Augmenting Interface Usability Heuristics for Computer-Use Agents

Anonymous ACL submission

Abstract

Recent advances enable general computer-use agents that interpret screens and execute grounded actions from human instructions, yet they still struggle to generalize to unseen and evolving interfaces. While improving agent capability remains important, agent-compatible interface design offers a complementary path by aligning interaction semantics with agents' prior knowledge. In this paper, we revisit Nielsen's 10 usability heuristics through the lens of computer-use agents, identifying which principles naturally transfer, where implicit design assumptions create agent-specific failures, and how safe additive augmentations can improve robustness without harming human usability. To evaluate these ideas, we introduce UI-Verse, a suite of controlled environments built around functionally similar interfaces with different applied heuristics. Experiments show that our augmented heuristics consistently improve task completion and modestly improve efficiency, with combined heuristics yielding further gains. Human studies further show that these designs preserve the original interaction workflow without observable usability regressions. Overall, our findings highlight interface design as a practical complementary avenue for improving the reliability and generalization of computer-use agents without hurting users.

1 Introduction

As software interfaces become the front door to both everyday life and professional work, general computer-use agents (CUAs) are emerging as a new route to automation: systems that operate software directly, much like humans do (Agashe et al., 2024; Wang et al., 2025; Hu et al., 2025). Powered by large multimodal models, they are able to interpret the screen and take grounded actions to handle tasks like online shopping, web browsing, replying to emails, and even help with daily work (Xie et al., 2024a; Wang et al., 2024; Bonatti et al., 2024a; Pan et al., 2024; Rawles et al., 2025). However, because real world interfaces are continually evolving and highly dynamic, today's general CUAs still struggle to generalize to unseen UIs. In practice, they frequently rely on test-time scaling and trial-and-error to navigate new environments. They behave more like novices than professionals who operate through stable procedural knowledge (Abhyankar et al., 2025; Liu et al., 2026). This limitation is typically treated as a model capability problem. Yet generalization in interactive environments is not determined solely by the agent; it is also shaped by how interface states are represented, how actions are exposed, and how consistently interaction semantics are organized.

While improving agent architectures remains essential, we argue that interface design itself plays a critical and under-explored role in enabling reliable CUAs. Motivated by this, we revisit Nielsen's ten usability heuristics (Nielsen and Molich, 1990) for human-centered interface design and examine their implications for CUAs' perception, grounding, and control. Motivated by this perspective, we revisit Nielsen's ten usability heuristics (Nielsen and Molich, 1990) for human-centered interface design and examine their implications for computer-use agents' perception, grounding, and control. As indicated in Figure 1(Left), our analysis (Section § 3.1) reveals that existing heuristics are not uniformly aligned with the needs of automation with CUAs. We therefore categorize them into two types. The first group consists of directly transferable principles, where stronger adherence consistently benefits both users and agents. The second group includes principles that require CUA specific adaptations. These principles remain valuable for human users, but when automation is explicit, they require additional design considerations to ensure stable agent perception, reduce reasoning burden, and support reliable action grounding.

Based on this distinction, we provide two levels

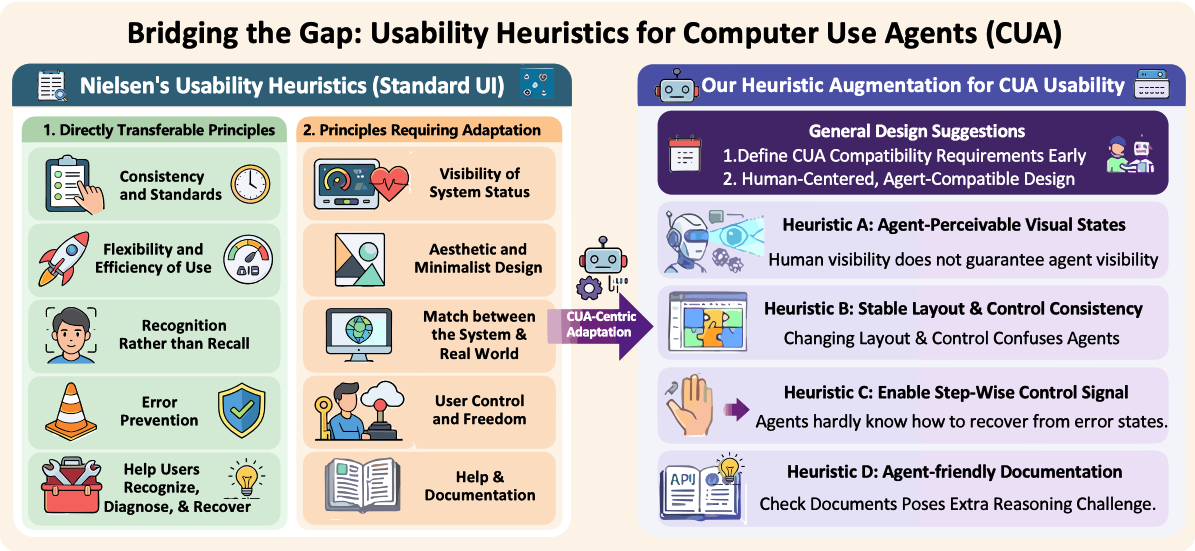

Figure 1: The left panel revisits Nielsen's 10 classical usability heuristics (Nielsen and Molich, 1990), originally proposed for *human* users, through the lens of general CUAs. Heuristics that benefit both humans and CUAs transfer directly (green), while those that may impair reliable CUA perception, reasoning, or grounding require adaptation (orange). The right panel presents our extension: two core design principles and four heuristics for designing interfaces that remain human-centered while better supporting reliable CUAs.

of interface design guidance. First, at the broader design level, we offer two general suggestions, summarized in Section § 3.2: (1) consider CUA compatibility early in the design process, and (2) adopt a human-centered yet agent-aware design perspective. Second, when computer-use agent automation is an explicit goal, we introduce four agent-specific heuristics as structured augmentations: (A) make visual states readily perceivable to agents, (B) maintain stable layouts and consistent controls, (C) provide step-wise control signals, and (D) expose agent-friendly documentation and explicit skills. Section § 3.3 presents the formal definitions, theoretical grounding, and concrete implementation examples for these heuristics.

To validate the impact of UI design on computer use and assess our proposed heuristics, we construct UI-Verse (Appendix A), a suite of controlled digital environments. These environments provide similar functionalities and comparable task goals, while differing in the usability heuristics reflected in their UI designs. We evaluate current CUAs in terms of both task success rate and efficiency, and further analyze their interaction trajectories by attributing the root causes of failed interactions. Our results show that UI design alone can substantially affect CUA performance. Across environments and design variants, interfaces aligned with our proposed heuristics consistently yield more reliable agent behavior, higher task success rates, and sometimes efficiency gains. When combined, these heuristics provide complementary benefits, leading to peak gains of up to 0.46 on related tasks. Our error-type analysis further explains the mechanisms behind these gains. Finally, human studies verify that these heuristic-based augmentations, while improving automation performance, do not necessarily compromise interface aesthetics, disrupt the original workflow, or degrade human usability.

Overall, our contributions are as follows: (1) We identify agent-compatible interface design as an important but underexplored challenge for general CUAs, and argue that robust automation depends not only on agent capability, but also on how interfaces expose states and actions. (2) We propose a human-centered, agent-compatible interface design framework with two general design suggestions and four agent-specific heuristics that extend classical usability principles to better support agent perception, grounding, and control. (3) We validate this framework through CUA evaluations and human studies, showing that these heuristics consistently improve CUA reliability and efficiency while preserving usability when designed properly.

