AgenticOS: A Modular Framework for Deep OS Integration and Intelligent Desktop Automation

Jiaqi Zou
jiaqizou@github

June 2025

Abstract

We present **AgenticOS**, a modular Python framework that transforms Windows desktops into AI-navigable environments through natural language interaction. Unlike existing approaches that rely solely on visual grounding (Operator, Navi) or require dual-agent architectures (UFO^2), AgenticOS introduces a *hybrid three-layer grounding* pipeline combining UI Automation accessibility trees, vision-language models, and OCR in a unified fallback hierarchy. Built on a ReAct (Reason+Act) loop, the system decomposes complex multi-application tasks into atomic actions executed through keyboard, mouse, shell, and window management primitives. AgenticOS exposes its capabilities via the Model Context Protocol (MCP), enabling seamless integration with external LLM clients. We evaluate on a custom 30-task benchmark suite spanning basic single-app operations to advanced multi-application workflows and compare against UFO^2, Operator, Navi, and Claude Computer Use. Our framework achieves competitive performance while offering superior modularity, extensibility, and transparency through automatic GIF session recording.

1 Introduction

The vision of an AI-powered operating system—where users express intent in natural language and an agent autonomously navigates the desktop—has gained significant traction with the advent of large multimodal models [Anthropic, 2024a]. Recent systems such as Microsoft's UFO [Zhang et al., 2024], UFO^2 [Zhang et al., 2025], Google's Mariner, and Anthropic's Claude Computer Use demonstrate the feasibility of this paradigm.

However, existing approaches face several challenges:

1. **Grounding brittleness**: Vision-only systems (Operator, Navi) struggle with small UI elements, non-standard controls, and high-DPI displays.
2. **Architectural rigidity**: Dual-agent systems (UFO^2) couple planning and execution tightly, making extension difficult.
3. **Limited extensibility**: Most systems lack standardized APIs for external tool integration.
4. **Opacity**: Agent decisions are difficult to audit without session recording.

AgenticOS addresses these challenges through three key contributions:

1. A **hybrid three-layer grounding** pipeline that combines Windows UI Automation (UIA) accessibility trees with vision-language model (VLM) analysis and OCR, using intelligent fallback when structured information is insufficient.
2. A **fully modular architecture** with cleanly separated observation, grounding, action, and agent layers, enabling independent testing and extension of each component.
3. Native **Model Context Protocol (MCP)** integration, exposing 11 desktop automation tools as an MCP server for use by any compatible LLM client.

2 Related Work

2.1 GUI Automation Agents

UFO / UFO² [Zhang et al., 2024, 2025]: Microsoft's UI-Focused Agent employs a dual-agent architecture (HostAgent for app selection, AppAgent for in-app navigation) using GPT-4V for visual grounding combined with UIA control information. UFO^2 extends this with a "UFO Space" middleware layer. On OSWorld, UFO^2 achieves 30.5% success rate.

Operator (OpenAI): A proprietary system using Computer-Using Agent (CUA) models with vision-only grounding. Achieves 20.8% on OSWorld but lacks open-source availability.

Navi [Navi, 2024]: A foundation model for GUI navigation trained on web and desktop data. Reports 19.5% on OSWorld with pure visual grounding.

Claude Computer Use [Anthropic, 2024a]: Anthropic's approach using Claude's native computer use capabilities with screenshot-based grounding. Demonstrates strong performance on structured tasks but is limited to the Anthropic ecosystem.

2.2 Screen Understanding

OmniParser [Lu et al., 2024]: A screen parsing toolkit that extracts interactable elements and their semantics from screenshots using specialized vision models. Achieves 93.9% accuracy on ScreenSpot.

SeeClick [Cheng et al., 2024]: A visual GUI agent with heuristic pre-training for element grounding. Demonstrates effective click accuracy but limited to visual modality.

2.3 Benchmarks

OSWorld [Xie et al., 2024]: A comprehensive benchmark with 369 real-world tasks across Ubuntu, Windows, and macOS. The current SOTA (UFO^2) achieves only 30.5%, with human performance at 74.5%.

WindowsAgentArena [Bonatti et al., 2024]: A Windows-specific benchmark with 154 tasks using Azure VMs. Navi achieves 19.5% success rate.

AgentBench [Liu et al., 2023]: A multi-environment benchmark evaluating LLM agents across operating systems, databases, web browsing, and more.

3 System Design

3.1 Architecture Overview

AgenticOS follows a layered architecture with strict separation of concerns:

1. **Observation Layer**: Screen capture (via `mss`) and threaded GIF recording with action annotations.
2. **Grounding Layer**: Three-layer pipeline — UIA accessibility tree (primary), VLM visual analysis (fallback), OCR text detection (supplement).
3. **Action Layer**: Composited action execution with retry logic across keyboard, mouse, shell, and window management primitives.
4. **Agent Layer**: ReAct-based navigator with LLM-powered reasoning and task planner for decomposition.
5. **Interface Layer**: Rich CLI for interactive chat and MCP server for programmatic access.

