses but never current-state satisfaction, and requirements flagged as process or history obligations must cite event- or interval-kind evidence rather than a single end-state read. The gate rejects any claim resting on inadmissible evidence with a typed reason, and the requirement stays unresolved; a claim made by the actor never counts as an observation.

Claim-time review. When the actor emits DONE, the controller conducts a bounded review. Each checker call receives the original instruction, the unsettled requirement rows, frozen conditions, candidate differences, the fallible prior, projected observations bounded in bytes and image count, recent action receipts, and the remaining budget. The checker either returns a report of per-requirement judgments citing observation refs, or issues read requests that the host executes once and answers with actual receipts; a read turn carries no judgment authority, and no read job survives the round. The controller discards judgments from any round during which the actor acted.

Maintenance under change. Binding gives judgments a lifetime as well as a location: a supported verdict authorizes completion only for the world versions on which it rests. The host maintains a version map over dependency keys—file digests, browser scope versions, screen captures, action epochs—refreshed on every observation and on explicit file-version reads, and no new conclusive judgment may rest on evidence whose dependencies no longer match. Under the default *scoped* policy, a refresh reopens exactly the rows whose recorded dependencies no longer match current versions and retains unaffected judgments; a byte- or property-identical re-read may revalidate the same interpretation, only within the same revision and never across an invalidation event.

Review produces a *vector* verdict rather than a scalar: each requirement is judged independently and routed by conclusion. A *contradicted* requirement returns a corrective card naming the source clause, what the host actually measured, the candidate difference, and the next action that would settle it; repair advice sets a target state against the current value and never asks for a blind toggle.

Table 1: Main Experiments

Method	Step	Success Rate (%)					
		OS	Office	Daily	Prof.	Work.	Avg.
OpenCUA-72B (Wang et al., 2026b)	100	61.13	44.73	49.95	72.58	22.16	44.91
Seed-1.8 (Seed, 2026)	100	66.67	68.80	67.05	71.43	42.38	61.87
DeepMiner-Mano-72B(Fu et al., 2025)	100	66.67	63.22	52.51	83.67	24.41	53.91
Claude-Sonnet-4.5	100	70.83	72.59	61.35	63.27	49.54	62.84
Kimi K2.5 (Team et al., 2026)	100	73.91	69.11	66.42	63.47	46.11	63.34
CoAct-1 w/ GPT-5 (Song et al., 2026)	100	75.00	62.93	57.94	71.43	47.87	59.93
GTA1 w/ GPT-5 (Yang et al., 2026b)	100	79.17	63.91	62.56	79.59	50.91	63.41
OS-Symphony w/ GPT-5 (Yang et al., 2026a)	100	79.17	65.73	67.76	69.23	57.98	65.84
Agent S3 w/ GPT-5 (Gonzalez-Pumariega et al., 2025)	100	77.50	66.46	61.23	69.80	51.37	62.63
UiPath w/ Opus 4.5 (UiPath, 2025)	100	70.83	74.13	68.33	73.47	52.97	67.14
Agent S3 w/ Opus 4.5	100	75.00	76.06	67.51	59.18	59.00	67.46
HIPPO w/ Opus 4.5 (HCIII Team, Lenovo, 2026)	100	87.50	74.27	69.27	**95.92**	64.31	74.49
VLAA-GUI w/ Opus 4.5 + MAI-UI	100	91.67	84.26	72.77	83.67	61.09	76.26
VLAA-GUI w/ Opus 4.6 (Han et al., 2026)	100	**91.70**	82.87	75.17	83.67	65.60	77.45
Ours w/ Qwen3.7-Plus+Holo2-30B-A3B	100	89.47	83.97	**82.95**	86.11	76.56	82.10
Ours w/ Qwen3.8-Max+Holo2-30B-A3B	100	87.50	**93.65**	82.21	73.47	**83.33**	**84.98**

An *unresolved* requirement returns an inspection card naming what to read, never what to fix; the actor must not rework unconfirmed state based on missing evidence. A *supported* requirement releases its share of the claim, and the claim is released as DONE only when every requirement is supported. The asymmetry is the point: a bare "wrong" verdict leaves the actor with a search problem over requirements, objects, and target states, whereas a localized discrepancy collapses repair to a near-atomic action.