2 Related Work

User-Centric Interface Design. User-centric interface design studies how interfaces help people perceive system state, act, and recover from errors. Foundational work introduced principles such as

visibility, consistency, error prevention, and recovery support (Norman, 1983; Nielsen and Molich, 1990; Nielsen, 1994). Related work on direct manipulation and minimalist design further emphasized continuous visual representations, incremental and reversible actions, and immediately observable effects, making interfaces more predictable and easier to control (Kang and Kim, 2007; Shneiderman, 1997). These ideas remain influential in AI-mediated settings. Human-centered AI emphasizes preserving human agency and helping users interpret and guide intelligent systems (Capel and Brereton, 2023), while later work extends classical design principles to generative AI applications, GUI-based machine learning tools, and AI-assisted design workflows (Weisz et al., 2024; Yamani et al., 2024; Lu et al., 2024). However, these efforts still primarily optimize interfaces for *human* users and do not systematically address settings where reliable operation by general UI-centered agents is also a core objective. Our work builds on this literature but shifts the focus toward human-centered, agent-compatible interface design.

General Computer-Use Agents. Recent work has substantially advanced general computer-use agents. Benchmarks such as OSWorld, OfficeBench, and Windows Agent Arena evaluate agent performance in realistic environments (Xie et al., 2024b; Wang et al., 2024; Bonatti et al., 2024b). Most prior work, however, focuses on improving the agent itself. UI-TARS and OpenCUA develop native GUI agents with screenshot-based perception, action modeling, and deliberate reasoning (Qin et al., 2025; Wang et al., 2025). Agent-S and CoAct further improve performance through stronger reasoning, memory, and Best-of-N sampling (Agashe et al., 2024; Song et al., 2025), while OSExpert shows that exploration-based learning can help agents acquire environment-specific skills (Liu et al., 2026). However, existing work largely treats the UI as fixed and places the burden of adaptation on the agent. In contrast, we treat interface design as an optimization target, shifting part of the burden to the environment to better support reliable human-agent collaboration.

3 From Classical Usability to Agent-Compatible Interface Design

Reliable automation with CUAs depends not only on agent capability, but also on whether interfaces expose states, actions, and transitions in ways that agents can consistently perceive, interpret, and execute. We therefore treat interface design not as a passive backdrop, but as an active factor shaping agent generalization. Our goal is not to replace classical human-centered usability principles, but to revisit and extend them for a setting in which interfaces are shared by both humans and agents, supporting effective collaboration while preserving human supervision. More broadly, we envision future digital workflows as human-centered and AI-assisted. To support this vision, we aim to design interfaces that agents can generalize to reliably while remaining understandable to humans and providing clear workflows for oversight. In this work, we proceed in three steps: first, we revisit Nielsen's ten usability heuristics through the lens of computer-use agents (Section § 3.1); second, we offer general design suggestions for considering automation goals and agent compatibility early in the design process (Section § 3.2); third, we operationalize these ideas into concrete CUA-specific heuristics that address agent perception, grounding, and control, enabling more robust generalization across environments (Section § 3.3).

3.1 Revisiting Nielsen's 10 Heuristics

Nielsen's ten usability heuristics provide a compact and influential framework for human-centered interface design, covering principles such as clarity, consistency, error prevention, and user control, with the full list shown in Figure 1 (Left). Although these heuristics were not originally developed for computer-use agents, we find that several transfer naturally to agent-compatible interface design, since interface properties that help humans understand interface states and follow effective workflows often also align with agents' priors and help them as well. In particular, we identify a subset of heuristics that are directly beneficial to CUAs. First, **consistency and standards** help establish shared structures across interfaces, including common layouts, controls, and descriptions, which support knowledge reuse and improve generalization across applications. Second, **flexibility and efficiency of use** also matter for agents, as efficient workflows reduce unnecessary interaction steps; this is especially important given that current CUAs still lag behind human experts in execution efficiency (Abhyankar et al., 2025; Liu et al., 2026). Third, **recognition rather than recall** is highly relevant because agents rely on limited context windows to maintain interaction history,

Original Heuristic	Reasoning for Heuristic Adaptation	Example Failure Mode
Visibility of System Status	Status visible to humans may be unreliable for CUAs, which perceive interfaces through discrete screenshots rather than continuous observation.	The UI signals a state through transient or hover feedback, which the agent misses and misinterprets.
Aesthetic and Minimalist Design	Visual simplicity and aesthetic design may reduce explicit cues that CUAs rely on for grounding, even when the UI remains intuitive for humans.	The UI uses icon-only controls or visually similar options, causing the agent to confuse and error.
Matching System and Real World	Human-friendly metaphors and conventions may not always align well with CUA learnt priors when interface semantics are implied by real-world analogies rather than stated explicitly.	The UI uses swiping to flip pages, following a physical book metaphor, but the agent fails to infer this convention and makes errors.
User Control and Freedom	Control and recovery mechanisms intuitive to humans may be inaccessible to CUAs, and multiple available workflows may confuse agents	The agent fails to find a recovery path, cannot use hidden shortcuts, and becomes stuck in an error state.
Help and Documentation	Searching for documentation adds to the agent's reasoning burden, making the task longer-horizon and more prone to potential cascading errors.	Navigating to the help page and further retrieve a complete and useful set of information is hard.

Table 1: Five usability heuristics that require adaptation for CUAs, along with the underlying reasoning and representative failure examples. Their original human-centered formulations do not fully address agent-specific constraints in perception, grounding, and control.

making explicit interface cues much easier to use than reconstructing prior states or procedures from memory. Finally, as with humans, **error prevention** and mechanisms that support **error diagnosis and recovery** help guide CUAs toward correct workflows and reduce avoidable failures. Taken together, these heuristics can be viewed as directly transferable principles that are broadly beneficial for CUA performance.

While several of Nielsen's heuristics align naturally with the needs of computer-use agents, others do not transfer as directly. CUAs interact with interfaces under constraints that differ fundamentally from those of human users, including discrete screenshot-based perception, bounded contextual memory, imperfect action grounding, and limited access to implicit shortcuts or recovery strategies. Under these constraints, interface properties that remain intuitive for humans may still introduce ambiguity or execution difficulty for agents. We therefore find that blindly applying some classical heuristics can create unintended barriers for CUA performance. As summarized in Table 1, these heuristics are not unhelpful in themselves; rather, their original human-centered formulations do not fully account for the perception, grounding, and control requirements of agent interaction. Across these examples, the central issue is that human usability often tolerates implicit cues, metaphor-based semantics, flexible control paths, and documentation-heavy support, whereas CUAs require more explicit, directly executable, and state-aligned interface signals. This observation motivates the augmented heuristics introduced next.

3.2 General Suggestions for Human-Centered, Agent-Compatible Interface Design

Interface design can support or hinder reliable CUA automation by shaping how clearly states, actions, and workflows are exposed. Classical usability heuristics may therefore need adaptation for agent operation. Based on this, we offer two general suggestions for interface designers.

Suggestion 1: Consider agent compatibility early when automation matters. When CUA-based automation is intended, agent compatibility should be considered early in the design process. This involves asking whether important states are observable, whether actions are easy to ground, and whether workflows remain stable enough for reliable execution.

Suggestion 2: Keep interface design human-centered while remaining agent-compatible. In most applications, CUAs are meant to assist humans rather than fully replace them. Humans should therefore remain able to understand the workflow, supervise agent behavior, and intervene when needed. We advocate a human-centered, agent-compatible design philosophy: interfaces

should remain readable and trustworthy for humans while also exposing enough explicit structure for reliable agent interaction.

