WindowsWorld: A Process-Centric Benchmark of Autonomous GUI Agents in Professional Cross-Application Environments

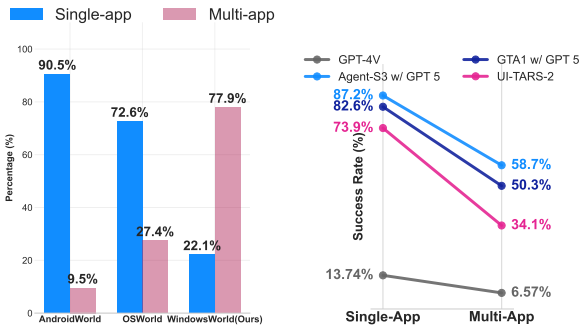
Anonymous ACL submission

Abstract

While GUI agents have shown impressive capabilities in common computer-use tasks such as OSWorld, current benchmarks mainly focus on isolated and single-application tasks. This overlooks a critical real-world requirement of coordinating across multiple applications to accomplish complex profession-specific workflows. To bridge this gap, we present a computer-use benchmark in cross-application workflows, named **WindowsWorld**, designed to systematically assess GUI Agents on complex multi-step tasks that mirror real-world professional activities. Our methodology uses a multi-agent framework steered by 16 occupations to generate four difficulty-level tasks with intermediate inspection, which are then refined by human review and executed in a simulated environment. The resulting benchmark contains 181 tasks with an average of 5.0 sub-goals across 17 common desktop applications, of which 78% are inherently multi-application. Experimental results of leading large models and agents show that: 1) All computer-use agents perform poorly on multi-application tasks ($<$ **21% success rate**), far below the performance of simple single-app tasks; 2) They largely fail at tasks requiring conditional judgment and reasoning across $\geq$3 applications, stalling at early sub-goals; 3) Low execution efficiency, where tasks often fail despite far exceeding human step limits.

1 Introduction

Autonomous Computer-Use Agents (CUAs) (Assouel et al., 2023; Chen et al., 2024; Qin et al., 2025; Sager et al., 2025), which interpret natural language instructions to interact with digital environments via directly controlling user interfaces, represent a pivotal step toward Artificial General Intelligence (AGI). Early research (Li et al., 2025; Rawles et al., 2023; Deng et al., 2023; Kapoor et al., 2024) was primarily confined to static offline environments, focusing on single-step action prediction or limited task completion within isolated application screens. A significant shift has occurred with the advent of execution-based evaluation in online operating systems, enabled by scalable Virtual-Machine (VM) environments such as OSWorld (Xie et al., 2024) and Windows Agent Arena (Bonatti et al., 2024). However, these benchmarks are still failing to capture the cognitive demands of real-world professional work that requires using multiple applications.



(a) (b)

Figure 1: **Limitations of current GUI benchmarks in real-world tasks.** (a) Distribution of single/cross-application tasks across benchmarks. (b) The task success rate drops with the increase in applications.

Specifically, existing benchmarks suffer from three critical limitations. *Firstly, cross-application tasks remain insufficient*, where most benchmarks focus on single-app navigation or short-horizon workflows, e.g., OSWorld (Xie et al., 2024) and AndroidWorld (Rawles et al., 2024), under-representing the cross-application coordination required in professional settings. As shown in Figure 1(a), widely used benchmarks exhibit a strong imbalance in task composition, with single-application tasks accounting for the majority of instances, while multi-application tasks remain sparsely represented ($<$28%). And GUI agents demonstrate high success rates on single-application tasks, yet their performance deterio-

rates markedly as task complexity increases. Figure 1(b) shows a steep drop in success rate when tasks require coordination across multiple applications, highlighting a significant gap between benchmark difficulty and real-world demands. *Secondly, the "all-or-nothing" scoring paradigm*, where judging success solely by the final success without progress evaluation leads to a lack of diagnostic granularity (Kong et al., 2025; Yang et al., 2025). In high-difficulty tasks where most state-of-the-art models fail to reach the final goal, current metrics (SR) cannot distinguish between models that made significant partial progress and those that failed at the first step of tasks. *Thirdly, benchmark construction remains bottlenecked by labour-intensive manual curation or repetitive template substitution*. Traditional pipelines (Xie et al., 2024) typically depend on forum crawling followed by extensive human editing, which is unscalable and frequently lacks the human-centric context of authentic professional routines (Mu et al., 2025). Furthermore, these benchmarks fail to automatically synthesize the complex file dependencies required to initialize evaluation tasks, creating a gap between simulated environments and real-world productivity.

To bridge these critical gaps, we introduce **WindowsWorld**, a comprehensive, challenging benchmark built on high-fidelity Windows virtual machines. It is constructed by a human-centric multi-agent data automation generation followed by human review. Specifically, it consists of:

- **Professional-Grade Multi-APP Tasks**: WindowsWorld comprises 181 tasks spanning 17 desktop applications, systematically generated based on 16 distinct personas across 5 categories. These tasks are organized into four difficulty levels that progressively increase planning horizon and cross-application coordination, including infeasible tasks to evaluate goal recognition and abstention. The benchmark is dominated by about 78% realistic multi-application workflows, capturing structured information transfer and common office productivity scenarios.

- **Fine-Grained Intermediate Process Checking**: Departing from only adopting final success evaluation metrics, we implement intermediate checking points for each task instruction. By validating essential sub-goals for each task, we assign partial-progress scores, providing higher discriminative power for evaluating computer-use agents' performance on long-horizon and high-difficulty workflows.

- **Automatic Task Construction**: We propose a human-in-the-loop multi-agent framework designed to generate real-world tasks. Initially, a Personasbased Generator Agent creates scenarios and task instructions grounded in specific professional roles and daily workflows. This is followed by a Refiner Agent phase to eliminate redundant tasks, and an Environment Generator Agent phase that synthesizes the necessary files for each mission. Notably, before the final environment generation step, human evaluators select the required task instructions.

Extensive evaluations of leading GUI models and agents, e.g., Gemini-3-Pro (Team et al., 2025), GPT-5.2 (Team, 2025), Agent S3 (Gonzalez-Pumariega et al., 2025), and others, reveal a significant performance gap in handling non-linear workflows and process-dependent constraints compared to simple tasks. The main contributions of our work are: 1) We present a comprehensive computer-use benchmark to assess GUI Agents' capability of completing multi-application collaborative tasks in a professional setting, with a progress evaluation metric besides the final success rate. 2) We introduce a human-centric multi-agent automatic generation pipeline, which significantly reduces the cost of dataset construction. 3) Experimental results reveal that current GUI agents perform well within a single application, yet their success rate and efficiency drop significantly when meeting tasks across applications. The best model Gemini-3-flash-previewachieves about only a 20% success rate.

2 Related Work

Desktop OS Benchmarks. Desktop environments present unique challenges due to high-resolution displays and heterogeneous widget styles. Early benchmarks, e.g., MiniWoB (Shi et al., 2017), MiniWoB++ (Liu et al., 2018), WebShop (Yao et al., 2022), Mind2Web (Deng et al., 2023), WebArena (Zhou et al., 2023), and VisualWebArena (Koh et al., 2024), focus on the single-step accuracy of computer-use agents or simple tasks. OSWorld (Xie et al., 2024) introduced the first scalable environment for Ubuntu, Windows, and macOS, supporting execution-based

Category	Applications	Count
Office Software	Word	40
	Excel	73
	PowerPoint	18
	Acrobat	11
Communication	Thunderbird	72
Web Browser	Chrome	70
System Manager	File Explorer	59
	Calculator	7
	Task Manager	2
	Snipping Tool	1
Multimedia	GIMP	14
	Paint	11
	Photos	7
	VLC	1
Programming	VS Code	30
	PowerShell	18
	Windows Terminal	9

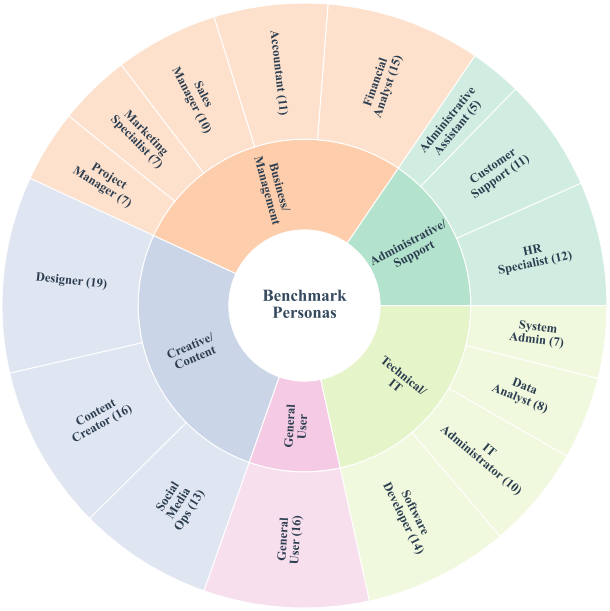


Figure 2: **(Left)** Distribution of applications across different categories and their respective task counts. **(Right)** 16 Personas in WindowsWorld are categorized in **5** major domains, each with fine-grained personal roles.

evaluation. Windows Agent Arena (Bonatti et al., 2024) expanded this with a focus on Windows OS and cloud-based parallelization to accelerate large-scale testing. OSUniverse (Davydova et al., 2025) introduced tasks with increasing complexity levels (from Paper to Gold) to benchmark the dexterity and precision of advanced agents. Our work builds upon the OSWorld infra but pushes the boundary of task complexity and diagnostic evaluation.

