LLM-Based GUI Agents

Bridging Human Interfaces and AI

Daniel Homola, AI Project Lead at BMW Research

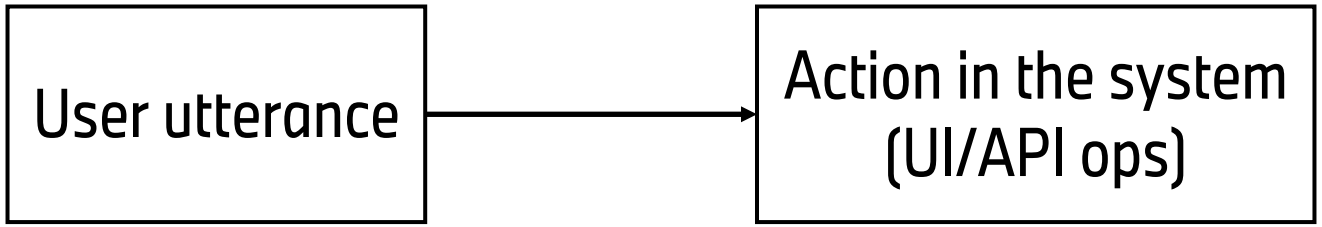

1.0 km
First Street
50
450 km
33
1.0 km
80 %
13 km
SW W NW
AQI 28
24
17:30
Suggestions
Weather
Parking Payments
Drive Recorder
Spotify
Beautiful Things
Benson Boone
22.0
22.0

THE FIRST-EVER

BMW PANORAMIC iDRIVE

Why (GUI) Agents, Why Now
Automotive & Beyond

- Agents can treat app UIs like APIs.
- Consumers switch platforms for better connected experiences; agents can orchestrate them.
- Imagine:
  - "Hey, buy milk, eggs, and pasta and have them delivered when I get home."
  - "Hey, book accommodation in Paris this weekend under €200/night, rating at least 8, near the Louvre."



Two Ways to Build Agents

API-first vs GUI-first

API Agent (What Most People Know)

- LLM picks **tools/APIs**, composes calls, returns results.
- Great for stable, documented APIs; narrow, high-precision tasks.
- Limitations:
  - Coverage gaps (depends on published endpoints or there is no API).
  - Execution is mostly hidden, no visual validation.
  - Brittle integrations across many providers.
  - Cost (sometimes paid APIs).

GUI Agent (When APIs Fall Short)

- Operates by interacting directly with GUIs rather than invoking predefined functions.
- Highly adaptable; generalizes and acts on any app by reading UI tree, pixels, or both — then tapping/typing like a person.
- Learns interfaces the way humans do. Not just clicks: aims for long-horizon tasks.
- Navigates flows visually; more intuitive user experience.

Limits & Challenges of GUI Agents

Maintainability

Fragile to layout and UI redesigns; higher upkeep across app versions.

Security & safety

Broad UI access risks unintended ops; needs guardrails, scopes, confirmations.

Efficiency

Longer trajectories mean longer time to reach the goal and a higher chance of compounding errors.

Table 4. Strategic criteria for selecting agent paradigms.

Scenario	**Recommended Approach**	**Rationale**
Stable, well-documented APIs	API Agents	Exploit robust endpoints for speed and reliability
Performance-critical operations	API Agents	Reduce latency and overhead via direct function calls
Controlled access to applications	API Agents	Ensure safety and security
Legacy or proprietary software	GUI Agents	Automate tasks without requiring new backend integration
Visual validation or UI testing	GUI Agents	Verify on-screen text or elements directly
Interactive or graphical manipulation	GUI Agents	Seamlessly replicate human-like interactions with visual elements
Partial API coverage	Hybrid	Combine UI-based steps where APIs are unavailable with direct calls for data-heavy tasks
Future-proofing	Hybrid	Facilitate switching from GUI to API as endpoints evolve

From "API Agents vs. GUI Agents: Divergence and Convergence" [1]

Hybrid Systems (Best of Both)

- **API tools** where available.
- **GUI agent** as another tool for gaps and scalable service integration, collaborative user experience.
- **Voice + orchestrator** as the glue.

Our Build: From Prototype to Research MVP

Same brain, two runtimes

Track A: Prototype
(Python + ADB)

- **Fastest iteration.**
- Python-only agent; control a device/emulator via **ADB**.
- Actions: screencap, UI dump, input tap/swipe/text.
- Build your own framework or use one like **mobile-agent-v3**, **droidrun**.

Track B: Research MVP
(Android app + Agent API)

- Android app with **AccessibilityService** executes on-device.
- Call your small local model on-device or remote Agent API (**FastAPI**).
- Caveats: as a user app, no raw key events.