3.3 Operationalizing Agent-Compatible Design: CUA-Specific Heuristics

Guided by the observations and suggestions above, we propose four CUA-specific usability heuristics that complement classical human-centered interface design. These heuristics are not intended to replace Nielsen's framework; rather, they serve as structured augmentations for settings where reliable software automation is an explicit goal. Each heuristic highlights interface properties that can reduce exploration burden, improve action grounding, and support more robust agent behavior, even in previously unseen environments.

Augmented Interface Usability Heuristic A

Design Agent-Perceivable Visual States.

Interfaces should expose visual states in ways that remain reliably perceivable to CUAs throughout task execution. Critical states and semantics should remain directly visible in the current interface state, even under discrete screen observations, rather than being conveyed through subtle, transient, or difficult-to-detect cues. More broadly, important visual states should always remain explicit, stable, and semantically clear for agent interaction.

Augmented Interface Usability Heuristic B

Stable Layout and Control Consistency.

Interfaces should maintain predictable layouts and consistent action semantics across similar contexts. CUAs struggle when controls shift position, change appearance, or use inconsistent labels. This may also happen when the control is designed from a physical world metaphor which the agent hardly learned. These scenarios force repeated rediscovery of action mappings. Stable layout and control consistency reduce this burden by allowing learned interaction patterns to transfer across subtasks and interface states. More broadly, this extends classical consistency and standards from semantic consistency to procedural regularity in how controls are organized and exposed for execution.

Augmented Interface Usability Heuristic C

Agent-Attainable Step-Wise Control Signals.

Interfaces should expose clear and attainable control signals that allow agents to move forward, go back, or recover at each stage of a workflow. For CUAs, control should rely on explicit interface actions rather than hidden shortcuts, implicit gestures, or recovery paths inferred from context. While multiple workflow options may benefit human flexibility, they can also increase ambiguity for agents. A single, explicit control path therefore makes workflows easier for CUAs to follow, verify, and recover within.

Augmented Interface Usability Heuristic D

Agent-Friendly Docs and Skill Exposure.

Documentation access in digital environments should follow a simple and explicit workflow, imposing minimal grounding, perception, and reasoning burden on agents. More importantly, procedures that agents are expected to use repeatedly should be exposed as explicit, operable skills rather than only as free-form help text. Structured in this way, plain documentation becomes reusable step-by-step procedural knowledge, making the correct operations easier to retrieve, execute, and verify while reducing unnecessary exploration.

4 Experiments

4.1 Experimental Setup

We evaluate if our proposed heuristics for CUAs can improve agent performance without sacrificing human usability. To support this, we introduce UI-Verse (Appendix A), a suite of controlled environments built around functionally similar interfaces that support comparable task goals. Each evaluation group contains two interfaces constructed with different sets of usability heuristics. See an example in Figure 2. We construct baseline UIs that implement Nielsen's ten usability heuristics while excluding our proposed augmented heuristics. We then build revised UIs that incorporate one or more of our proposed heuristics to better support reliable computer-use agents. For each evaluation group, we manually design computer-use tasks that depend on the target UI regions affected. We then compare agent task success, efficiency, and the distribution of error types in CUA trajectories across the two interface variants.

Models For each evaluation group, we test the performance of several computer-use agents, including: (1) agents built on general VLMs with UI grounding capability, such as Qwen-3-VL (Bai

Methods →		Agent Performance		Trajectory Error Attribution		
Models ↓	**Eval ↓**	**Success**	**Efficiency**	**Perception**	**Grounding**	**Reasoning**
Results w/ and w/o Heuristic A: Agent-Perceivable Visual States						
Qwen3-VL	Baseline UI	0.13 ± 0.02	102 ± 11	0.09 ± 0.00	0.62 ± 0.00	0.15 ± 0.01
	Revised UI	$\mathbf{0.31} \pm \mathbf{0.01}$	$\mathbf{88} \pm \mathbf{4}$	0.04 ± 0.00	0.32 ± 0.01	0.33 ± 0.02
OpenCUA	Baseline UI	0.12 ± 0.00	229 ± 24	0.09 ± 0.00	0.37 ± 0.03	0.43 ± 0.03
	Revised UI	$\mathbf{0.32} \pm \mathbf{0.00}$	$\mathbf{157} \pm \mathbf{6}$	0.04 ± 0.00	0.15 ± 0.02	0.50 ± 0.02
Agent-S3	Baseline UI	0.26 ± 0.04	268 ± 46	0.07 ± 0.00	0.42 ± 0.02	0.25 ± 0.01
	Revised UI	$\mathbf{0.49} \pm \mathbf{0.03}$	$\mathbf{193} \pm \mathbf{23}$	0.03 ± 0.00	0.12 ± 0.00	0.37 ± 0.02
Results w/ and w/o Heuristic B: Stable Layout and Control Consistency						
Qwen3-VL	Baseline UI	0.26 ± 0.04	139 ± 5	0.07 ± 0.00	0.70 ± 0.05	0.09 ± 0.01
	Revised UI	$\mathbf{0.38} \pm \mathbf{0.03}$	$\mathbf{138} \pm \mathbf{2}$	0.06 ± 0.02	0.52 ± 0.02	0.04 ± 0.00
OpenCUA	Baseline UI	0.13 ± 0.03	$\mathbf{178} \pm \mathbf{57}$	0.07 ± 0.00	0.28 ± 0.02	0.53 ± 0.05
	Revised UI	$\mathbf{0.23} \pm \mathbf{0.03}$	210 ± 16	0.09 ± 0.00	0.30 ± 0.10	0.38 ± 0.08
Agent-S3	Baseline UI	0.18 ± 0.10	220 ± 86	0.08 ± 0.01	0.47 ± 0.03	0.28 ± 0.09
	Revised UI	$\mathbf{0.45} \pm \mathbf{0.21}$	$\mathbf{149} \pm \mathbf{29}$	0.08 ± 0.02	0.37 ± 0.01	0.18 ± 0.11

Table 2: Agent performance and trajectory error attribution on baseline UIs designed under Nielsen's 10 usability heuristics and revised UIs augmented with Heuristic A and Heuristic B. We show that meticulously designed interface themselves are effectively shaping robust CUA behavior in perception, grounding, and reasoning.

et al., 2025); (2) agents built on specialized computer-use models fine-tuned on computer-use datasets, such as OpenCUA (Wang et al., 2025); and (3) Agent-S framework (Gonzalez-Pumariega et al., 2025) that incorporate both a planning module, supported by GPT-5.4, and a grounding module, based on UI-TARS. We do not allow best-of-N sampling at inference time, set the maximum number of interaction steps to 30, and use an inference temperature of 1.0 for all models. We serve all open-source models locally with vLLM on a single 80GB GPU. Our environments are implemented on virtual machines and share the same codebase foundation as OSWorld (Xie et al., 2024b).

Evaluation Metrics We measure agent performance and efficiency using average success rate and average execution time (in seconds) across multiple runs under different evaluation settings and UI design variants. To better understand how the proposed usability heuristics affect agent behavior, we also conduct trajectory-level error attribution. In this stage, a strong LLM serves as an automated evaluator that examines the intermediate interface states and interaction trajectory of each failed run, and classifies the failure as a perception, grounding, or reasoning error. We then report the proportion of each error type among all interaction trajectories to compare failure patterns across evaluation groups.