4 Experiments

4.1 Setup

Benchmark. OSWorld:361 canonical tasks across ten application domains (Chrome, multi-application workflows, LibreOffice Calc/Impress/Writer, GIMP, VS Code, Thunderbird, OS utilities, VLC), each executed in a live Docker and scored by the benchmark's official execution-based checkers. Scores are terminal-state only: the checker inspects files, application state, and system configuration after the episode ends. We never modify tasks, scorers, or the actor.

Agent. The actor is a hierarchical GUI agent that uses a Qwen3.7-Plus(Bai et al., 2025) or Qwen3.8-max(Qiu et al., 2026) reasoning model and a Holo2-30B-A3B(Company, 2025) grounding model, and it operates on screenshots at 1920×1080. The framework wraps the actor without modification; all governance is conducted through the ledger, evidence store, and controller described in §3.

Budgets. The budgets are identical for every system: at most 100 steps, at most four auxiliary verification calls per episode, two checker calls per completion claim, four read acquisitions per review round, and three held completion claims. These caps are part of the method rather than a limitation of the evaluation, since they make honest release well-defined.

Protocol. The protocol consists of one fresh run over all 359 tasks with a concurrency of 8, where each task is placed in an owned container restored from the canonical snapshot. A task is included in the results only if it reaches a natural terminal state and receives an official score; infrastructure failures are reported, never imputed as zero, and never selectively re-run. The evaluation set differs from the canonical OSWorld protocol: nine Google Sheets tasks that are commonly treated as optional by the community are omitted, and two additional tasks are excluded because Macy and IMDb.com were consistently unreachable due to network failures.

4.2 Main results

Overall Success Rate And SOTA Position Under the Max 100 Steps budget, Ours Qwen3.8-max with Holo2-30B-A3B attains the highest average success rate in Table 1 at 84.98%, exceeding Ours w/ Qwen3.7-Plus with Holo2-30B-A3B at 82.10% by 2.88 points and the strongest external baseline, VLAA-GUI w/ Opus 4.6 (Han et al., 2026)at 77.45%, by 7.53 points. This establishes

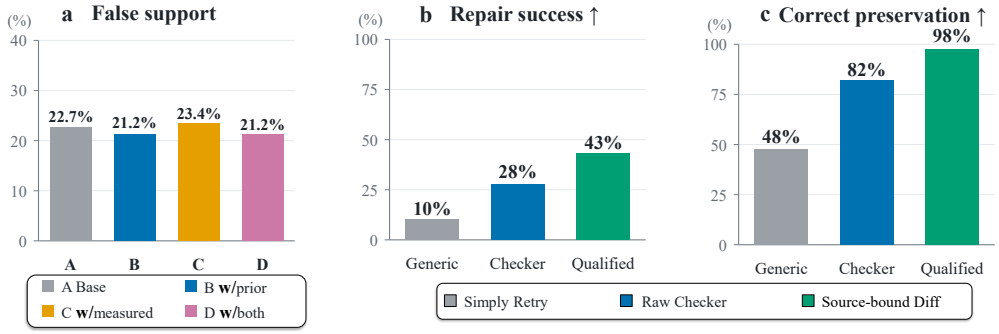

Figure 4: **Error location, not evidence volume.** **(a)** In a controlled cohort of 70 specifications, augmenting the same judgment protocol with a prior, a measured supplement, or both leaves false support on mechanically verified violations essentially flat: 22.7%, 21.2%, 23.4%, and 21.2% for the four evidence configurations. The bottleneck is the binding of the judgment, not the volume of the evidence. **(b)** Across 270 repair branches with the same actor and the same initial states, a specification-generated source-bound discrepancy repairs 43% of violated requirements, versus 28% for the raw verdict of the checker and 10% for generic retry. **(c)** On the paired correct states, the same feedback also preserves 98% of artifacts, versus 82% for checker feedback and 48% under generic retry: naming the error both repairs more and corrupts less.

a clear aggregate lead under an identical 100-step setting rather than a marginal tie with a single baseline. The table entry of 84.98% is lower than the official Qwen3.8-max result of 86.1% because the holo grounding operations interfere with the complete context; when Qwen3.8-max is used alone, the score reaches 86.65%, which is 0.55 points above the official 86.1%. On this evaluation set, that 0.5-point margin indicates that headroom is close to exhausted, so the absolute gain is necessarily small.