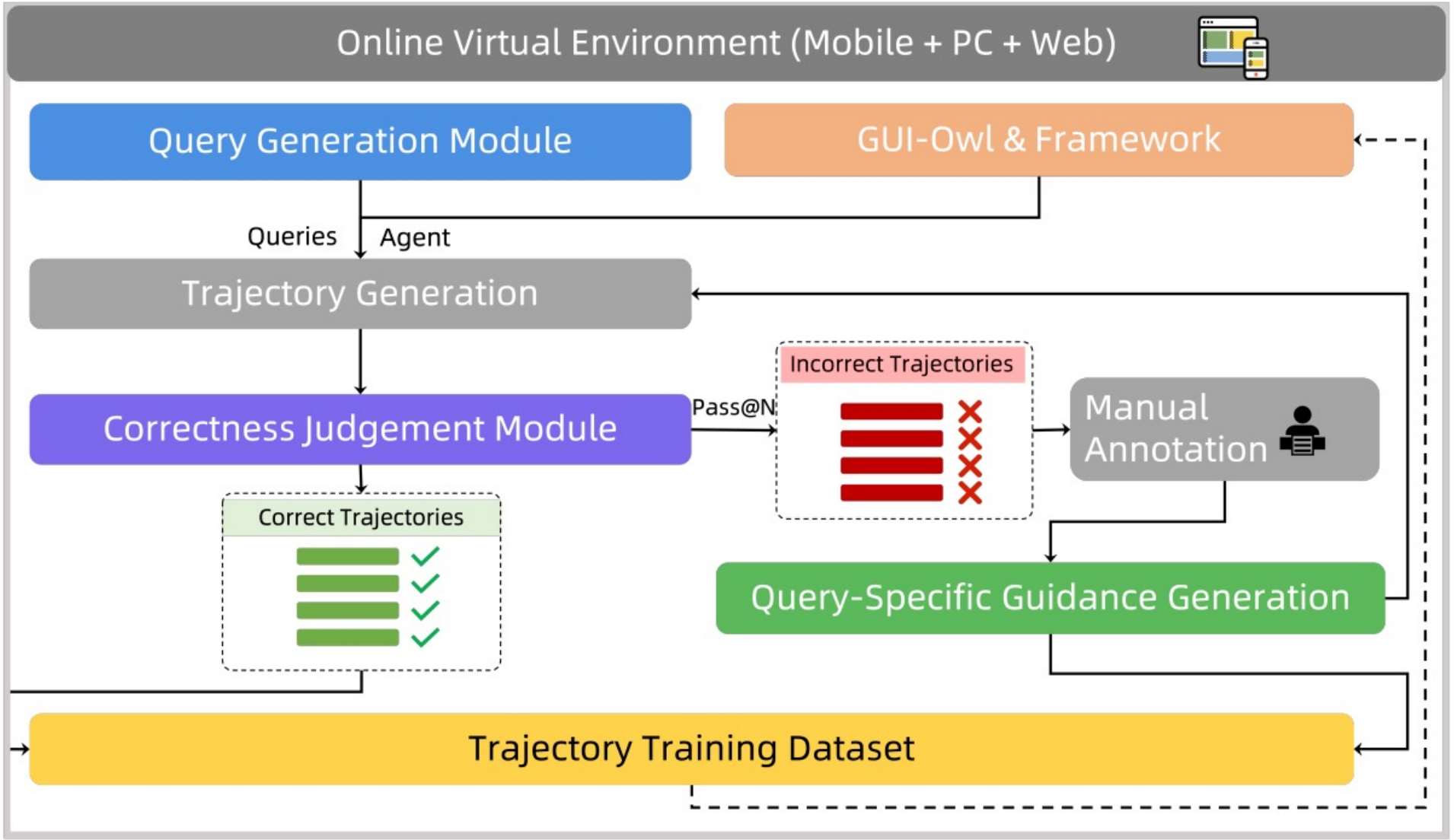
Practical Insights (What Worked for Us)

- Start atomic
- Evaluate in two layers
  - Bottom-up (atomic)
  - Top-down (tasks)
- Harden the executor
- Plan in chunks
- Observability: Log multimodal traces (tools like **Langfuse** help triage)
- Realtime voice: Build your own voice mode (tools like **LiveKit** for low-latency audio, plug-in your STT/TTS)

Model Choice Notes

- Mix specialized VLM with a tight executor.

- Go small when you can:
  - **GUI-Owl-7B/-32B** (built on the **Qwen2.5-VL** backbone); current state-of-the-art for GUI automation.
  - Foundational design: GUI-Owl unifies perception, grounding, reasoning, planning, and action execution in one model; works across platforms (mobile, desktop, web).



From "Mobile-Agent-v3: Fundamental Agents for GUI Automation" [2]

Papers to Look Into

API Agents vs. GUI Agents: Divergence and Convergence [1] (Microsoft): comparative study.

Mobile-Agent-v3: Fundamental Agents for GUI Automation [2] (Alibaba): advanced framework using GUI-Owl as the brain.

Small Language Models are the Future of Agentic AI [3] (Nvidia): small, specialized, fast.

3 Takeaways

Hybrid complementarity: use APIs (speed, security, reliability) when available, and use the GUI agent (human-machine collaboration, coverage, visual on-screen validation) when it fits the task. End-to-end workflows can be API-only, GUI-only, or hybrid.

How to ship: Prototype with a device bridge (e.g., ADB), productize a platform executor + Agent API.

What matters:

- Deterministic execution (e.g. executor-side heuristics).
- Evals (atomic→task).
- Observability (debug actions, timings).

Resources

[1] C. Zhang, S. He, L. Li, S. Qin, Y. Kang, Q. Lin, S. Rajmohan, and D. Zhang, "API Agents vs. GUI Agents: Divergence and Convergence," 2025. [link]

[2] J. Ye, X. Zhang, H. Xu, H. Liu, J. Wang, Z. Zhu, Z. Zheng, F. Gao, J. Cao, Z. Lu, J. Liao, Q. Zheng, F. Huang, J. Zhou, and M. Yan, "Mobile-Agent-v3: Fundamental Agents for GUI Automation," 2025. [link]

[3] P. Belcak, G. Heinrich, S. Diao, Y. Fu, X. Dong, S. Muralidharan, Y. C. Lin, and P. Molchanov, "Small Language Models are the Future of Agentic AI," 2025. [link]