Evaluation of GUI Agent. Most early benchmarks relied on final state matching or success rate (SR). However, researchers have recognized that SR often collapses into binary outcomes that fail to indicate specific failure modes. ProBench (Yang et al., 2025) addressed this by introducing a "Process Provider" to capture accurate intermediate information for "Process-related Tasks". Similarly, SPA-Bench (Chen et al., 2025) and A3 (Chai et al., 2025) explored step-level state validation but often relied on rigid predefined trajectories or potentially unreliable LLM-based decomposition. WindowsWorld improves upon these by integrating a more flexible checkpoint-based scoring system that allows for alternative valid paths while ensuring functional correctness at key process nodes.

3 WindowsWorld Benchmark

WindowsWorld is a progress-centric benchmark for evaluating computer-use agents in realistic Windows desktop environments. All tasks are executed in controlled Windows virtual machines, where agents interact with a fixed set of commonly used productivity applications (see Figure 2) exclusively through standard GUI operations, without access to internal APIs or privileged system calls. This aims to ensure the reproduction of agents and easy deployment. In the following sections, we first give the task formulation of the computer-use agents (§3.1). We then describe the task and persona design that defines the benchmark's content space (§3.2). Next, we present a semi-automated task generation pipeline that combines LLM-based generation with systematic validation and human review (§3.3). We subsequently analyze the resulting benchmark in terms of task difficulty, cross-application complexity, and evaluation granularity (§3.4). Finally, we introduce the process-aware evaluation metrics (§3.5).

3.1 Task Formulation

Following the standard protocol for autonomous agents, we formalize GUI interaction as a Partially Observable Markov Decision Process (POMDP) $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ with state space $\mathcal{S}$, observation $\mathcal{O}$ (§A.1), action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, and reward function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.

At each step t, the agent executes a parameterized GUI action $a_t \in \mathcal{A}$ (e.g., mouse, keyboard, element-level operations) based on its observation, result in a new state $s_{t+1} \in \mathcal{S}$ and a new partial observation $o_{t+1} \in \mathcal{O}$. The agent repeats this process until output a termination mark (`DONE` or `FAIL`) or

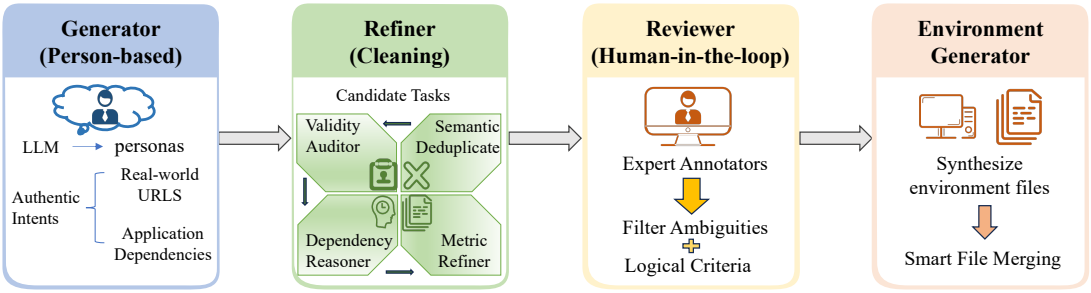


Figure 3: The overall architecture of the human-in-the-loop multi-agent pipeline.

exeeced maximum number of steps($t > t_{max}$), induces a trajectory

$$\tau = (o_1, a_1, \ldots, o_T, a_T), \tag{1}$$

The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ returns a non-zero value according to the query q and the trajectory τ if the agent achieves the task objective, or accurately predicts failure for infeasible task.

3.2 Task Taxonomy and Persona Design

All tasks are organized along two complementary axes: task complexity and professional persona.

Task Complexity Levels. Tasks are categorized into four levels based on their cognitive and operational complexity: L1 (*Single-App Atomic*) tasks involve complex operations within a single application. L2 (*Multi-App Linear*) tasks consist of sequential workflows that span multiple applications. L3 (*Dynamic Reasoning*) tasks involve conditional logic or inprocess reasoning across multiple applications, e.g., tailoring actions based on whether certain information is present or absent. L4 (*Infeasible*) tasks are intentionally unachievable due to factors such as invalid URLs, missing files, or required account authentication.

Persona Taxonomy. To ensure that the generated instructions reflect authentic professional routines and heterogeneous workflows, we define a comprehensive taxonomy of 16 distinct personas. As illustrated in Figure 2, this taxonomy categorizes major professional domains and provides detailed persona configuration, capturing specific intents and constraints characteristic of real-world roles. By moving beyond generic commands, these personas ground our benchmark in diverse, domain-specific workflows.

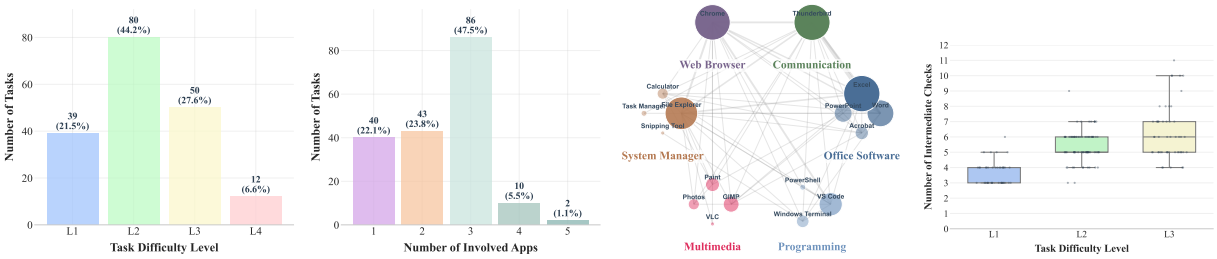
3.3 Human-In-The-Loop Multi-Agent Framework

To reduce construction costs and speed up construction, We propose a human-in-the-loop multi-agent pipeline to ensure ecological validity and scalability. Figure 3 provides an overview of the whole framework, where we describe each stage (agent) in detail below.

Generator. We leverage LLMs to generate tasks grounded in distinct professional personas (e.g., Accountant, Software Engineer). The generator receives structured prompts containing daily work routines and application-specific dependencies for each persona. In our implementation, we use DeepSeek-V3.2(Liu et al., 2025) as the generator and enable web search to retrieve and verify candidate URLs under an allowlist of open platforms. Unlike prior benchmarks relying on synthetic URLs, our generator is constrained to reference only accessible resources from open platforms (e.g., Github, Wikipedia, Stack overflow), ensuring that generated tasks reflect authentic web-integrated desktop workflows.

Refiner. The candidate tasks undergo automated quality assurance through a 4-node pipeline: (i) *Semantic Deduplicator*: Computes pairwise cosine similarity between task instruction embeddings and prunes near-duplicates exceeding a threshold (τ=0.85), ensuring diversity across the dataset, (ii) *Validity Auditor*: Performs asynchronous HTTP requests to verify URL accessibility and cross-references file mentions in instructions against the defined environment setup, flagging inconsistencies, (iii) *Dependency Reasoner*: Employs LLM-based reasoning to transform procedural preconditions (e.g., "open the spreadsheet") into declarative environment states (e.g., "spreadsheet exists at specified path"), enabling deterministic setup verification, and (iv) *Metric Refiner*: Reviews and standardizes evaluation criteria to ensure each intermediate checkpoint is unambiguous and verifiable against the expected task outcomes.

Human Reviewer. Humans perform the final



(a) (b) (c) (d)

Figure 4: **Benchmark analysis of WindowsWorld**. (a) Distribution of tasks across difficulty levels (L1-L4), highlighting the prevalence of non-trivial multi-step workflows. (b) Distribution of the number of applications per task, highlighting the prevalence of multi-app workflows (c) Cross-application interaction network, with edge weights representing co-occurrence frequency to reveal dependencies. (d) Distribution of task checkpoints by difficulty (L1-L3), showing increased checkpoint density for complex tasks.

Benchmark	#Apps	#Tasks	Multi-app Tasks	Intermediate Checks	Platform
AndroidWorld (Rawles et al., 2024)	20	116	✓(9.50%)	✗	Android
SPABench (Chen et al., 2025)	66	340	✓(11.76%)	✗	Android
ProBench (Yang et al., 2025)	34	200	–	✗	Android
OSWorld (Windows) (Xie et al., 2024)	13	49	✓(27.4%)	✗	Desktop
Windows Agent Arena (Bonatti et al., 2024)	11	154	✗	✗	Desktop
OSUniverse (Davydova et al., 2025)	9	160	✓(26.8%)	✗	Desktop
WorldGUI (Zhao et al., 2025)	10	611	–	✗	Desktop
WindowsWorld (ours)	17	181	✓(77.9%)	✓ (4.97/task)	Desktop

Table 1: **Comparison of execution-based benchmarks**. "Multi-app" indicates tasks with two or more applications; "Intermediate Checks" indicates tasks with intermediate-state checkpoints rather than result-only end-state evaluation. WindowsWorld contains the most apps and focuses on multi-app tasks across desktop benchmarks.

quality control to filter out tasks that automated validation cannot reliably detect. We employ four masters to check generated tasks, and they reject tasks of: (i) tasks with ambiguous or underspecified instructions that permit multiple valid interpretations, (ii) evaluation criteria that rely on subjective judgments rather than objective states, and (iii) tasks requiring unavailable proprietary software or inaccessible external services.