Domain-Level Evidence And Visual Verification Limits Domain-level results show that the gains are concentrated in artifact-bound and text-checkable domains. Office reaches 93.65%, which is 9.39 points above VLAA-GUI w/ Opus 4.5 + MAI-UI at 84.26% and 10.78 points above VLAA-GUI w/ Opus 4.6 at 82.87%. Work reaches 83.33%, which is 17.73 points above VLAA-GUI w/ Opus 4.6 at 65.60%. Daily reaches 82.95% with Ours Qwen3.7-Plus with Holo2-30B-A3B and 82.21% with Ours Qwen3.8-max with Holo2-30B-A3B, compared with 75.17% for VLAA-GUI w/ Opus 4.6. These domains provide textual artifacts or state that can be checked against evidence, so evidence-governed execution can constrain completion claims more reliably.

Variant Trade-Offs And Evaluation Set Saturation The two Ours variants reveal a clear trade-off. Moving from Qwen3.7-Plus to Qwen3.8-max raises the average from 82.10% to 84.98%, with Office improving by 9.68 points and Work by 6.77 points, while Daily drops by 0.74 points, OS by 1.97 points, and Prof. by 12.64 points. Prof. is therefore the main source of variance and limits cross-domain consistency. On this set, the single-model Qwen3.8-max result of 86.65% is only 0.5 points above the official 86.1%, so the task set is already close to saturation. The 84.98% table entry for Ours Qwen3.8-max with holo2 30b is thus explained by context interference from holo grounding operations, while the domain-level pattern reflects the boundary between text-checkable domains and visually verified domains.

Controlled evidence: information volume versus error location. A controlled analysis isolates why additional evidence does not translate into better judgments (Figure 4a). We expand 70 requirement specifications into 1,050 judgment cells, hold the judgment protocol fixed, and vary only the evidence bundle: a base configuration (A), an added requirement prior (B), an added measured supplement (C), and both additions (D). False support on mechanically verified violations remains essentially flat across the four configurations—22.7%, 21.2%, 23.4%, and 21.2%, respectively—and replicates at 16–19% on a second model (qwen3.8-flash(Team, 2026)). The bottleneck therefore lies in how a judgment binds to evidence, not in the volume of evidence provided. The same comparison isolates the complementary factor: whether feedback names the location of the error. Across 270 repair branches that share the actor and the initial states, we replace only the feedback content. Generic retry, which asks the actor to recheck and finish without locating any failure, repairs 10% of violated requirements. The raw verdict of the checker, which identifies the failing requirement but not the measured discrepancy, repairs 28%. A source-bound discrepancy, generated from the specification

Table 2: Ablation study on the contribution of the evidence-governed harness and requirement prior. The actor-only baseline yields a score of 73.3. Enabling the harness alone improves the score to 77.6, a gain of 4.3 points. The full system, which further incorporates the requirement prior, achieves the best performance of 82.1, corresponding to an additional gain of 4.5 points over the harness-only variant and a total improvement of 8.8 points over the baseline.

System	Harness	Prior	score	Δ
Actor only	✗	✗	73.3	—
+ evidence-governed harness	✓	✗	77.6	+4.3
+ requirement prior	✓	✓	**82.1**	+4.5

and pairing each requirement with its measured state and repair scope, repairs 43% . On the paired correct states (Figure 4c), the same feedback preserves 98% of artifacts , versus 82% for checker feedback and 48% under generic retry. Naming the location of the error thus repairs more violations while corrupting fewer correct artifacts. These three experiments lead to a clear conclusion for GUI tasks: when the model is asked to identify its own errors in isolation, it seldom succeeds, and the resulting gain is not statistically significant. Informing the model that an error exists does allow it to detect a subset of errors, but this detection remains statistically indistinguishable from chance. Only when the model is told where the error is located can it repair the problem correctly and completely.