4.2 Main Results

Consistent performance gains from pure UI design shifts. Tables 2 and 3 present the results for individual heuristic augmentations. The main finding is that improving agent compatibility through UI design alone can substantially improve CUA performance on selected tasks. Across models and agentic frameworks, applying any single one of our proposed heuristics consistently increases task success rate relative to the baseline UI. The largest absolute gain reaches 35% for Qwen3-VL under Heuristic D, showing that improvement can be substantial. In addition to higher success rates, we also observe efficiency gains in most settings. A likely explanation is that improved interface design helps agents complete tasks more directly, rather than becoming trapped in error states and repeatedly exploring until the maximum interaction budget is exhausted. Overall, these results suggest that UI design itself is a strong and underexplored lever for improving the robustness and efficiency of computer-use agents, and that our proposed heuristics are practically useful for supporting more reli-

Methods →		Agent Performance		Trajectory Error Attribution		
Models ↓	**Eval ↓**	**Success**	**Efficiency**	**Perception**	**Grounding**	**Reasoning**
Results w/ and w/o Heuristic C: Enable Step-wise Control Signal						
Qwen3-VL	Baseline UI	0.10 ± 0.00	121 ± 7	0.09 ± 0.01	0.71 ± 0.05	0.10 ± 0.04
	Revised UI	$\mathbf{0.30} \pm \mathbf{0.00}$	$\mathbf{64} \pm \mathbf{4}$	0.07 ± 0.00	0.51 ± 0.00	0.12 ± 0.00
OpenCUA	Baseline UI	0.15 ± 0.07	$\mathbf{224} \pm \mathbf{8}$	0.08 ± 0.01	0.26 ± 0.07	0.52 ± 0.03
	Revised UI	$\mathbf{0.35} \pm \mathbf{0.07}$	243 ± 28	0.05 ± 0.01	0.32 ± 0.05	0.28 ± 0.09
Agent-S3	Baseline UI	0.20 ± 0.00	$\mathbf{212} \pm \mathbf{5}$	0.08 ± 0.01	0.40 ± 0.00	0.33 ± 0.01
	Revised UI	$\mathbf{0.40} \pm \mathbf{0.14}$	224 ± 24	0.05 ± 0.02	0.38 ± 0.10	0.17 ± 0.02
Results w/ and w/o Heuristic D: Agent-friendly Documentation and Skill Exposure						
Qwen3-VL	Baseline UI	0.03 ± 0.00	155 ± 3	0.10 ± 0.00	0.70 ± 0.01	0.18 ± 0.01
	Revised UI	$\mathbf{0.38} \pm \mathbf{0.00}$	$\mathbf{86} \pm \mathbf{7}$	0.08 ± 0.00	0.47 ± 0.01	0.07 ± 0.01
OpenCUA	Baseline UI	0.08 ± 0.04	264 ± 1	0.08 ± 0.00	0.37 ± 0.01	0.46 ± 0.01
	Revised UI	$\mathbf{0.23} \pm \mathbf{0.00}$	$\mathbf{207} \pm \mathbf{1}$	0.11 ± 0.00	0.37 ± 0.03	0.29 ± 0.03
Agent-S3	Baseline UI	0.18 ± 0.04	230 ± 84	0.08 ± 0.01	0.44 ± 0.02	0.30 ± 0.00
	Revised UI	$\mathbf{0.41} \pm \mathbf{0.04}$	$\mathbf{195} \pm \mathbf{17}$	0.08 ± 0.00	0.39 ± 0.06	0.12 ± 0.04

Table 3: Agent performance and trajectory error attribution on baseline UIs designed under Nielsen's 10 usability heuristics and revised UIs augmented with Heuristic C and Heuristic D. Visible improvements are consistent

able agent interaction.

Joint and complementary benefits from combining all proposed heuristics. Table 4 reports the results when all four augmented heuristics are implemented together. All evaluated agents achieve substantial improvements over the baseline UI. For example, Qwen3-VL improves from 0.13 to 0.59, corresponding to an absolute gain of 46% and roughly a 3.5× improvement over the baseline, while Agent-S3 improves from 0.21 to 0.61, corresponding to an absolute gain of 40%. These joint gains are also larger than those achieved under any single heuristic alone, indicating that the four heuristics provide complementary benefits when combined. They are further accompanied by large reductions in execution time, as well as clear decreases in grounding and reasoning errors, with perception errors also decreasing more modestly. Overall, this pattern suggests that the heuristics target different failure modes in agent interaction and therefore work better together than in isolation.

Error-type shifts with different heuristics. The trajectory-level error attribution results in Tables 2–4 show that different heuristics reduce different failure types, in ways that largely match their intended roles. Heuristic A most consistently improves grounding, with smaller gains in perception, suggesting that clearer visual states help agents recognize interface state and ground actions correctly. Heuristic B and Heuristic C mainly reduce reasoning errors, indicating that stable layouts, consistent controls, and step-wise interaction signals help agents plan, follow, and recover. Heuristic D yields the strongest reduction in reasoning errors, while also improving grounding in several cases, suggesting structured documents and explicit skill exposure reduce procedural burden.

4.3 Human Studies

We conduct comprehensive human studies with two goals: (1) to evaluate whether our agent-compatible heuristics preserve human usability, and (2) to assess the reliability of our automatic trajectory error analyzer. Specifically, we examine whether the revised interfaces remain effective for human users, while also measuring how closely the analyzer's judgments align with human annotations on error attribution. Additional details on annotators, experimental configurations, and more comprehensive results are provided in Appendix B.

Human-Centered Evaluation of Revised Interfaces In line with our general suggestion in Section §3.2, we conduct human studies to examine

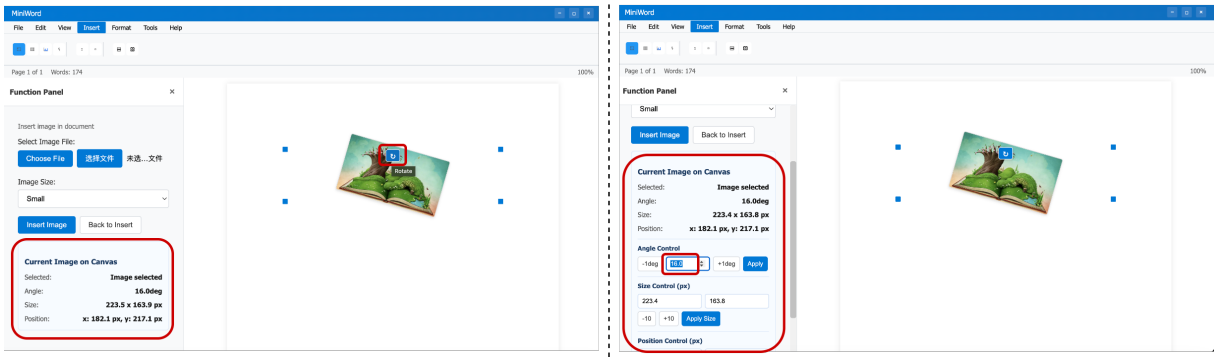

Figure 2: **Left:** Baseline UI; **Right:** Revised UI incorporating our augmented usability heuristic B. While the original UI requires direct dragging to mimic rotation or translation, the revised UI adopts a layout that provides explicit controls for rotating and repositioning the target image, simplifying the action space for CUAs. More UI design examples following our heuristics that helps CUAs can be found in Appendix A.