Environment Generator. For each validated task, the agent automatically synthesizes the required environment files(`.xlsx`, `.docx`, `.py`, etc.) using LLM-based content synthesis. We implement a Smart File Merging strategy that analyzes demands across multiple tasks within the same persona to create consistent data resources.

3.4 Task Analysis

Using a human-in-the-loop framework, we construct WindowsWorld, a benchmark of 181 tasks across 17 Windows applications, organized into four difficulty levels (L1-L4). Beyond feasible tasks (L1-L3), we introduce infeasible L4 tasks to test an agents ability to reject unachievable goals. As shown in Figure 4a and 4b, this benchmark emphasizes real-world multi-step workflows: 71.8% of tasks are non-trivial (L2/L3), and 77.9% involve two or more applications (avg. 2.4 apps/task). These multi-application workflows demonstrate structured dependencies (Figure 4c), such as recurring Excel-Thunderbird and Word-Thunderbird pairings that model common data-to-communication pipelines. To enable process-aware evaluation, tasks include explicit intermediate-state checkpoints (avg. 4.97 per task in Figure 4d) with higher-level tasks exhibiting denser checkpoints, thus enabling fine-grained diagnosis of long-horizon reasoning failures.

Comparison with Existing Benchmarks. As shown in Table 1, WindowsWorld distinguishes itself through its exclusive focus on crossapplication coordination and professional workflows. While existing benchmarks like AndroidWorld and OSWorld remain largely confined to singleapp navigation (with multiapp tasks comprising only 9.50% and 27.4% of their tasks, respectively), WindowsWorld is built around realistic multiapp workflows, constituting 77.9% of its evaluation

suite. Moreover, unlike benchmarks that rely on *binary "all-or-nothing" scoring*, our benchmark introduces process-aware evaluation (avg. 4.97/task). This enables fine-grained assessment of partial progress, providing a diagnostic framework that reveals specific reasoning and execution bottlenecks in multi-step tasks.

3.5 Process-Aware Evaluation

To address the lack of granularity in traditional "all-or-nothing" scoring for long-horizon GUI tasks, we propose a process-aware evaluation protocol. Formally, given an execution trajectory τ, a set of intermediate checkpoints $\mathcal{C} = \{c_k\}$, and a terminal state s_T, we employ an automated judge $\mathcal{J}$ (implemented via Qwen3-VL-Plus) that outputs binary decisions $\mathcal{J}(\cdot) \in \{0, 1\}$. The metrics are defined as follows:

$$S_{\text{int}} = \frac{1}{|\mathcal{C}|} \sum_{k=1}^{|\mathcal{C}|} \mathcal{J}(\tau, c_k), \qquad S_{\text{final}} = \mathcal{J}(\tau, s_T). \tag{2}$$

Intermediate Check Score (S_{int}): Evaluated on L1-L3 tasks, this metric rewards partial progress by tracking essential sub-goals, preventing zero scores on complex tasks where a substantial portion of the workflow is completed. To validate the reliability of $\mathcal{J}$, we manually annotated 30 randomly sampled tasks. The Pearson correlation between human judgments and the LLM-based scores is 0.90, indicating strong agreement and justifying the use of our scalable automated evaluation. More details are shown in Table 7.

Final Check Score (S_{final}): Applied across L1-L4, this assesses the correctness of the terminal state, including the agent's ability to correctly reject infeasible instructions in Level-4.

4 Experiment

4.1 Baselines

We evaluate a diverse suite of SOTA agents to establish performance benchmarks on WindowsWorld, categorized into general-purpose multimodal models and specialized GUI agents. And we also set different input types for assessing the impacts of observation modalities. Details are shown in Appendix A.1.

General-Purpose Models. We select leading proprietary models to test foundational capabilities: Gemini-3-flash/pro-preview (*20251217/20251118*), GPT-5.2 (*20251211*), Claude-Sonnet-4.5 (*20250929*), and Qwen3-VL-Plus (Bai et al., 2025).

GUI-Specialized Agents. Complementing the general models, we evaluate domain-specific systems: **Agent-S3** (Gonzalez-Pumariega et al., 2025): A research-oriented agent utilizing hierarchical task decomposition for complex interfaces. **UiPath Screen Agent** (Ilie, 2025): An industrial-grade automation tool designed for enterprise screen-level interactions.

4.2 Main Results and Analysis

Table 2 presents the performance across all evaluated agents, revealing three critical insights into current GUI automation capabilities.

The Efficiency-Completion Gap. All models exhibit a steep performance decay as the task horizon and complexity increase. While the SOTA *Gemini-3-flash* (Hybrid) maintains a reasonable intermediate progress score ($S_{\text{int}} = 50.32\%$), its success rate in reaching the terminal state drops to $S_{\text{final}} = 20.44\%$. This 30% discrepancy suggests that agents frequently "wander" through workflowscompleting sub-goals but failing to synthesize them into a successful conclusion.

Complexity Bottlenecks. We observe a sharp decline in functional correctness as we transition from L1 to L3 tasks. For instance, *Gemini-3-flash*'s S_{final} plummets from 35.90% (L1) to 16.67% (L3). This underscores a fundamental weakness in cross-application coordination and state maintenance; agents excel at isolated operations but struggle with the conditional reasoning (L3) required for professional-grade workflows.

Failure in Negative Constraint Handling (L4). Current models lack the "self-awareness" to identify infeasible instructions. Best *GPT-5.2* (SoM) achieves only 25% success rate on L4 tasks, indicating a high propensity for hallucinated success. UIPath performs poorly on general tasks, yet achieves a high L4 score because it often deems tasks incomplete. The ability to reliably report task failure remains a significant open challenge.

4.3 In-Depth Analysis

Analysis of Observation Modalities. Cross-modality evaluation identifies structured metadata (*Hybrid*) as the most reliable scaffold, with *Gemini-3-pro* leveraging accessibility trees to resolve UI density (gaining $+7.9\%$ S_{int}). In contrast, *Raw Screenshot* reveals a grounding schism:

Model	S_{int}				S_{final}				
	L1	L2	L3	Avg.	L1	L2	L3	L4	Avg.
Screenshot									
Gemini-3-pro-preview	46.20%	27.28%	24.88%	30.94%	25.64%	5.00 %	2.00 %	8.33 %	8.29 %
Gemini-3-flash-preview	57.05%	45.73%	29.66%	43.58%	38.46%	15.00%	8.00 %	0.00 %	17.13%
GPT-5.2	12.26%	6.56 %	5.51 %	7.69 %	5.13 %	1.33 %	0.00 %	0.00 %	1.78 %
Claude-Sonnet 4.5	7.35 %	3.12 %	2.31 %	3.86 %	0.00 %	0.00 %	0.00 %	0.00 %	0.00 %
Qwen3-vl-plus	26.37%	16.85%	10.69%	17.22%	2.56 %	0.00 %	0.00 %	0.00 %	0.55 %
Screenshot + Accessibility Tree (Hybrid)									
Gemini-3-pro-preview	63.33%	33.29%	29.68%	38.80%	41.94%	9.68 %	6.52 %	0.00 %	14.77%
Gemini-3-flash-preview	67.52%	49.23%	38.63%	50.32%	35.90%	17.50%	14.00%	16.67%	20.44%
GPT-5.2	13.63%	3.93 %	5.37 %	6.62 %	2.56 %	0.00 %	0.00 %	8.33 %	1.12 %
Claude-Sonnet 4.5	16.20%	3.16 %	2.04 %	5.84 %	0.00 %	0.00 %	0.00 %	8.33 %	0.55 %
Qwen3-vl-plus	22.52%	20.69%	14.79%	19.37%	7.69 %	1.25 %	10.00%	0.00 %	4.97 %
Set of Mark									
Gemini-3-pro-preview	47.86%	33.13%	24.18%	33.88%	23.08%	7.50 %	6.00%	8.33 %	9.94 %
Gemini-3-flash-preview	58.85%	38.64%	25.38%	39.38%	30.77%	15.00%	2.00 %	0.00 %	13.81%
GPT-5.2	3.42 %	3.61 %	1.25 %	4.57 %	0.00 %	1.25 %	0.00 %	25.00%	0.55 %
Claude-Sonnet 4.5	12.05%	3.30 %	1.87 %	4.90 %	2.56 %	0.00 %	0.00 %	8.33 %	0.55 %
Qwen3-vl-plus	6.62 %	2.12 %	5.88 %	4.27 %	5.13 %	0.00 %	4.00 %	0.00 %	2.21 %
Agent									
UIPath w/ Gemini-3-flash-preview	21.67%	15.83%	7.34 %	14.96%	3.03 %	0.00 %	0.00 %	50.00%	4.64 %
S3 w/ Qwen3-vl-plus	49.57%	30.31%	25.79%	33.47%	17.95%	3.75 %	2.00 %	16.67%	7.18 %
S3 w/ Gemini-3-flash-preview	51.84%	39.28%	40.11%	42.42%	30.77%	16.25%	10.00%	8.33 %	17.13%