Ablation of the evidence-governed harness and the requirement prior. Table 2 decomposes the gain into the two components of the framework on the Qwen3.7-Plus configuration. The actor-only baseline, which receives the task instruction and the raw trajectory without requirement tracking, attains a score of 73.3. Enabling the evidence-governed harness alone—requirement contracts, evidence bindings, and the evidence-governed controller, but no requirement prior—raises the score to 77.6, a gain of 4.3 points. The full system, which further incorporates the requirement prior, reaches 82.1, an additional gain of 4.5 points and a total improvement of 8.8 points over the baseline. The two components therefore contribute comparably, indicating that requirement-scoped evidence governance and the requirement prior address complementary failure modes rather than redundant signals. These complementary gains justify jointly deploying both components in practice.

Remaining Bottleneck and Verification Limits The above experiments clearly indicate that although current GUI models generally suffer from severe hallucination and poor single-pass execution efficiency, hallucination can be further compressed by imposing constraints on evaluable states. Nevertheless, the overall bottleneck remains the hallucination of GUI models, that is, the inability to confirm truth by viewing the same image multiple times. Consequently, under the setting of checking its own trajectory, GUI execution lags far behind text-only execution. This gap also constitutes the key obstacle to realizing loops similar to those of coding agents, because no reliable online verification signal is available to provide rewards. In coding agents, such a signal can be derived from executable tests, compilers, or type checkers, which offer objective and reproducible feedback. In contrast, GUI tasks lack a general-purpose verifier that can independently confirm whether an action has achieved its intended effect.

5 Conclusion

We introduced *task drift*—the persistent mismatch between the believed progress of a GUI agent and the actual satisfaction of the original requirements—and attributed it to two generators: judgments that are wrong when formed, and judgments that are right when formed but invalidated by later state change. Our framework prevents both by construction: a requirement contract anchors completion criteria to the verbatim request, requirement–evidence associations bind every judgment to the world state it read, and an evidence-governed controller routes localized verdicts to inspection, correction, or scoped revalidation. On OSWorld, governed execution raises the task score of a Qwen3.7-Plus agent from 73.3 to 82.1 without modifying the actor or the benchmark scorers. The guarantee of the framework is deliberately narrow—it releases completion only when every requirement carries a current, admissible, source-bound support—yet its premise generalizes: wherever an agent must decide, from partial observation, that its work is done, what verification is permitted to conclude matters more than how much of it is performed.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools for [tasks with required disclosure]. We have not used generative AI tools for [other tasks with required disclosure], and [the rest of the required disclosure tasks] are not applicable to this work. Additionally, we used generative AI tools for [tasks with recommended disclosure]. We have reviewed all AI-assisted work. [Elaborate. For example, "we checked LLM-generated research ideas for potential plagiarism through a manual literature survey", "LLM-generated code was verified and tested for correctness by 2 authors", etc.]. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

See the ICLR 2027 AI Policy for Authors for more details. This statement should not be more than 1 page.

ETHICS STATEMENT

(This section is **recommended** and does not count toward the page limit.)

If authors feel that their paper submission raises questions regarding the Code of Ethics, they are encouraged to include a paragraph of Ethics Statement (at the end of the main text before references) to address potential concerns where appropriate. Topics include, but are not limited to, studies that involve human subjects, practices to data set releases, potentially harmful insights, methodologies and applications, potential conflicts of interest and sponsorship, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues (e.g., IRB, documentation, research ethics). This statement should not be more than 1 page.

REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

It is important that the work published in ICLR is reproducible. Authors are strongly encouraged to include a paragraph-long Reproducibility Statement at the end of the main text (before references) to discuss the efforts that have been made to ensure reproducibility. This paragraph should not itself describe details needed for reproducing the results, but rather reference the parts of the main paper, appendix, and supplemental materials that will help with reproducibility. For example, for novel models or algorithms, a link to an anonymous downloadable source code can be submitted as supplementary materials; for theoretical results, clear explanations of any assumptions and a complete proof of the claims can be included in the appendix; for any datasets used in the experiments, a complete description of the data processing steps can be provided in the supplementary materials. Each of the above are examples of things that can be referenced in the reproducibility statement.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents. *arXiv preprint arXiv:2504.00906*, 2025.

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*, 2025.

Mahesh Balakrishnan, Ashwin Bharambe, Davide Testuggine, David Geraghty, David Mao, Vidhya Venkat, Ilya Mironov, Rithesh Baradi, Gayathri Aiyer, and Victoria Dudin. Logact: Enabling agentic reliability via shared logs. *arXiv preprint arXiv:2604.07988*, 2026.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*, 2024.

Tianpeng Bu, Xin Liu, Qihua Chen, Hao Jiang, Shurui Li, Hongtao Duan, Lu Jiang, Lulu Hu, Bin Yang, and Minying Zhang. Recovering policy-induced errors: Benchmarking and trajectory synthesis for robust gui agents. *arXiv preprint arXiv:2605.29447*, 2026.

Hyungjoo Chae, Seonghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, et al. Web-shepherd: Advancing prms for reinforcing web agents. *Advances in Neural Information Processing Systems*, 38:63314–63356, 2026.

Shiyang Chen. Governance decay: How context compaction silently erases safety constraints in long-horizon llm agents. *arXiv preprint arXiv:2606.22528*, 2026.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using llm agents. *arXiv preprint arXiv:2606.17573*, 2026.

H Company. Holo2 - open foundation models for navigation and computer use agents, 2025.

Chaoqun Cui, Jing Huang, Shijing Wang, Liming Zheng, Qingchao Kong, and Zhixiong Zeng. Agentic reward modeling: Verifying gui agent via online proactive interaction. *arXiv preprint arXiv:2602.00575*, 2026.

Jon Doyle. A truth maintenance system. *Artificial intelligence*, 12(3):231–272, 1979.

Tianyu Fu, Anyang Su, Chenxu Zhao, Hanning Wang, Minghui Wu, Zhe Yu, Fei Hu, Mingjia Shi, Wei Dong, Jiayao Wang, et al. Mano technical report. *arXiv preprint arXiv:2509.17336*, 2025.

Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. Scaling agents for computer use. *arXiv preprint arXiv:2510.02250*, 2025.

Orlena CZ Gotel and CW Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of IEEE international conference on requirements engineering*, pp. 94–101. IEEE, 1994.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Qijun Han, Haoqin Tu, Zijun Wang, Haoyue Dai, Yiyang Zhou, Nancy Lau, Alvaro A Cardenas, Yuhui Xu, Ran Xu, Caiming Xiong, et al. Vlaa-gui: Knowing when to stop, recover, and search, a modular framework for gui automation. *arXiv preprint arXiv:2604.21375*, 2026.

HCIII Team, Lenovo. Hippo agent. `https://github.com/wadang/muscle-memagent`, 2026. GitHub repository, accessed 2026-09-26.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Yujiang Li, Zhenyu Hou, Yi Jing, Jie Tang, and Yuxiao Dong. Compactionrl: Reinforcement learning with context compaction for long-horizon agents. *arXiv preprint arXiv:2607.05378*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Zichuan Lin, Feiyu Liu, Yijun Yang, Jiafei Lyu, Yiming Gao, Yicheng Liu, Zhicong Lu, Yangbin Yu, Mingyu Yang, Junyou Li, et al. Ui-voyager: A self-evolving gui agent learning via failed experience. *arXiv preprint arXiv:2603.24533*, 2026.

Han Luo, Bingbing Wen, and Lucy Lu Wang. Agentic abstention: Do agents know when to stop instead of act? *arXiv preprint arXiv:2606.28733*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Yasmine Omri, Ziyu Gan, Zachary Broveak, Robin Geens, Zexue He, Alex Pentland, Marian Verhelst, Tsachy Weissman, and Thierry Tambe. Agent memory: Characterization and system implications of stateful long-horizon workloads. *arXiv preprint arXiv:2606.06448*, 2026.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.