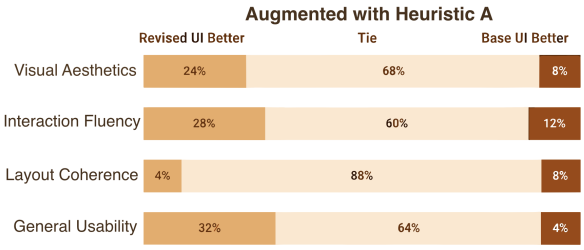

Figure 3: Sample of human evaluation results on whether the augmented interface affects general usability. The full human evaluation across all four heuristic augmentations is shown in Figure 4 in Appendix B.

whether the proposed heuristics preserve interface quality for human users, rather than improving agent performance at the expense of human experience. Annotators compare randomly sampled screenshots and standard interaction trajectories from selected tasks under both the baseline and revised interfaces along four dimensions: **visual aesthetics**, **interaction fluency**, **layout coherence**, and **perceived overall usability**. For each task, they are shown six screenshots from the baseline interface and six corresponding screenshots from the revised interface, and judge whether the revised UI is better, the baseline UI is better, or whether the two are tied on each dimension. As shown in Figure 3, most judgments are ties, indicating that the augmented designs generally preserve human usability. In some cases, especially under Heuristic A, annotators slightly prefer the revised UI. Overall, these results suggest that our augmented heuristics improve support for CUAs without meaningfully harming human experience.

Reliability of Automatic Error Analysis We further evaluate the reliability of our automatic trajectory error analyzer by comparing its judgments with human. Human evaluators inspect failed interaction trajectories and estimate the extent to which each failure is primarily caused by perception, grounding, or reasoning errors. As shown in Table 6 (Appendix B), the automatic analyzer produces error categorizations that are broadly consistent with human judgments across all four heuristics. In particular, both human and automatic evaluations reveal similar distribution shifts in how revised interfaces reduce different types of agent errors, supporting our core findings in Section 4.2.

5 Conclusion

In this paper, we argue that improving CUAs needn't be treated solely as a model-side problem, but also as a problem of agent-compatible UI design. From this perspective, we revisit Nielsen's usability heuristics through the lens of CUAs, identify where classical human-centered design transfers and where adaptation is needed, and propose a human-centered, agent-compatible framework with two general design suggestions and four concrete heuristics. To study this question, we introduce UI-Verse, a controlled benchmark for analyzing how interface design affects agent behavior, and evaluate multiple CUAs on interfaces with and without our augmented heuristics. Results show that agent-compatible interfaces can consistently improve task success and often efficiency, while human studies suggest these gains can be achieved without usability regressions. Overall, we view this work as an initial step toward co-designed digital environments that are usable for humans and dependable for CUAs, enabling more robust automation and more effective human-AI collaboration in both everyday and professional workflows.

Reproducibility Statement

In Section §4.1, we describe our experimental setup in detail, including the model configurations, computational resource requirements, and access to the models used in our study. We also report the core implementation details needed to reproduce our experiments. To promote transparency and enable future extensions of this work, we will publicly release the codebase and data upon acceptance. These resources are intended to support the replication of our main results and provide a foundation for subsequent research.

Limitations

This work has several limitations. Although UI-Verse provides controlled environments for comparing functionally similar interfaces, the environments are still simpler than large-scale real-world software ecosystems. Real applications often involve more complex workflows, richer visual states, heterogeneous interaction patterns, and highly dynamic interface changes. It therefore remains unclear whether the observed benefits of our augmented heuristics will fully generalize to broader and more complex software settings. In addition, our current evaluation covers a limited set of tasks and interface variants, which may not capture the full diversity of real user goals, software domains, and interface designs. Future work should evaluate agent-compatible interface design in larger, more realistic, and more dynamic applications.

References

Reyna Abhyankar, Qi Qi, and Yiying Zhang. 2025. Osworld-human: Benchmarking the efficiency of computer-use agents. *Preprint*, arXiv:2506.16042.

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

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, Lawrence Jang, and Zack Hui. 2024a. Windows agent arena: Evaluating multi-modal os agents at scale. *Preprint*, arXiv:2409.08264.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, and 1 others. 2024b. Windows agent arena: Evaluating multi-modal os agents at scale. *arXiv preprint arXiv:2409.08264*.

Tara Capel and Margot Brereton. 2023. What is human-centered about human-centered ai? a map of the research landscape. In *Proceedings of the 2023 CHI conference on human factors in computing systems*, pages 1–23.

GitHub. 2026a. About github copilot coding agent. https://docs.github.com/copilot/concepts/agents/coding-agent/about-coding-agent. Official documentation, accessed 2026-03-31.

GitHub. 2026b. Model context protocol (mcp) and github copilot coding agent. https://docs.github.com/en/copilot/concepts/agents/coding-agent/mcp-and-coding-agent. Official documentation, accessed 2026-03-31.

Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. 2025. The unreasonable effectiveness of scaling agents for computer use. *Preprint*, arXiv:2510.02250.

Md Mehedi Hasan and 1 others. 2025. Model context protocol (mcp) at first glance. *arXiv preprint arXiv:2506.13538*.

HKUDS. 2026. Cli-anything: Making all software agent-native. https://github.com/HKUDS/CLI-Anything. GitHub repository, accessed 2026-03-31.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shenzhi Wang, Xinchen Xu, Shuofei Qiao, Zhaokai Wang, Kun Kuang, Tieyong Zeng, Liang Wang, and 10 others. 2025. Os agents: A survey on mllm-based agents for general computing devices use. *Preprint*, arXiv:2508.04482.

Seonghoon Kang and Won Kim. 2007. Minimalist and intuitive user interface design guidelines for consumer electronics devices. *Journal of Object Technology*, 6(3):39–52.

Jiateng Liu, Zhenhailong Wang, Rushi Wang, Bingxuan Li, Jeonghwan Kim, Aditi Tiwari, Pengfei Yu, Denghui Zhang, and Heng Ji. 2026. Osexpert: Computer-use agents learning professional skills via exploration. *arXiv preprint arXiv:2603.07978*.

Yuwen Lu, Tiffany Knearem, Shona Dutta, Jamie Blass, Clara Kliman-Silver, and Frank Bentley. 2024. Ai is not enough: A hybrid technical approach to ai adoption in ui linting with heuristics. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Microsoft. 2026a. Agent flows overview - microsoft copilot studio. https://learn.microsoft.com/en-us/microsoft-copilot-studio/flows-overview. Official documentation, accessed 2026-03-31.

Microsoft. 2026b. Knowledge sources summary - microsoft copilot studio. https://learn.microsoft.com/en-us/microsoft-copilot-studio/knowledge-copilot-studio. Official documentation, accessed 2026-03-31.

Microsoft. 2026c. What is microsoft copilot studio? https://learn.microsoft.com/en-us/microsoft-copilot-studio/fundamentals-what-is-copilot-studio. Official documentation, accessed 2026-03-31.

Model Context Protocol. 2025. Model context protocol. https://modelcontextprotocol.io/docs/getting-started/intro. Official documentation, accessed 2026-03-31.

Jakob Nielsen. 1994. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 152–158.

Jakob Nielsen and Rolf Molich. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256.

Donald A Norman. 1983. Design principles for human-computer interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 1–10.

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

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, and 16 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *Preprint*, arXiv:2501.12326.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. 2025. Androidworld: A dynamic benchmarking environment for autonomous agents. *Preprint*, arXiv:2405.14573.

Ben Shneiderman. 1997. Direct manipulation for comprehensible, predictable and controllable user interfaces. In *Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 33–39.

Linxin Song, Yutong Dai, Viraj Prabhu, Jieyu Zhang, Taiwei Shi, Li Li, Junnan Li, Silvio Savarese, Zeyuan Chen, Jieyu Zhao, Ran Xu, and Caiming Xiong. 2025. Coact-1: Computer-using agents with coding as actions. *arXiv preprint*.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, and 1 others. 2025. Opencua: Open foundations for computer-use agents. *Preprint*, arXiv:2508.09123.