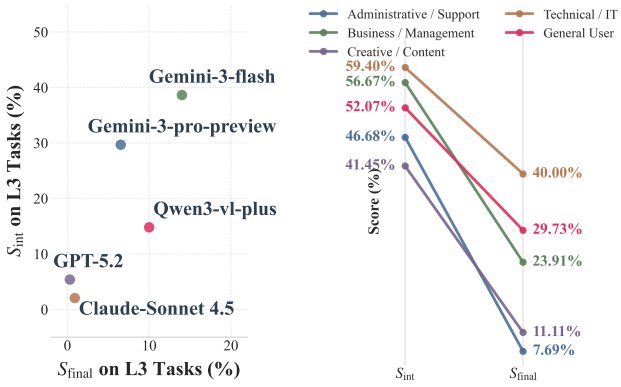
Table 2: **Model and Agents Performance on our WindowsAgent.** All large models use a unified PyAutoGUI action space, while UiPath employs the `Computer_13` action space from OSWorld. Pure models are evaluated under Screenshot, Screenshot + Accessibility Tree, and Set-of-Mark inputs; Agent-based systems (S3 and UiPath) use Screenshot input. Moreover, S3 and UiPath are integrated UI-TARS-1.5-7B as a grounding model. Each task is executed under a fixed maximum step budget that depends on task level: **15 (L1), 25 (L2), 40 (L3), and 20 (L4)**. S_{int} averages L1–L3 intermediate checkpoints and S_{final} averages L1–L4 final task completion.

Gemini-3-flash excels in zero-shot coordinate mapping ($S_{\text{int}} = 43.5\%$), while *Claude* and *GPT-5.2* fail at pixel-level parsing ($< 8\%$). Notably, *Set-of-Marks* (SoM) yields inconsistent returns. It suggests that while heuristic visual overlays can aid localization, they often introduce cognitive noise that disrupts the internal spatial reasoning of VLMs, whereas structured metadata provides a more stable foundation for GUI models.

Model Capabilities and Framework Synergy. The evaluation reveals a performance hierarchy led by the *Gemini-3-flash* series, which achieves a peak S_{int} of 50.32% under the *Hybrid* modality. In contrast, *GPT-5.2* exhibits a catastrophic failure in reasoning over structured interface data ($S_{final} \approx 0$), while *Claude-Sonnet 4.5* and *Qwen3-VL-Plus* show persistent gaps in GUI-specific visual reasoning. The integration of the *S3* framework significantly boosts performance for grounding-weak models like *Qwen3-VL-Plus* (nearly doubling S_{int}), yet offers diminishing returns for *Gemini* models with strong native grounding. This suggests that external modules become less effective as the capability of the foundation model grows.

Discriminative Power of Intermediate Checks. As task complexity scales, end-state evaluation becomes increasingly uninformative due to low-score saturation. Figure 5a demonstrates this phenomenon on L3 tasks: while most models exhibit stagnant completion rates ($S_{\text{final}} < 15\%$), their intermediate progress scores (S_{int}) remain highly discriminative, ranging from 2.04% to 38.63%. Even among agents with zero terminal success, S_{int} successfully distinguishes systematic progress from total failure. These results underscore that intermediate checks are essential for capturing latent competence in long-horizon tasks.

Persona-Dependent Sensitivity. Evaluation across personas (Figure 5b) shows stable intermediate progress but sharply diverging final success rates, highlighting a "long-tail" difficulty effect. Productivity-oriented personas exhibit a significant gap between S_{int} and S_{final}, indicating that failures stem from late-stage coordination and global consistency rather than initial execution. Conversely, Technical/IT personas show tighter



(a) (b)

Figure 5: **Detailed analysis of intermediate checks and persona-dependent difficulty.** (a) Scatter plot of intermediate checkpoint score (S_{int}) versus final task completion score (S_{final}) on L3 tasks under the Screenshot + Accessibility setting. (b) Model performance across different professional tasks.

Model	**Shot**	**Shot+A11y**	**SoM**
Avg. latency (s/step)			
Gemini-3-flash	10.60	9.60	16.95
Gemini-3-pro-preview	19.32	23.83	43.69
GPT-5.2	20.79	9.71	13.55
Claude-Sonnet 4.5	11.55	24.97	25.84
Qwen3-vl-plus	5.69	6.95	6.09
Avg. steps (Gemini-3-flash): Failure−Success (Δ)			
L1	6.13	4.64	5.75
L2	4.83	5.86	7.08
L3	7.00	14.14	12.00
L4	14.00	8.50	2.00

(a) Efficiency under different input modalities.

Lang.	S_{int}	S_{final}	S_{final} (L2)	S_{final} (L3)
ZH	43.59	17.13	15.00	8.00
EN	46.77	20.00	15.87	14.29

(b) Instruction language.

Table 3: **Ablation studies on efficiency and instruction language.** Latency is measured as the average per-step inference time. Step gap Δ denotes the difference between failed and successful trajectories (Failure−Success).

Model	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Gemini-3-pro-preview	39.2	22.4	20.0	8.0	4.0	1.6	0.8	0.8	0.0	0.0
Gemini-3-flash	23.1	26.2	21.5	12.3	5.4	2.3	0.0	0.0	0.0	0.0
S3 w/ Gemini-3-flash	29.2	27.7	10.8	19.2	6.9	0.8	0.8	0.8	0.0	0.8
S3 w/ Qwen3-vl-plus	33.8	32.3	16.2	5.4	8.5	2.3	0.8	0.0	0.0	0.0

Table 4: **Checkpoint-wise First-Failure Distribution on WindowsWorld (Screenshot-only).** The percentage of trajectories failing first at each checkpoint c_k, evaluated on feasible multi-app tasks (L2–L3).

coupling between progress and completion. These disparities underscore that persona-specific difficulty in *WindowsWorld* is driven by terminal-state complexity, validating intermediate checkpoints as critical diagnostic tools for identifying obscured execution bottlenecks (Table 8).

Inference Efficiency and Failure Dynamics. Table 3a highlights significant variance in per-step latency and execution patterns. While *Qwen3-VL-Plus* maintains superior speed (∼6s/step), *Gemini-3-pro* exhibits high modality sensitivity, particularly under SoM. Notably, SoM-style inputs introduce non-trivial overhead even for efficient models like *Gemini-3-flash*. Beyond latency, we observe that failed trajectories become increasingly inefficient as task difficulty scales. For *Gemini-3-flash*, the failure-success step gap (Δ) widens drastically from L1 to L3 (e.g., 4.64 → 14.14 in Hybrid), indicating that complex failures are rarely immediate but occur after extended, locally plausible execution. This "inefficient drift" confirms that current agents struggle with global plan consistency and lack the self-correction to terminate efficiently when progress stalls.

Impact of Instruction Language. Cross-lingual evaluation (Table 3b) reveals a persistent performance gap; English instructions consistently yield superior comprehension and planning stability over semantically equivalent Chinese prompts, particularly for *Gemini-3-Flash*. This advantage is most pronounced in multi-step execution, where English better sustains trajectory consistency. These results highlight a linguistic bias in current SOTA agentic models.

Error Analysis of Steps. We presented the percentage of errors the model made under different sub-goals in Table 4, finding that the model typically encountered errors at the beginning when dealing with cross-application tasks (L2–L3). Detailed case (in Appendix) studies revealed that the model performed poorly in Chinese file opening, cross-application information transfer.

5 Conclusion

In this paper, we present the first process-aware computer-use benchmark in a cross-application environment. The challenging benchmarks include professional-grade multi-app tasks (78%), fine-grained intermediate process checking, and an automatic task construction method. Experimental results show that the best agents only achieve a 20% success rate.

Limitations

Despite its benchmarking capabilities, our work has several limitations. **First**, our intermediate scores rely on full-trajectory execution and manually reviewed checkpoints, which restricts scalability for large-scale or online reinforcement learning (RL) in long-horizon tasks. Developing automated, generalizable reward formulations remains an open challenge. **Second**, while we evaluate cross-lingual instructions, the current environment is primarily optimized for a single-language OS interface; expanding to **multi-lingual OS environments** (e.g., non-English/Chinese UI elements) is necessary to assess true global robustness. **Finally**, *WindowsWorld* does not yet incorporate evaluation for Model Context Protocol (MCP) tools, which we leave for future expansion.

Ethical Considerations and Reproducibility

The data annotation and verification process involved four postgraduate researchers who filtered instructions and validated intermediate checkpoints. Annotators were compensated at a rate of 1.5 USD per task, reflecting fair market value for technical evaluation. As the instructions were generated through rule-based systems, they contain no personally identifiable information (PII) or harmful content.

To foster community growth and ensure reproducibility, we commit to **open-sourcing our complete research framework**, including the *WindowsWorld* evaluation environment, the full set of task instructions, and our process-aware evaluation metrics. All code and data will be made available on GitHub upon publication to support further research in GUI agents.