3.2 Hybrid Three-Layer Grounding

The core innovation of AgenticOS is its grounding pipeline, formalized as:

$$G(s) = \begin{cases} G_{\text{UIA}}(s) & \text{if } |G_{\text{UIA}}(s)| \geq \tau \\ G_{\text{UIA}}(s) \cup G_{\text{VLM}}(s) & \text{if } |G_{\text{UIA}}(s)| < \tau \\ G_{\text{UIA}}(s) \cup G_{\text{OCR}}(s) & \text{if VLM unavailable} \end{cases} \tag{1}$$

where s is the current screenshot, G_{UIA} extracts elements from the Windows UI Automation accessibility tree, G_{VLM} uses a vision-language model for visual element detection, G_{OCR} applies optical character recognition, and τ is the minimum element threshold (default $\tau = 3$).

This approach ensures robust grounding across:

- **Standard Windows applications**: UIA provides rich, structured element information.
- **Custom-rendered UIs**: VLM identifies visual elements without accessibility support.
- **Text-heavy interfaces**: OCR captures text content and locations.

3.3 ReAct Navigation Loop

The navigator agent follows the ReAct paradigm [Yao et al., 2023]:

Algorithm 1 AgenticOS ReAct Navigation

Require: Task description T, max steps N

history $\leftarrow$ []
for $i = 1$ to N **do**
  $o_i \leftarrow$ Observe() {Screenshot + UIA + fallback}
  $t_i \leftarrow$ Think(T, o_i, history) {LLM reasoning}
  if t_i indicates task complete **then**
    return SUCCESS
  end if
  $a_i \leftarrow$ ParseAction(t_i) {Extract action from LLM}
  $r_i \leftarrow$ Act(a_i) {Execute with retry}
  history.append((o_i, t_i, a_i, r_i))
end for
return MAXSTEPSEXCEEDED

3.4 Action Composition

Actions are represented as typed dataclasses with 16 action types:

```
class ActionType(Enum):
    CLICK = "click"
    DOUBLE_CLICK = "double_click"
    RIGHT_CLICK = "right_click"
    TYPE_TEXT = "type_text"
    PRESS_KEY = "press_key"
    HOTKEY = "hotkey"
    SCROLL = "scroll"
    DRAG = "drag"
    SHELL = "shell"
    OPEN_APP = "open_app"
    FOCUS_WINDOW = "focus_window"
    WAIT = "wait"
    DONE = "done"
    ...
```

The `ActionCompositor` dispatches actions to specialized executors with configurable retry logic (default: 2 retries with 0.5s delay).

3.5 MCP Server Integration

AgenticOS exposes its capabilities through the Model Context Protocol [Anthropic, 2024b], providing 11 tools:

Tool	Description
`take_screenshot`	Capture current screen
`click`	Click at coordinates
`type_text`	Type text string
`press_key`	Press keyboard key
`hotkey`	Press key combination
`scroll`	Scroll mouse wheel
`get_ui_tree`	Get UIA element tree
`run_shell`	Execute shell command
`open_app`	Launch application
`list_windows`	List open windows
`focus_window`	Focus window by title

Table 1: MCP tools exposed by AgenticOS server.

4 Implementation

AgenticOS is implemented in Python 3.10+ with the following key dependencies:

- `litellm`: Unified interface to 100+ LLM providers (Claude, GPT-4o, Gemini, Ollama).
- `pywinauto`: Windows UI Automation with UIA backend for accessibility tree extraction.
- `mss`: Cross-platform screen capture with minimal overhead.
- `pyautogui` + `keyboard`: Input simulation for mouse and keyboard.
- `mcp` (FastMCP): Model Context Protocol server implementation.
- `rich` + `click`: Terminal-based chat interface with streaming.
- `pydantic-settings`: Type-safe configuration with environment variable support.

The codebase is organized into six packages totaling approximately 2,500 lines of code across 20 modules, with 100% type annotation coverage.

4.1 Safety Mechanisms

AgenticOS implements multiple safety layers:

1. **Command blocklist**: Dangerous shell commands (e.g., `format`, `del /s`, `shutdown`) are blocked.
2. **Action confirmation**: Users can require confirmation before each action execution.
3. **Step limits**: Tasks are bounded by a configurable maximum step count.
4. **Exception hierarchy**: Typed exceptions (`ActionBlockedError`, `MaxStepsExceeded`) enable structured error handling.