Zilong Wang, Yuedong Cui, Li Zhong, Zimin Zhang, Da Yin, Bill Yuchen Lin, and Jingbo Shang. 2024. Officebench: Benchmarking language agents across multiple applications for office automation. *Preprint*, arXiv:2407.19056.

Justin D Weisz, Jessica He, Michael Muller, Gabriela Hoefer, Rachel Miles, and Werner Geyer. 2024. Design principles for generative ai applications. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–22.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024a. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. 2024b. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Preprint*, arXiv:2404.07972.

Asma Z Yamani, Haifa Abdullah Al-Shammare, and Malak Baslyman. 2024. Establishing heuristics for improving the usability of gui machine learning tools for novice users. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–19.

A UI-Verse with Augmented Usability Heuristics

In this section, we introduce UI-Verse, a suite of controlled environments built around functionally similar interfaces that support comparable task goals. Within each evaluation group, UI-Verse includes two interfaces constructed with different sets of usability heuristics. By evaluating CUAs on the same tasks across these interface variants, we can quantitatively examine how interface usability heuristics affect agent performance. We provide the list of implemented unit functionalities for environments in UI-Verse in Section A.1, and present exemplar UI designs in Section A.2 to illustrate how our proposed heuristics may map to real-world interface design choices.

A.1 List of Functionalities Implemented

In our experiments, we implement 63 unit functionalities for document processing in MiniWord. All UI-Verse environments preserve the same underlying functionality set, while allowing the interface realization to vary according to different usability heuristics. This design ensures that performance differences arise from interface design rather than functional disparity. Table 5 summarizes all implemented unit functionalities. We leave the extension toward broader and more complex functionality coverage to future work.

A.2 Augmented Heuristics in Interface Design

Although each of our augmented heuristics (A–D) can be implemented in multiple ways, we provide representative examples to illustrate how baseline interfaces can be revised to better support agent interaction. Figures 5, 6, and 7 present example pairs of baseline and revised UI designs from our UI-Verse environments. The key interface changes are highlighted with red boxes in the figures, while the individual captions further explain the specific modifications and why the revised designs are more helpful for agents.

B Additional Results for Human Studies

B.1 Experimental Settings

Four volunteer researchers with backgrounds in computer science and AI participated in our human studies. We designed separate annotation interfaces for the two experiments, as shown in Figure 8 and Figure 9.

In the first study, annotators were first introduced to the definitions of the evaluation dimensions for human-centered UI quality. They were then presented with six screenshots from the baseline interface and six corresponding screenshots from the revised interface, and asked to compare the two interfaces and judge which one better matched their expectations along each dimension. The annotation interface supports efficient switching across screenshots and provides shortcuts for quickly toggling between the two interfaces.

In the second study, annotators were first introduced to the definitions of erroneous trajectories and the three error categories. They then examined the interaction trajectories between CUAs and the interfaces, including the agent's intermediate planning, reasoning, actions, and the resulting UI changes over time. Based on the full trajectory, annotators used the interface to assign a probability distribution over the three error types: perception, grounding, and reasoning. The annotation interface supports efficient navigation across screenshots and provides shortcuts for jumping directly to the final stage. The agent's reasoning and thoughts are presented in natural language in a terminal panel.

B.2 More Detailed Experimental Results

Due to space limitations, we provide the detailed results for the second part of the human study in Table 6. These results show that although human judgments and the automatic analyzer may differ slightly in absolute values, they remain broadly consistent overall. In particular, both exhibit similar distribution shifts under our usability heuristics, further supporting the reliability of the automatic analysis.

C Discussions

Scope and boundary of our setting. Our work focuses on a particular and practically important stream of computer-use agents: general, UI-centered agents that operate existing software through the same interface surface shared with human users. This framing is aligned with our paper's core setting, where interfaces are treated as a shared medium for both humans and agents, with an emphasis on preserving human supervision, interpretability, and intervention when needed. In this sense, our study is most relevant to workflows in which software remains human-facing and agents must generalize through the visible interface,

Methods →		Agent Performance		Trajectory Error Attribution		
Models ↓	**Eval ↓**	**Success**	**Efficiency**	**Perception**	**Grounding**	**Reasoning**
Results w/ and w/o All Augmented Heuristics A-D						
Qwen3-VL	Baseline UI	0.13 ± 0.02	129 ± 7	0.09 ± 0.00	0.68 ± 0.03	0.10 ± 0.02
	Revised UI	$\mathbf{0.59} \pm \mathbf{0.03}$	$\mathbf{62} \pm \mathbf{5}$	0.05 ± 0.00	0.28 ± 0.02	0.08 ± 0.02
OpenCUA	Baseline UI	0.12 ± 0.04	224 ± 23	0.08 ± 0.00	0.32 ± 0.03	0.48 ± 0.03
	Revised UI	$\mathbf{0.54} \pm \mathbf{0.04}$	$\mathbf{149} \pm \mathbf{11}$	0.04 ± 0.00	0.20 ± 0.02	0.22 ± 0.03
Agent-S3	Baseline UI	0.21 ± 0.05	280 ± 55	0.08 ± 0.01	0.43 ± 0.02	0.28 ± 0.03
	Revised UI	$\mathbf{0.61} \pm \mathbf{0.04}$	$\mathbf{140} \pm \mathbf{16}$	0.05 ± 0.00	0.20 ± 0.02	0.14 ± 0.03

Table 4: Agent performance and trajectory error attribution on baseline UIs and revised UIs augmented with all proposed heuristics A–D. Combining all heuristics yields the strongest overall improvements, suggesting that the proposed heuristics provide complementary benefits for supporting reliable CUA interaction.

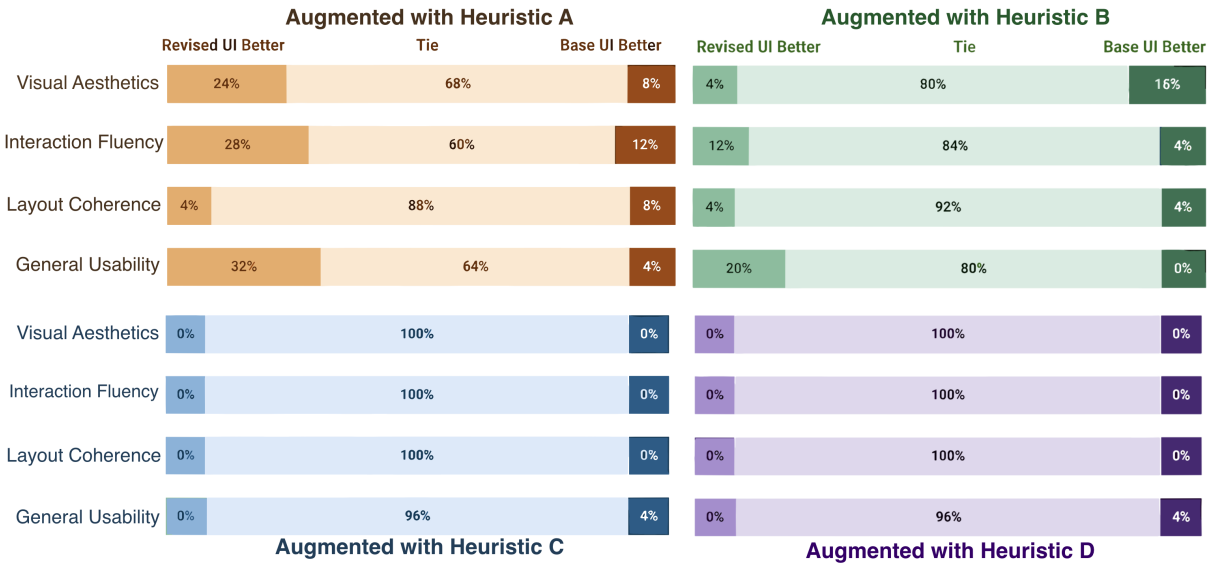

Figure 4: Full human evaluation of revised interfaces under all four heuristic augmentations. Across all four revised UIs, annotators overwhelmingly judge the revised and baseline interfaces as comparable on visual aesthetics, interaction fluency, layout coherence, and general usability. This indicates that our agent-oriented revisions improve CUA compatibility without materially degrading human usability.

rather than through privileged back-end access.