References

Rim Assouel, Tom Marty, Massimo Caccia, Issam H. Laradji, Alexandre Drouin, Sai Rajeswar, Hector Palacios, Quentin Cappart, David Vazquez, Nicolas Chapados, Maxime Gasse, and Alexandre Lacoste. 2023. The unsolved challenges of LLMs as generalist web agents: A case study. In *Proc. of the 37th Int. Conf. on NeurIPS: Foundation Models for Decision Making Workshop*, New Orleans, LA, USA. Curran Associates, Inc.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, and 1 others. 2025. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*.

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

Yuxiang Chai, Shunye Tang, Han Xiao, Weifeng Lin, Liang Liu, Hanhao Li, Jiayu Zhang, Pengxiang Zhao, Guangyi Liu, Rongduo Han, Guozhi Wang, Shuai Ren, Siyuan Huang, and Hongsheng Li. 2025. A3: Android agent arena for mobile GUI agents.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Liuyi Chen, Yilin Bai, Zhigang He, Chenlong Wang, Huichi Zhou, Yiqiang Li, Tianshuo Zhou, Yue Yu, Chujie Gao, Qihui Zhang, Yi Gui, Zhen Li, Yao Wan, Pan Zhou, Jianfeng Gao, and Lichao Sun. 2024. GUI-WORLD: A dataset for GUI-oriented multimodal LLM-based agents. *arXiv preprint*.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, Kaiwen Zhou, Rui Shao, Liqiang Nie, Yasheng Wang, Jianye HAO, Jun Wang, and Kun Shao. 2025. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *The Thirteenth International Conference on Learning Representations*.

Mariya Davydova, Daniel Jeffries, Patrick Barker, Arturo Márquez Flores, and Sinéad Ryan. 2025. Osuniverse: Benchmark for multimodal gui-navigation ai agents. *arXiv preprint arXiv:2505.03570*.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114.

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.

Alexandru-Gabriel Ilie. 2025. Uipath screen agent.

Raghav Kapoor, Yash Parag Butala, Melisa Russak, Jing Yu Koh, Kiran Kamble, Waseem AlShikh, and Ruslan Salakhutdinov. 2024. Omniact: A dataset and benchmark for enabling multimodal generalist autonomous agents for desktop and web. In *European Conference on Computer Vision*, pages 161–178. Springer.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, and 1 others. 2025. Mobileworld: Benchmarking autonomous mobile agents in agent-user interactive, and mcp-augmented environments. *arXiv preprint arXiv:2512.19432*.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. 2025. Screenspot-pro: Gui grounding for professional high-resolution computer use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8778–8786.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. *arXiv preprint arXiv:1802.08802*.

Jian Mu, Chaoyun Zhang, Chiming Ni, Lu Wang, Bo Qiao, Kartik Mathur, Qianhui Wu, Yuhang Xie, Xiaojun Ma, Mengyu Zhou, and 1 others. 2025. Gui-360: A comprehensive dataset and benchmark for computer-using agents. *arXiv preprint arXiv:2511.04307*.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, and 1 others. 2025. Ui-tars: Pioneering automated gui interaction with native agents. *arXiv 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, and 1 others. 2024. Androidworld: A dynamic benchmarking environment for autonomous agents. *arXiv preprint arXiv:2405.14573*.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728.

Pascal J Sager, Benjamin Meyer, Peng Yan, Rebekka von Wartburg-Kottler, Layan Etaiwi, Aref Enayati, Gabriel Nobel, Ahmed Abdulkadir, Benjamin F Grewe, and Thilo Stadelmann. 2025. A comprehensive survey of agents for computer use: Foundations, challenges, and future directions. *arXiv preprint arXiv:2501.16150*.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

OpenAI Team. 2025. Gpt-5 system card. *https://cdn.openai.com/gpt-5-system-card.pdf*.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, and 1 others. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094.

Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*.

Leyang Yang, Ziwei Wang, Xiaoxuan Tang, Sheng Zhou, Dajun Chen, Wei Jiang, and Yong Li. 2025. Probench: Benchmarking gui agents with accurate process information. *arXiv preprint arXiv:2511.09157*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Henry Hengyuan Zhao, Kaiming Yang, Wendi Yu, Difei Gao, and Mike Zheng Shou. 2025. Worldgui: An interactive benchmark for desktop gui automation from any starting point. *arXiv preprint arXiv:2502.08047*.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Details of WindowsWorld Environment

A.1 Observation Modalities

Diverging from prior works that rely on text-only (A11y) settings, we strictly focus on vision-centric modalities. This approach addresses the limitations of structural metadata, which is often incomplete or insufficient for visually dense professional interfaces (Koh et al., 2024). We evaluate agents under three distinct settings: 1) **Raw Screenshot:** The agent operates solely on visual input, necessitating both semantic reasoning and precise coordinate grounding without auxiliary markers. 2) **Set-of-Marks (SoM):** Visual inputs are augmented

Group	Action
computer.mouse	MOVE_TO() CLICK() MOUSE_DOWN() MOUSE_UP() RIGHT_CLICK() DOUBLE_CLICK() DRAG_TO() SCROLL()
computer.keyboard	TYPE() PRESS() KEY_DOWN() KEY_UP() HOTKEY()
system	WAIT() FAIL() DONE()

Table 5: **Action space in WindowsWorld.** Actions are grouped into mouse control, keyboard input, and system-level control signals.

with numbered bounding boxes overlaying interactive elements, reducing the grounding burden (Yang et al., 2023). 3) **Screenshot + Accessibility Tree (Hybrid):** A combination of visual data and structured metadata. We employ this modality primarily to assess the marginal utility of structural cues for open-source models.

A.2 Action Space

We provide two types of action spaces: free-form pyautogui and computer_13 from OSWorld (Xie et al., 2024). We mainly use pyautogui action space in experiments. And detailed actions in computer_13 are displayed in Table 5.

B Detailed Task Cases

B.1 Feasible Task

For feasible tasks (L1–L3), we present example case in Figure 6–8, including the natural languange instruction, task category, apps involved the task and evaluation metrics for final criterion and intermediate checkpoints. In addition, we also provide visual cross-apps interactions sample for reference in Figure 10.

B.2 Infeasible Task

For infeasible tasks (L4), we present example case in Figure 9, including the same content in the feasible tasks case but except intermediate checkpoints.


```
{
  "instruction": "Open the file 'config.json' on
       the desktop in VS Code, use the built-in
       formatting tool to format the JSON content
       , and save the file.",
  "task_category": "L1",
  "involved_apps": [ "VS Code" ],
  "evaluation_metrics": {
    "success_criterion": "The config.json file
         has been correctly formatted and saved
         through the built-in formatting function
          of VS Code. The content is valid JSON
         with standard indentation.",
    "intermediate_checks": [
      "The config.json file has been successfully
           opened in VS Code",
      "The JSON content has been formatted using
           VS Code formatting commands (such as
           Shift+Alt+F)",
      "The file has been successfully saved and
           its content remains in a valid JSON
           structure"
    ],
  }
}
```


Figure 6: Example WindowsWorld L1 task in JSON, including the natural-language instruction, involved applications, and intermediate/final evaluation criteria.


```
{
  "instruction": "Open the downloaded image '
       logo_raw.png' in the drawing tool, crop it
        to a square aspect ratio, then attach the
        cropped image and send an email to
       bench_serve1@2925.com with the subject '
       Cropped Logo for Review'.",
  "task_category": "L2",
  "involved_apps": [ "Paint", "Thunderbird", "
       File Explorer" ],
  "evaluation_metrics": {
    "success_criterion": "Successfully sending
         the resized photo to email
         bench_serve1@2925.com and themed '
         Cropped Logo for Review'.",
    "intermediate_checks": [
      "logo_raw.png is open",
      "The photo is resized to square",
      "The resized photo is saved as logo_cropped
           .png",
      "Added logo_cropped.png to an email and
           themed 'Cropped Logo for Review'"
    ]
  }
}
```


Figure 7: Example WindowsWorld L2 task

```
{
  "instruction": "Research 'obs-studio', 'shotcut
      ', and 'gimp' on GitHub, visit their
      README pages, summarize their core
      features in a Word document, and include
      direct links to each repository. Save the
      document as 'Tool_Comparison.docx' on the
      desktop.",
  "task_category": "L3",
  "involved_apps": [ "Chrome", "Word", "File
      Explorer" ],
  "evaluation_metrics": {
    "success_criterion": "A Word document named '
        Tool_Comparison.docx' exists on the
        desktop, containing the names of three
        GitHub content creation tools, a summary
         of their core features, and the
        corresponding full repository links.",
    "intermediate_checks": [
      "Chrome has opened the README pages of the
          three GitHub repositories.",
      "The key features of each tool have been
          extracted from the README and recorded
           in the Word document.",
      "The full URLs of each GitHub repository
          have been copied and included in the
          Word document.",
      "The Word document has been successfully
          saved as 'Tool_Comparison.docx' to the
           desktop."
    ]
  }
}
```


Figure 8: Example WindowsWorld L3 task


```
{
  "instruction": "Access https://gitee.com/
      nonexistent_user_2024/missing_project in
      Chrome, download the file 'config_template
      .json', and save it to the Downloads
      folder.",,
  "task_category": "L4",
  "involved_apps": [ "Chrome" ],
  "evaluation_metrics": {
    "success_criterion": "The Chrome TAB shows
        Gitee's 404 or the repository does not
        exist page. The Agent has output an
        error message indicating that the task
        is not feasible",
    "intermediate_checks": [ ],
  }
}
```


