Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-26)
Demonstrations Track

LLM-to-Map: Transparent Conversational Tool Orchestration for Real-Time Multi-Domain Simulation

Marouane Benbrahim , **Kavya Gautam** , **Zonghan Zhang** and **Zhiqian Chen**
Mississippi State University
mb4194@msstate.edu, kg1623@msstate.edu, zz239@msstate.edu, zchen@cse.msstate.edu

Abstract

Coupled power–traffic simulations are valuable for studying EV charging stress and outage propagation, but existing tools typically require scripts and opaque configurations. We present **LLM-to-Map**, a demo system that exposes a real-time multi-domain simulator through structured LLM tool orchestration. Natural-language requests are mapped to typed tool calls that coordinate SUMO traffic simulation, PyPSA power-flow analysis, V2G actions, and map operations. The agent operates over a fixed tool schema and emits auditable execution logs in the UI so users can inspect every action and parameter. Tool execution is deterministic: each request resolves to explicit API calls, and unsafe or destructive actions can require confirmation. The tool layer provides JSON schemas and parameter validation, preventing arbitrary code execution and enabling reproducible runs. The agent receives live system state (loads, EV statistics, V2G status, time, temperature) to support context-aware multi-step commands; when an LLM is unavailable, a deterministic parser preserves the same tool interface. The backend synchronizes cross-domain events (EV charging demand and substation failures) and streams state updates to a 3D map interface. We describe the architecture, coupling and synchronization, and a demo workflow that showcases multi-step scenario control and reporting for non-experts without writing scripts. Demo video: https://youtu.be/CpEgCZPl_2g; code: https://github.com/MarouaneBenbrahim/Map-LLM.

1 Introduction

Urban transportation and power networks are tightly interdependent: traffic signals and EV charging depend on electric supply, while grid load and restoration dynamics increasingly depend on mobility patterns. Existing co-simulation platforms provide powerful infrastructure models, but they are often controlled by scripts or configuration files that are inaccessible to non-experts. Our goal is to lower this barrier with a natural-language interface while retaining explicit, inspectable control over simulator actions.

The core contribution is a structured LLM control layer with typed tools, deterministic execution, and transparent auditable logs; Figure 1 shows the overall workflow.

Contributions. We make three contributions:

- Structured tool orchestration by an LLM over typed APIs.
- Deterministic execution with auditable logs.
- Demonstration of multi-step scenario control in a coupled power–traffic simulator.

2 Related Work

Large language models have begun to interface with maps and geospatial representations. MapGPT introduces map-guided prompting for vision-and-language navigation, using map structure to support global exploration [Chen *et al.*, 2024]. GeoGPT targets general geospatial tasks through an autonomous GPT-style interface [Zhang *et al.*, 2023]. Tag Map proposes a text-based map for LLM-driven spatial reasoning and navigation [Zhang *et al.*, 2024]. These systems emphasize map understanding, navigation, or planning, typically without explicit execution logs or safety controls.

LLM-to-Map differs in its interaction model and control guarantees. Unlike prior systems focused on navigation, this work performs *stateful, safety-constrained orchestrated control* of simulator actions via typed tools. The LLM is constrained to a fixed tool schema, actions are executed deterministically in the backend, and every tool call is surfaced in an auditable log for reproducibility.

We next describe the system architecture and coupling that enable this control layer.

3 System Overview

Figure 1 summarizes the architecture. The backend is a Flask service that coordinates the simulators and exposes tool APIs. The traffic domain uses SUMO [Lopez *et al.*, 2018] (via TraCI) to simulate vehicles and EV charging behavior. The power domain uses PyPSA [Brown *et al.*, 2018] to compute DC power flow in a synthetic midtown Manhattan distribution model. The UI is built with Mapbox GL JS (3D terrain/buildings enabled) and receives state updates over WebSocket (Socket.IO).