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.

Zihan Qiu, Zekun Wang, Xiao Li, Yanpeng Li, Yang Xu, Yixuan Wang, Huaqing Zhang, Rui Men, Bochao Mao, Chengruidong Zhang, et al. On the design of qwen3. 8-next architecture: Evaluation, efficiency, and training stability. *arXiv preprint arXiv:2608.30320*, 2026.

Chris 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. In *International Conference on Learning Representations*, volume 2025, pp. 406–441, 2025.

Bytedance Seed. Seed1. 8 model card: Towards generalized real-world agency. *arXiv preprint arXiv:2603.20633*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Linxin Song, Yutong Dai, Viraj Prabhu, Jieyu Zhang, Taiwei Shi, Li Li, Junnan Li, Zeyuan Chen, Jieyu Zhao, Ran Xu, et al. Coact-1: Computer-using multi-agent system with coding actions. In *International Conference on Learning Representations*, volume 2026, pp. 127091–127107, 2026.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Ziwei Chai, Y Charles, HS Che, Cheng Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

UiPath. Uipath screen agent. `https://github.com/xlang-ai/OSWorld/tree/main/mm_agents/uipath`, 2025. GitHub repository, accessed 2026-09-26.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *arXiv preprint arXiv:2509.02544*, 2025.

Xinyu Jessica Wang, Haoyue Bai, Yiyou Sun, Haorui Wang, Shuibai Zhang, Wenjie Hu, Mya Schroder, Bilge Mutlu, Dawn Song, and Robert D Nowak. The long-horizon task mirage? diagnosing where and why agentic systems break. *arXiv preprint arXiv:2604.11978*, 2026a.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. Opencua: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806, 2026b.

Tianyuan Wu, Chaokun Chang, Lunxi Cao, Wei Gao, and Wei Wang. Crab: A semantics-aware checkpoint/restore runtime for agent sandboxes. *arXiv preprint arXiv:2604.28138*, 2026.

Wei Wu. When errors become narratives: A longitudinal taxonomy of silent failures in a production llm agent runtime. *arXiv preprint arXiv:2606.14589*, 2026.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Bowen Yang, Kaiming Jin, Zhenyu Wu, Zhaoyang Liu, Qiushi Sun, Zehao Li, JingJing Xie, Zhoumianze Liu, Fangzhi Xu, Kanzhi Cheng, et al. Os-symphony: A holistic framework for robust and generalist computer-using agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22300–22330, 2026a.

Yan Yang, Dongxu Li, Yutong Dai, Yuhao Yang, Ziyang Luo, Zirui Zhao, Zhiyuan Hu, Junzhe Huang, Amrita Saha, Zeyuan Chen, et al. Gta1: Gui test-time scaling agent. In *International Conference on Learning Representations*, volume 2026, pp. 48295–48323, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Mengqi Yuan, Zilong Zhou, Xinzhuang Xiong, Weiming Wu, Jiayang Sun, Jiamin Song, Kaiqian Cui, Bowen Wang, Haoyuan Wu, Yitong Li, et al. Osworld2. 0: Benchmarking computer use agents on long-horizon real-world tasks. *arXiv preprint arXiv:2606.29537*, 2026.

Boxuan Zhang, Jianing Zhu, Zeru Shi, Dongfang Liu, and Ruixiang Tang. Agentforesight: Online auditing for early failure prediction in multi-agent systems. *arXiv preprint arXiv:2605.08715*, 2026a.

Yuzhe Zhang, Xianwei Xue, Xingyong Wu, Mengke Chen, Chen Liu, Xinran He, Run Shao, Feiran Liu, Huanmin Xu, Qiutong Pan, et al. Don't act blindly: Robust gui automation via action-effect verification and self-correction. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28924–28941, 2026b.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v (ision) is a generalist web agent, if grounded. *arXiv preprint arXiv:2401.01614*, 2024.

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. *Advances in neural information processing systems*, 36:46595–46623, 2023.