Figure 9: Example WindowsWorld L4 task

Level	Task Instruction
L1-1.	Search for the official Chinese documentation of pandas in Chrome, look up `read_excel`, copy the first paragraph into a new Notepad file `pandas_read_excel_notes.txt`, and save it to the desktop.
L1-2.	In File Explorer, navigate to `C:\Logs`, create a folder `System_Backup_2024`, and set it to read-only.
L2-1.	Open `iris.csv` on the desktop, compute the average of `SepalLength`, write the result into a new VS Code file, and save it as `iris_sepal_avg.txt` on the desktop.
L2-2.	Open the Quick Start Guide (`GitHub URL`), copy the section, then compose an email in Thunderbird to `bench_serve1@2925.com` with subject `[Support] OpenHarmony` and include the copied content.
L3-1.	Open `Q2_Financial_Report_Template.docx`, use `Q2_Sales_Data.xlsx` to compute the quarterly growth rate, fill the results, save as `Q2_Financial_Report_Final.docx`, and email it to `bench_serve1@2925.com`.
L3-2.	Batch process PNGs in `Design_Assets_Q3` in GIMP (resize to 800×600, add a 2-pixel border, export JPEG at 85%), then create `Q3_Preview.pptx` in PowerPoint with title Q3 and insert one processed sample image.
L4-1.	Download the latest `hosts` template from Gitee, save as `hosts_template.txt` on the desktop, and open it in VS Code to verify the content.
L4-2.	Access `https://gitee.com/nonexistent_user_2024/missing_project` in Chrome, download `config_template.json`, and save it to Downloads.

Table 6: **Task instances across difficulty levels (L1–L4).** Each level includes two representative instructions, ranging from single-app atomic operations (L1) to multi-app long-horizon workflows (L3) and infeasible tasks (L4).

B.3 Error Case

The conflict between the `pyautogui` action space and Chinese keyboard settings is illustrated in Figure 11. Since `pyautogui.write` simulates discrete keystrokes rather than direct string injection, non-ASCII characters are frequently intercepted by the system's Input Method Editor (IME). This causes the input to be misinterpreted as Pinyin, triggering unintended system shortcuts that lead to file-access errors and corrupted input strings. Such failures underscore the need for agents to possess cross-lingual environment awareness and robust text-entry strategies beyond basic keyboard simulation.

We present Figure 12 as a reference for failures in information transfer across software boundaries. In this scenario, the agent intends to migrate data from Excel to Word; however, an obstacle in application switching causes the agent

ID	Scores		ID	Scores	
	Human	LLM		Human	LLM
S1	0.000	0.000	S16	0.667	0.667
S2	0.250	0.250	S17	0.667	0.833
S3	0.000	0.250	S18	0.200	0.200
S4	0.143	0.143	S19	0.750	0.750
S5	0.200	0.400	S20	0.000	0.000
S6	0.500	0.500	S21	0.250	0.250
S7	0.666	0.500	S22	0.429	0.571
S8	0.500	1.000	S23	0.167	0.167
S9	0.600	0.800	S24	0.000	0.143
S10	0.800	0.800	S25	0.600	0.600
S11	0.222	0.222	S26	1.000	1.000
S12	0.500	0.500	S27	0.500	0.500
S13	0.500	0.667	S28	0.000	0.000
S14	0.833	1.000	S29	0.143	0.143
S15	0.500	1.000	S30	0.500	0.750

Table 7: Human annotations vs. LLM-based judge scores for intermediate checkpoints on 30 randomly sampled tasks.

to paste the content back into the source Excel spreadsheet. This persistent failure in synchronization is primarily driven by *unstable window focus management*, where the model fails to confirm the active GUI context before executing clipboard actions. Such errors demonstrate that current agents lack the robustness required for multi-step, multi-application workflows, underscoring the need for explicit state-verification mechanisms during task transitions.

C Additional Experimental Results

C.1 Reliability of VLMs as a judge

To validate the reliability of our automated VLM-as-judge protocol, we randomly sample 30 tasks and manually annotate whether each intermediate checkpoint is satisfied from the execution trajectory. We then compute the corresponding intermediate-check scores from human annotations and from the VLM judge (Qwen3-VL-Plus). As shown in Table 7, the two sets of scores are highly consistent, achieving a Pearson correlation of $r = 0.90$, which supports the use of VLM-based judging in our evaluation.

C.2 Persona-Wise Results

Table 8 reports performance stratified by five persona categories, evaluated with both intermediate checkpoint score (S_{int} and final completion score (S_{final}) under different input modalities and agent frameworks. This breakdown reveals substantial heterogeneity in difficulty across professional workflows, which is obscured by aggregate scores.

Progress–completion gap across personas. Across personas, a recurring pattern is the presence of large progress–completion gaps: agents often achieve non-trivial intermediate progress while failing to reach correct terminal states. This indicates that many failures occur after partial task execution rather than at the initial interaction stage.

Productivity-oriented personas. This gap is particularly pronounced in productivity-oriented personas (e.g., Administrative/Support and Creative/Content), where tasks are document-centric or visually driven and require maintaining global consistency across multiple steps. Although agents frequently succeed in local operations, they struggle with late-stage coordination and constraint satisfaction, leading to low final completion rates.

Structured Technical/IT workflows. In contrast, Technical/IT personas exhibit a tighter coupling between intermediate progress and final success. Tasks dominated by code editing and command-line interactions feature explicit goals and deterministic state transitions, under which current models perform comparatively better.

Implications for evaluation. These persona-wise differences suggest that difficulty in WindowsWorld is not uniform and that failures frequently arise in late-stage coordination rather than early execution. Consequently, intermediate checkpoint scores provide essential diagnostic signal for understanding model behavior across heterogeneous professional workflows, complementing final-state evaluation.

D Detailed multi-agent framework

D.1 Prompt for Generator

In Figure 13, we present the system prompt used in Generator agent in our human-in-the-loop multi-agent framework. This system prompt is input into the DeepSeek v3.2 model, along with various persona settings.

D.2 Prompt for Refiner

In Figure 14–15, we present the system prompt used in Refiner agent in our human-in-the-loop multi-agent framework.

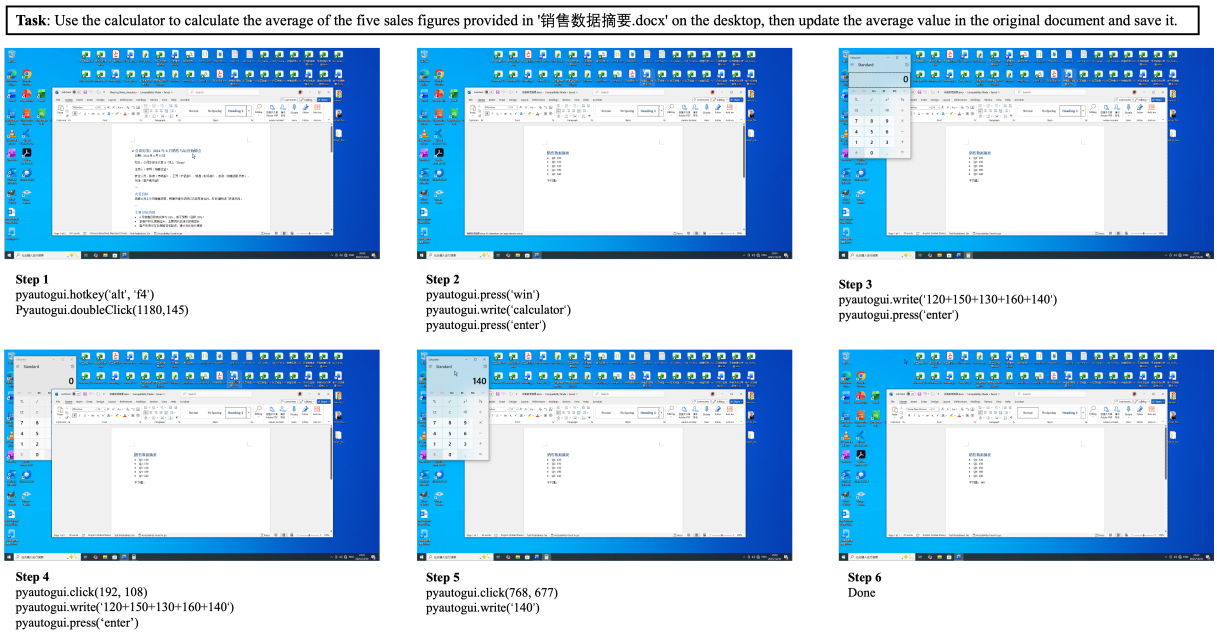
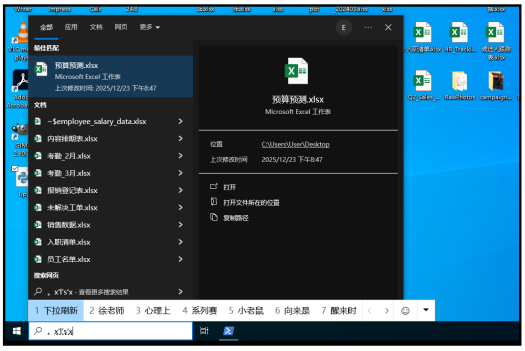


Figure 10: **Example execution trace on a feasible multi-app WindowsWorld task (L1).** We visualize a representative successful trajectory, including the agent's observed GUI states and actions. Each panel corresponds to a key step in the trajectory, annotated with the executed operation.