Automation-native software beyond graphical interfaces. We acknowledge that UI-based interaction represents only one stream within the broader agentic software landscape. An important alternative direction is automation-native software, in which systems are designed from the outset for direct machine interaction through APIs, protocols, or structured tool interfaces rather than human-facing graphical UIs. Recent developments point toward this paradigm. The Model Context Protocol (MCP), for example, standardizes how AI systems connect to external tools and data sources, enabling more direct and structured integration beyond GUI operation (Model Context Protocol, 2025; Hou et al., 2025; Hasan et al., 2025). Similarly, CLI-Anything advocates making software directly agent-native through command-line-style interfaces, thereby reducing reliance on fragile screen-based interaction (HKUDS, 2026). Our paper is not primarily targeted at these non-UI-first workflows. Instead, we focus on the complementary regime in which agents must still operate through existing graphical interfaces shared with human users. More broadly, we believe that even as AI agents become increasingly capable, human supervision will remain important in many real-world workflows, especially when trustworthiness, safety, and accountability are critical. In such settings,

All functionalities imeplmented in ENvironements of UI-Verse						
New Doc	Open Doc	Save Doc	Print Doc	Refresh Page	Close Page	New Tab
Pagination	Undo	Redo	Cut	Direct Copy	Format Copy	Paste
Find Text	Replace	Clear Format	Painter	Zoom In	Zoom Out	Print Preview
Page Setup	Toggle Bar	Toggle Ruler	Hide	Focus Mode	Night Mode	Insert Image
Insert Table	Insert Chart	Insert Link	Equation	Insert Symbol	Section Break	Use Template
Bold	Italic	Underline	Strikethrough	Left Align	Center Align	Right Align
Justify	Text Color	Highlight	Format Paint	Clear Format	Superscript	Subscript
Line Spacing	Page Setup	Header Footer	Spell Check	Comment	Bulleted List	Numbere List
Add Indent	Cut Indent	Count	Settings	Page Break	Share Doc	Help Doc

Table 5: All implemented unit functionalities in MiniWord. Each cell corresponds to one functionality shared across all UI-Verse environments. While interface layouts may vary under different usability heuristics, the underlying functionality set is kept fixed.

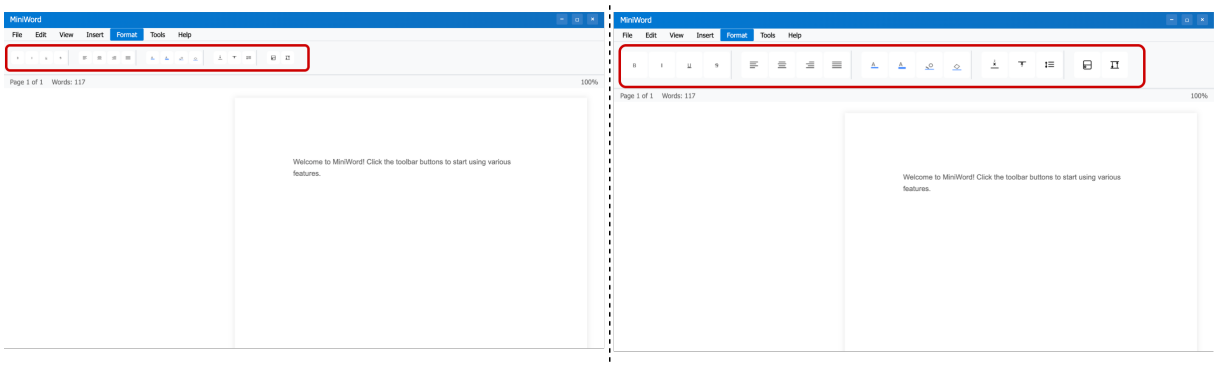

Figure 5: **Left:** Baseline UI; **Right:** Revised UI incorporating our augmented usability heuristics A. Specifically, the revised UI only uses larger buttons for better agent visibility.

interfaces that are understandable to human users remain necessary not only to support oversight and intervention, but also to reduce cognitive burden and make collaboration with agents more practical.

Vertically integrated copilots within proprietary software ecosystems. A second important boundary case is domain-specific copilots developed by organizations that already own the product, source code, internal APIs, and enterprise data boundary. In such settings, firms can build much stronger task-specific assistants by grounding them directly in proprietary system structure, workflows, permissions, and organizational knowledge. This pattern is increasingly visible in practice. Google has embedded Gemini directly into Workspace applications such as Gmail, Docs, Sheets, Slides, Drive, and Chat. Adobe has introduced AI assistants across its software stack, including Acrobat, Express, and Photoshop. Similar trends also appear in Microsoft Copilot Studio and GitHub Copilot, where assistants can leverage enterprise knowledge, structured tool integrations, and repository-level context (Microsoft, 2026c,a,b; GitHub, 2026a,b). In such vertically integrated scenarios, agent capability may depend much more on privileged system integration than on generalization through visible interfaces alone. As a result, changing the UI itself may have a more limited effect on performance than in the setting we study. In contrast, our work focuses on general computer-use agents that operate openly accessible software through existing graphical interfaces, without relying on privileged internal access or platform-specific integration. We see this as a complementary path toward making AI assistance more broadly available, since it allows users to benefit from automation across diverse software environments rather than only within closed proprietary ecosystems. In this sense, our aim is to support a more open and widely usable agent ecosystem in which the convenience of AI can be shared across applications and users.

Business incentives and barriers to open computer-use agents. These distinctions also suggest a broader strategic implication. Companies that control both the software stack and the agent layer may have stronger incentives to invest in tightly integrated proprietary copilots than to optimize their products for open, third-party gen-

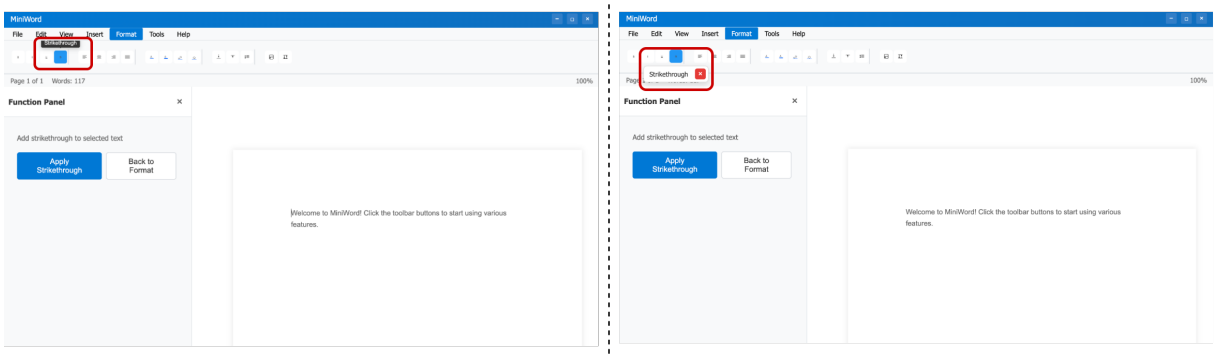

Figure 6: **Left:** Baseline UI; **Right:** Revised UI incorporating our augmented usability heuristic A. Specifically, the revised UI avoids hover-only textual descriptions of buttons, making functionality labels visible and reducing agent confusion about button meanings.