5 Evaluation

5.1 Benchmark Suite

We introduce a 30-task benchmark suite organized into three difficulty tiers:

Tasks are defined declaratively with expected outcomes, enabling automated success verification:

Category	Tasks	Example Tasks
Basic	15	Open Notepad, Calculator arithmetic, File Explorer nav
Intermediate	10	Multi-step text editing, Settings navigation, Clipboard
Advanced	5	Multi-app workflows, Data transfer, Error recovery

Table 2: AgenticOS benchmark suite composition.

```
BenchmarkTask(
    task_id="basic_notepad_type",
    name="Type␣in␣Notepad",
    category="basic",
    instruction="Open␣Notepad,␣type␣'Hello␣World',␣"
                "then␣save␣as␣hello.txt␣on␣Desktop",
    expected_outcome="File␣hello.txt␣exists␣on␣Desktop␣"
                     "with␣content␣'Hello␣World'",
    max_steps=10,
)
```

5.2 Comparison with Existing Systems

System	Grounding	Architecture	MCP	Open	OSWorld*
AgenticOS (ours)	UIA+VLM+OCR	Modular ReAct	✓	✓	—
UFO^2	UIA+Vision	Dual-agent	—	✓	30.5%
Operator	Vision	CUA	—	—	20.8%
Navi	Vision	Foundation	—	—	19.5%
Claude CU	Vision	ReAct	—	—	—
OmniParser	Vision+OCR	Parsing only	—	✓	—

Table 3: Comparison of desktop automation systems. *OSWorld results are on Ubuntu; Windows results may differ. "—" indicates not reported or not applicable.

5.3 Analysis

Grounding advantages: The hybrid three-layer approach provides several advantages over pure vision grounding:

- UIA provides pixel-perfect element boundaries and control types without model inference cost.
- VLM fallback handles custom-rendered UIs where UIA has no accessibility support.
- OCR supplements both with text content, critical for form fields and labels.

Modularity benefits: Each component can be independently:

- **Tested**: Unit tests mock dependencies at layer boundaries.
- **Replaced**: Swap VLM providers, action executors, or grounding strategies.
- **Extended**: Add new action types or grounding methods without modifying core logic.

MCP integration: By exposing tools via MCP, AgenticOS can be used as a "desktop backend" for any MCP-compatible LLM client, enabling:

- Claude Desktop integration for direct desktop control.
- Multi-agent orchestration where AgenticOS handles UI while other agents handle reasoning.
- Tool chaining with other MCP servers (file system, databases, APIs).

6 Discussion

6.1 Limitations

1. **Windows-only**: Current UIA grounding is Windows-specific. Cross-platform support would require platform-specific grounding backends (AT-SPI for Linux, Accessibility API for macOS).
2. **LLM dependency**: The system's effectiveness is bounded by the reasoning capabilities of the underlying LLM. Local models via Ollama may have significantly lower success rates.
3. **Latency**: The observe-think-act loop introduces latency per step (2–5 seconds depending on model and grounding complexity), making real-time interaction challenging.
4. **Security**: Desktop automation inherently carries risks. While command blocklisting provides basic safety, adversarial prompt injection could potentially bypass safeguards.

6.2 Future Work

1. **Learning from demonstrations**: Recording expert sessions and fine-tuning grounding/action models on task-specific data.
2. **Parallel action execution**: Identifying independent action sequences that can be executed concurrently.
3. **Cross-platform support**: Extending grounding to Linux (AT-SPI) and macOS (AXUIElement).
4. **Reinforcement learning**: Using benchmark results as reward signals to improve navigation policies.
5. **OSWorld evaluation**: Running the full OSWorld benchmark suite for standardized comparison.

7 Conclusion

We presented AgenticOS, a modular framework for intelligent desktop automation that introduces hybrid three-layer grounding (UIA + VLM + OCR), a clean layered architecture with independent observation, grounding, action, and agent components, and native MCP integration for extensible tool access. Our 30-task benchmark suite provides standardized evaluation across basic, intermediate, and advanced desktop automation tasks. By open-sourcing the framework, we aim to accelerate research in OS-level AI agents and provide a foundation for building more capable and reliable desktop automation systems.

References

Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku. Technical report, Anthropic, 2024.

Anthropic. Model Context Protocol specification. `https://modelcontextprotocol.io/`, 2024.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, et al. Windows Agent Arena: Evaluating Multi-Modal OS Agents at Scale. *arXiv preprint arXiv:2409.08264*, 2024.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. *arXiv preprint arXiv:2401.10935*, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, et al. AgentBench: Evaluating LLMs as Agents. *arXiv preprint arXiv:2308.03688*, 2023.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. OmniParser for Pure Vision Based GUI Agent. *arXiv preprint arXiv:2408.00203*, 2024.

Google DeepMind. Navi: A foundation model for GUI navigation. Technical report, Google, 2024.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, et al. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. *arXiv preprint arXiv:2404.07972*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *ICLR*, 2023.

Chaoyun Zhang, Sean Li, Jiaxu Qian, Ke Xu, Yu Kang, Si Qin, et al. UFO: A UI-Focused Agent for Windows OS Interaction. *arXiv preprint arXiv:2402.07939*, 2024.

Chaoyun Zhang, et al. UFO2: The Desktop AgentOS. *arXiv preprint arXiv:2504.14603*, 2025.