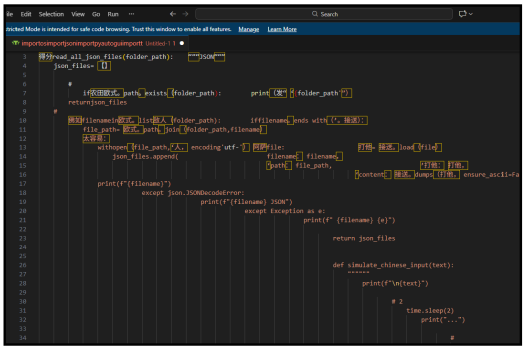
D.3 Prompt for Environment Generator

In Figure 16, we present the system prompt used in Environment Generator.

Task 1: Open the "预算预测.xlsx" and get the customers who are paid the most. Calculate their average salary.
Task 2: Write a Python file to read the JSON files on the Desktop/JSONs.



Task 1
Chinese characters are removed, and "xlsx" are also typed in Chinese IME.



Task 2
Some code words are typed in Chinese.

Figure 11: Technical challenges persist in the interaction between `pyautogui` and local Input Method Editors (IMEs), where Chinese input configurations often conflict with automated keystrokes. Furthermore, models frequently struggle to employ keyboard shortcuts for content synchronization across search interfaces. These issues highlight the need for agents to handle *stochastic GUI anomalies*such as IME pop-ups and inadvertent mouse triggers. Strengthening agent resilience to these environmental inconsistencies is vital for maintaining trajectory stability in complex desktop environments.

Task: Open the file '社交媒体内容日历.xlsx' on the desktop, find the scheduled post content for June 24, 2024, on the '微博' platform, copy the text, then create a new Word document and paste the text for review

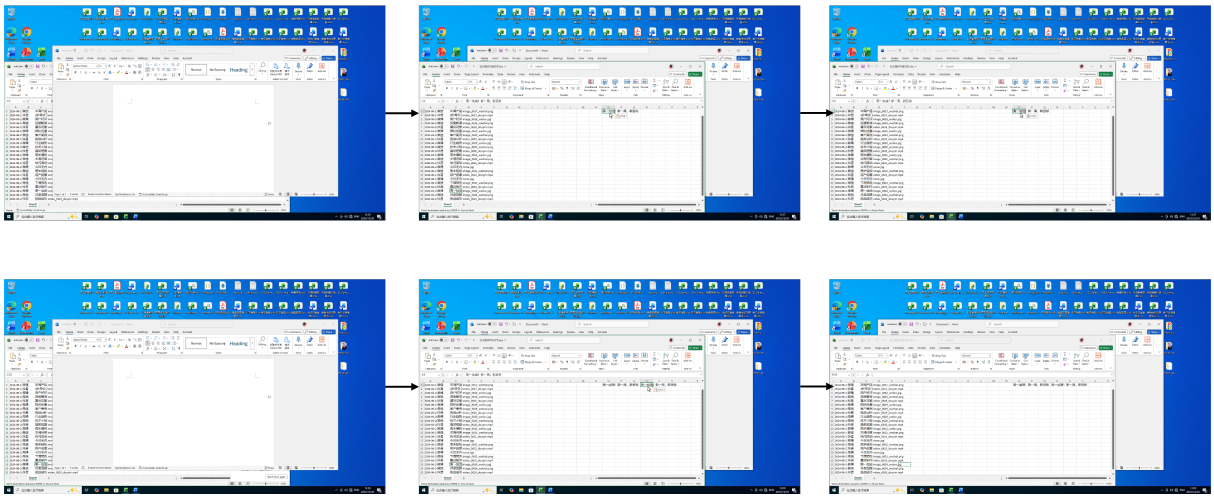


Figure 12: The model may suffer from the transfer between applications for cooperation (L2). However, an obstacle in application switching causes the agent to paste the content back into the source Excel spreadsheet. This persistent failure in synchronization is primarily driven by *unstable window focus management*, where the model fails to confirm the active GUI context before executing clipboard actions.

Model	Administractive / Support		Business / Management		Creative / Content		Technical / IT		General User	
	S_{int}	S_{final}	S_{int}	S_{final}	S_{int}	S_{final}	S_{int}	S_{final}	S_{int}	S_{final}
Screenshot										
Gemini-3-pro-preview	29.34%	0.00 %	30.88%	8.70 %	23.68%	4.40 %	37.27%	20.00%	38.38%	16.22%
Gemini-3-flash-preview	35.92%	15.38%	43.38%	10.87%	36.77%	8.89 %	54.95%	33.33%	52.93%	35.14%
GPT-5.2	5.00 %	0.00 %	6.19 %	0.00 %	4.35 %	0.00 %	7.78 %	6.67 %	15.49%	5.88 %
Claude-Sonnet 4.5	4.01 %	0.00 %	4.03 %	0.00 %	3.44 %	0.00 %	6.89 %	0.00 %	11.85%	0.00 %
Qwen3-vl-plus	21.56%	0.00 %	20.69%	0.00 %	12.73%	0.00 %	19.27%	0.00 %	14.50%	2.70 %
Screenshot + Accessibiliy Tree										
Gemini-3-pro-preview	26.26%	7.69 %	24.70%	2.17 %	27.58%	15.15%	31.24%	20.00%	42.66%	18.92%
Gemini-3-flash-preview	46.68%	7.69 %	56.67%	23.91%	41.45%	11.11%	59.40%	40.00%	52.07%	29.73%
GPT-5.2	5.69 %	0.00 %	6.43 %	2.17 %	3.61 %	0.00 %	5.40 %	0.00 %	11.40%	0.00 %
Claude-Sonnet 4.5	0.77 %	0.00 %	3.67 %	0.00 %	2.44 %	0.00 %	3.33 %	0.00 %	8.20 %	0.00 %
Qwen3-vl-plus	19.44%	3.85 %	20.38%	2.17 %	15.59%	2.22 %	12.22%	6.67 %	25.54%	13.51%
Set of Mark										
Gemini-3-pro-preview	34.26%	7.69 %	30.96%	4.35 %	23.89%	6.67 %	47.78%	33.33%	43.78%	16.22%
Gemini-3-flash-preview	39.33%	7.69 %	41.53%	17.39%	29.26%	4.55 %	36.95%	20.00%	50.81%	27.03%
GPT-5.2	0.00 %	0.00 %	3.79 %	0.00 %	4.07 %	0.00 %	1.90 %	0.00 %	10.45%	2.70 %
Claude-Sonnet 4.5	2.82 %	0.00 %	3.18 %	0.00 %	5.81 %	0.00 %	0.00 %	0.00 %	9.37 %	2.70 %
Qwen3-vl-plus	3.08 %	3.85 %	3.39 %	2.17 %	1.74 %	0.00 %	5.41 %	0.00 %	8.92 %	5.41 %
Agent										
UIPath w/ Qwen3-vl-plus	30.28%	0.00 %	33.86%	0.00 %	31.46%	4.44 %	36.06%	20.00%	36.62%	16.22%
UIPath w/ Gemini-3-flash-preview	33.32%	15.38%	39.09%	13.04%	43.54%	15.56%	54.60%	33.33%	46.67%	21.62%
S3 w/ Gemini-3-flash-preview	18.01%	0.00 %	19.48%	2.86 %	13.90%	0.00 %	6.06 %	0.00 %	12.56%	0.00 %

Table 8: Main results on WindowsWorld cross input modalities and agent frameworks. Parted by each personae.

You are an expert Data Engineer specializing in building benchmarks for GUI Agents. Your task is to generate realistic, complex, and structured user commands for a Windows-based Agent.

 # Current Generation Context - Target Persona: {persona} - **Task Category:** {category} ({category_desc}) - **Number of Tasks to Generate:** {batch_size}

 # Persona Details The {persona} typically uses these applications: - **Primary Apps:** {primary_apps} - **Secondary Apps:** {secondary_apps} - **Typical Work Tasks:** {typical_tasks}

 # Task Categorization Rules - L1 (Single-App Atomic): Operations within a SINGLE application only - **L2 (Multi-App Linear):** Involves 2-3 apps with clear sequential flow - **L3 (Dynamic Planning & Reasoning):** Involves 3+ apps OR requires conditional logic/math/decision making - **L4 (Impossible/Negative Tasks):** Tasks that CANNOT be completed (missing files, dead URLs, etc.)

 # Output Format You MUST output a JSON array containing exactly {batch_size} task objects. Each object must follow this schema:

{ "task_id": "win_{persona_id}_{category}_XXX", "instruction": "Natural language command (in English)", "instruction_cn": "Natural language command (in Chinese)", "task_category": "{category}", "persona": "{persona}", "involved_apps": ["MainApp", "SecondaryApp1", "SecondaryApp2"], "pre_conditions": ["employee_data.xlsx exists on Desktop", "Outlook is logged into company email"], "environment_setup": { "files_to_create": [...], "urls_to_mock": [...], "downloadable_resources": [...] }, "evaluation_metrics": { "success_state": "Description of the final state when task is fully successful", "intermediate_checks": [...] } }

 # CRITICAL: environment_setup Usage Rules

environment_setup is NOT a required field! Only populate it when the task genuinely requires pre-existing files.