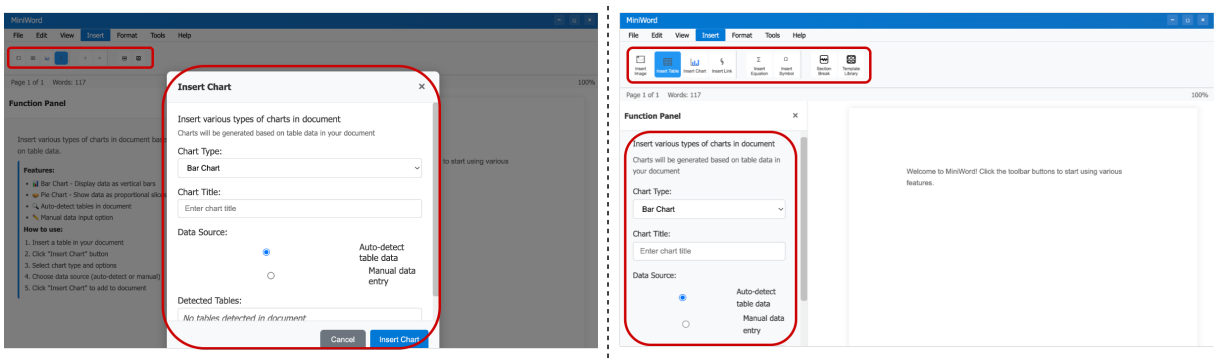

Figure 7: **Left:** Baseline UI; **Right:** Revised UI incorporating all of our augmented usability heuristics for CUAs. Specifically, the revised UI uses larger buttons, adds text-based functionality descriptions to improve agent visibility, and adopts a more consistent layout by always using sidebars. Beyond the visible interface changes, the revised environment also provides more consistent controls, agent-executable shortcuts for directly returning to the previous state, and a documented skill set that can be leveraged by CUAs.

eral computer-use agents. Such integration can improve product differentiation, strengthen ecosystem lock-in, and help firms capture a larger share of the value created by AI-assisted workflows within their own platforms. Under these conditions, firms may have limited incentive to expose stable external interfaces, interoperable automation pathways, or agent-friendly UI structures for general-purpose third-party agents. As a result, parts of the software ecosystem may evolve toward greater vertical integration and higher access barriers for open, cross-application CUAs. Our work should therefore not be interpreted as a universal claim about all automation settings. Instead, it focuses on an important and underexplored regime: open, heterogeneous, cross-application environments in which software remains human-facing, agents lack privileged internal access, and interface design still materially shapes reliability, oversight, and generalization. While we propose interface heuristics that support more reliable general computer-use agents, we also acknowledge that these usability of heuristics for interface design can be used in the opposite direction. to privilege proprietary agents and disadvantage open, third-party ones. We hope this work can help motivate a more open software ecosystem in which the availability and convenience of AI are shared broadly across users and applications.

Augmentation →	Heuristic A						Heuristic B					
Error Type →	Perception		Grounding		Reasoning		Perception		Grounding		Reasoning	
Eval ↓	Auto	Human	Auto	Human	Auto	Human	Auto	Human	Auto	Human	Auto	Human
Baseline UI	0.07	0.15	0.42	0.33	0.25	0.26	0.08	0.14	0.47	0.36	0.28	0.33
Revised UI	0.03	0.08	0.12	0.11	0.37	0.33	0.08	0.11	0.37	0.27	0.18	0.25

Augmentation →	Heuristic C						Heuristic D					
Error Type →	Perception		Grounding		Reasoning		Perception		Grounding		Reasoning	
Eval ↓	Auto	Human	Auto	Human	Auto	Human	Auto	Human	Auto	Human	Auto	Human
Baseline UI	0.08	0.12	0.44	0.38	0.30	0.32	0.08	0.13	0.43	0.34	0.28	0.32
Revised UI	0.08	0.10	0.39	0.29	0.12	0.20	0.05	0.09	0.20	0.16	0.14	0.14

Table 6: Human validation of automatic trajectory error attribution under different interface heuristics, based on trajectories generated by the Agent-S framework. While the automatic evaluator and human annotators do not always produce identical absolute error proportions, they exhibit strong agreement in overall error patterns. Notably, both capture similar shifts in the distribution of perception, grounding, and reasoning errors introduced by different interface heuristics.

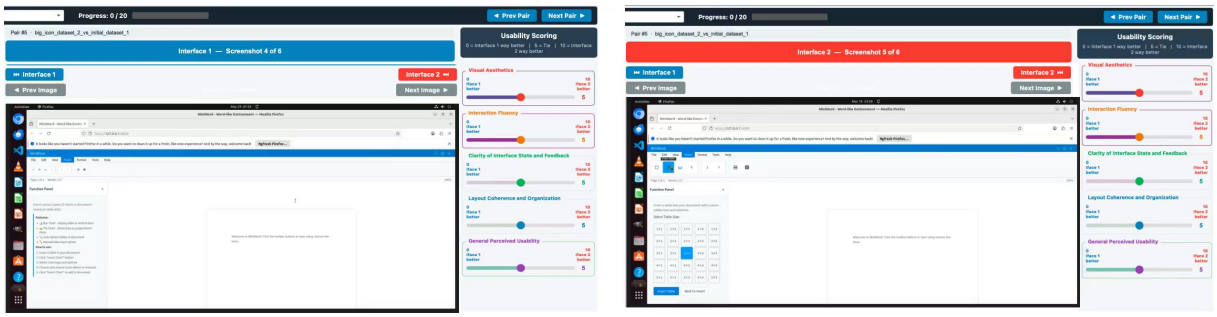

Figure 8: Human annotation interface used in the first part of our human study.

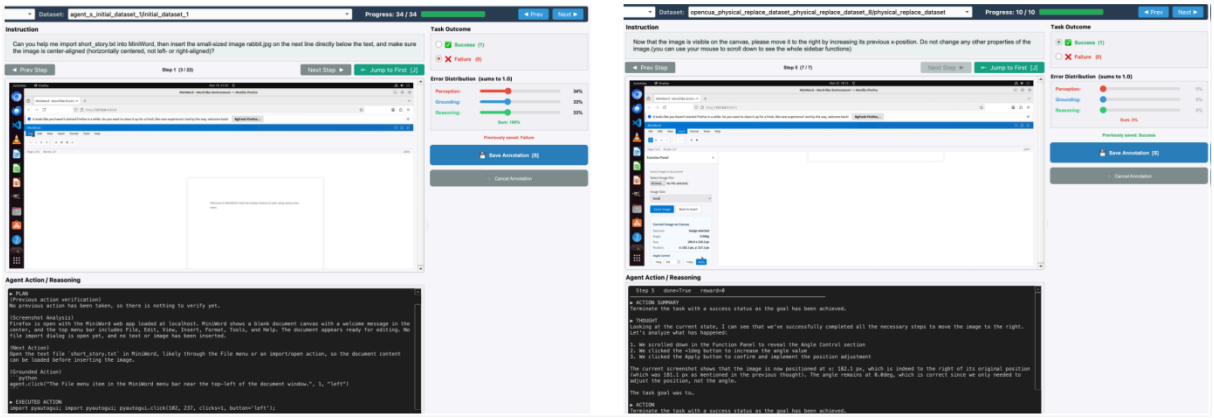

Figure 9: Human annotation interface used in the second part of our human study.