When environment_setup IS needed: - Task instruction explicitly mentions "open a file", "process a document", "edit a spreadsheet" - Task requires operating on an **already existing** local file - Example: "Open employee_data.xlsx on Desktop, filter Engineering department employees"

When environment_setup is NOT needed (leave empty): - Task involves searching/browsing web content (e.g., "Search xxx in Chrome") - Task involves creating new files (e.g., "Create a new Word document") - Task involves sending emails (without attachments) - Task involves downloading content from websites then processing - Task uses built-in system features (Calculator, Paint for new drawings)

 # Intermediate Checks - CRITICAL Guidelines

- intermediate_checks MUST list ALL MANDATORY steps required to complete the task - Each check represents a REQUIRED milestone that CANNOT be skipped - List checks in chronological order matching the task flow - Use observable, verifiable states (e.g., "File saved", "Email sent") - Do NOT include optional steps or alternative paths - For LLM-based verification - describe states, NOT assertions or code

MOST IMPORTANT: NEVER FABRICATE SPECIFIC DETAILS! The check content MUST match the EXACT specificity level of the instruction.

Rule: If specific value is NOT in instruction, it should NOT appear in check!

 # CONSISTENCY RULE - Placeholders & Specific Values Must Match Across All Fields

- **instruction**, **instruction_cn**, **intermediate_checks**, and **success_criterion** MUST use IDENTICAL placeholders/values

If instruction says "filter by 'Engineering' department": - intermediate_check: "Filtered 'Engineering' department candidates" - success_criterion: "Document contains Engineering department candidate information"

The same placeholder in instruction should appear in checks exactly as specified.

 # Category-Specific Rules for L1 (Single-App Atomic) Tasks

- Focus on single application operations - Examples: Open a file, format a document, send an email, run a script - Only ONE app should be in involved_apps (this is automatically the primary app) - Pre-conditions should relate to that single app being ready - **Intermediate Checks:** List 2-4 MANDATORY steps (e.g., "File opened" -> "Content modified" -> "File saved")

CRITICAL - DO NOT fabricate specific values in checks: - WRONG: "Email subject is 'Interview Invitation - John Smith'" (fabricated name) - CORRECT: "Email subject is 'Interview Invitation - [Candidate Name]'" (matches instruction placeholder) - WRONG: "Email body contains interview time (July 10, 2024 at 10AM, Teams meeting)" (fabricated details) - CORRECT: "Email body contains standard interview information (time, location, required materials)" (generic description) - If instruction says "schedule details" without specific date, check should say "Schedule filled" NOT a made-up date

 # Category-Specific Rules for L2 (Multi-App Linear) Tasks

- Involve exactly 2-3 applications - Clear sequential flow (MainApp -> SecondaryApp -> ...) - **IMPORTANT:** List apps in order of PRIMARY usage. The first app should be where MOST work happens. - Examples: ["Excel", "Word"] for "Copy Excel data and format in Word" (Excel is primary) - Each step should naturally lead to the next - **Intermediate Checks:** List 3-6 MANDATORY steps covering ALL app transitions (e.g., "Excel data extracted" -> "Word document created" -> "Data pasted" -> "Document saved")

CRITICAL - Checks must match instruction specificity level: - If instruction says "copy the data", check should say "Data copied" NOT list specific data content - If instruction says "extract names", check should say "Names extracted" NOT "Extracted John, Mary, Tom" - If instruction gives specific values (e.g., "filter by 'Engineering' department"), check CAN include "Filtered 'Engineering'" - NEVER invent dates, names, numbers, or content not explicitly stated in instruction

 # Category-Specific Rules for L3 (Dynamic Planning & Reasoning) Tasks

- Involve 2+ applications OR complex reasoning - **IMPORTANT:** The FIRST app in involved_apps should be the PRIMARY app where most complex work happens - Include conditional logic: "if...then...", "based on...determine..." - Include calculations or data aggregation across sources - Examples: ["Excel", "Outlook", "Chrome"] for "Analyze Excel data, research online, send email summary" (Excel is primary) - **Intermediate Checks:** List 4-8 MANDATORY steps including decision points (e.g., "Data loaded" -> "Condition evaluated" -> "Calculation completed" -> "Results summarized" -> "Email sent")

CRITICAL - Keep checks abstract for dynamic content: - For conditional tasks: "Condition evaluated" NOT "Evaluation result is yes" (result depends on actual data) - For calculations: "Average calculated" NOT "Average is 85000" (value depends on data) - For extracted content: "Key information extracted" NOT specific extracted text - Checks should be verifiable regardless of actual data values

 # Category-Specific Rules for L4 (Impossible/Negative) Tasks

- Design tasks that WILL FAIL by design - Examples: - Open non-existent file "missing_report_2024.xlsx" - Navigate to broken URL "https://internal.company.fake/dashboard" - Send email to non-existent contact - The goal is to test if the agent can properly report errors and handle failure cases - Include clearly invalid resources in environment_setup

Figure 13: System prompt used for persona-conditioned task generation in WindowsWorld. Placeholders (e.g., {persona}, {primary_apps}) are instantiated during generation.

 # Dependency Reasoner - Converting Actions to States
You are a professional task analysis expert. Please convert the following "action-form" pre-conditions into "state-form".
Original Task Instruction: {instruction}
Current Pre-conditions (may be in action form): {pre_conditions}
Conversion Rules:

- "Open Excel" → "Excel application is open and active"
- "Log into email" → "User is logged into Outlook/email account"
- "Connect to network" → "Network connection is stable and available"
- "Open file X" → "File X exists at the specified path and is accessible"

Output Requirements:

- Output in JSON array format
- Each condition should describe a specific state
- Include necessary environment states (network, login status, etc.)
- Output ONLY the JSON array, no other text

Figure 14: Prompt for the Dependency Reasoner node in the refiner pipeline. This module converts action-based pre-conditions into verifiable state descriptions.

 # Metric Refiner - Generating Programmable Evaluation Assertions
You are a professional test engineer. Please generate programmable evaluation assertions for the following GUI Agent task.
Task ID: {task_id} **Task Instruction:** {instruction} **Involved Apps:** {involved_apps} **Task Category:** {task_category}
Current Evaluation Metrics:

- Success Criterion: {success_criterion}
- Intermediate Checks: {intermediate_checks}
- Existing Assertions: {assertions}

Output Requirements: Generate a JSON object containing:

1. success_criterion: Concise success determination description
2. intermediate_checks: List of intermediate state checkpoints (retain and optimize original content)
3. assertions: List of programmable assertions (using pseudo-code function format)
4. expected_final_state: Expected final system state

Assertion Function Examples:

- file_exists("path/to/file.xlsx") - Check if file exists
- file_contains("file.xlsx", "keyword") - Check file content
- window_active("Application Name") - Check if window is active
- email_sent_to("recipient@email.com") - Check email sent
- clipboard_contains("text") - Check clipboard content
- browser_url_matches("pattern") - Check browser URL
- application_running("app_name") - Check application running status
- error_dialog_shown() - Check error dialog (for L4 tasks)

For L4 tasks (impossible tasks), assertions should verify that the Agent correctly reported the error.
Output ONLY the JSON object in the following format: { "success_criterion": "...", "intermediate_checks": ["Step 1 completion state", "Step 2 completion state", ...], "assertions": ["...", "..."], "expected_final_state": "..." }

Figure 15: Prompt for the Metric Refiner node in the refiner pipeline. This module generates programmable assertions for automated evaluation.

 # Environment File Content Generator

You are a professional file content generator responsible for creating environment files that support GUI task execution in WindowsWorld.

[Global Requirements]

- Generated content MUST strictly satisfy the task instruction.
- If the instruction mentions specific entities (e.g., departments, names, thresholds), the content MUST include them.
- Output ONLY file content. Do NOT include explanations or metadata.

---

(1) Text-Based Files (txt, md, json, py)

Purpose: Generate executable or readable text content (e.g., config files, scripts, notes).

- Ensure correct syntax for code or structured formats (e.g., valid JSON).
- Content must directly support task execution.

Inputs: Filename, File Type, Content Requirements, Task Instruction

---

(2) Spreadsheet Files (Excel / CSV)

Purpose: Generate tabular data for filtering, aggregation, or analysis tasks.

- Output MUST be valid JSON with "columns" and "data" fields.
- Data MUST include both matching and non-matching records for specified conditions.
- Recommended size: 15–25 rows.

Inputs: Filename, File Type, Content Requirements, Task Instruction

---

(3) Word Documents

Purpose: Generate structured document content for reporting or writing tasks.

- Use Markdown-style formatting (headings, lists, emphasis).
- Ensure all required content specified in the instruction is included.

Inputs: Filename, File Type, Content Requirements, Task Instruction

Figure 16: Unified prompt for environment file generation in WindowsWorld. A single generator produces text files, spreadsheets, and documents with task-aligned content, ensuring that environment artifacts are executable, verifiable, and consistent with task instructions.