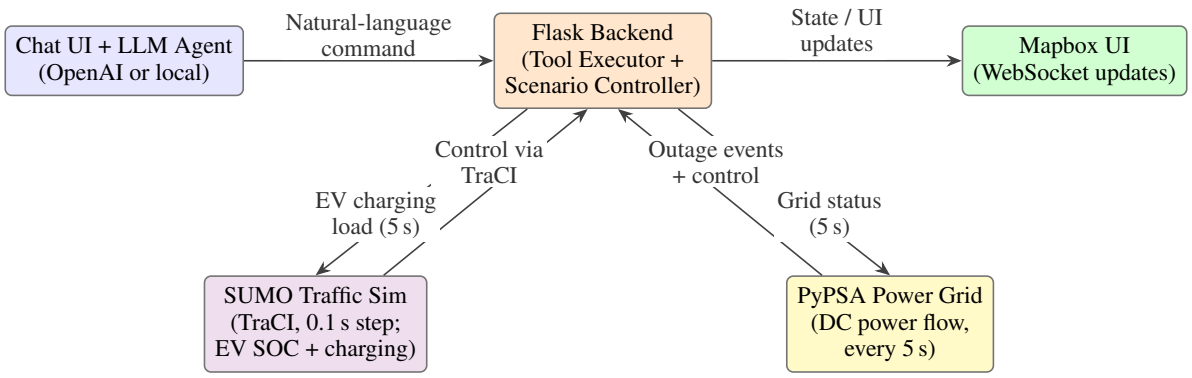

Figure 1: Workflow: natural language to tool calls, simulator execution, and UI updates.

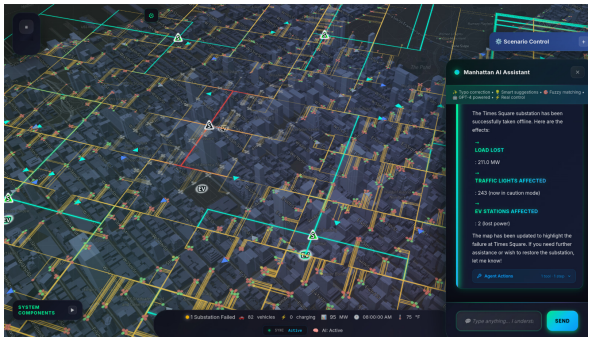

Figure 2: UI view with map layers and chat-based tool execution log.

Model scope. The demo includes eight substations (e.g., Times Square, Penn Station), a distribution network with primary/secondary cables, and 3,481 traffic-light locations loaded from a local dataset. Eight EV charging stations are modeled (20 chargers each) and assigned to the nearest substation by Manhattan distance. The road network is derived from OpenStreetMap and converted to SUMO format.

Visualization. The map (Figure 2) renders layered infrastructure (substations, traffic lights, cables, EV stations, and vehicles) and supports programmatic camera focus from the LLM (e.g., fly-to a substation). A compact side panel exposes KPIs and scenario controls; the chat panel surfaces tool execution steps for traceability.

Status and metrics. The dashboard continuously updates counts of online substations, aggregate load, EV fleet composition, charging activity, and vehicle-to-grid (V2G) participation. These metrics are derived from the same backend state used by the LLM, ensuring that conversational queries and visual indicators remain consistent.

Data inputs. Traffic-light locations are loaded from a local JSON dataset; when unavailable, the system can synthesize a Manhattan grid of intersections to maintain a complete signal map. This ensures the demo remains functional even if external preprocessing is skipped.

4 Coupling and Synchronization

The coupling layer explicitly propagates events across domains:

- **Grid → Traffic.** When a substation fails, the backend marks it offline, turns off powered traffic lights (visualized as black), and disables attached EV stations. Vehicles charging at those stations are released and rerouted to operational stations.
- **Traffic → Grid.** EV charging demand is aggregated per station and mapped to its parent substation. These dynamic EV loads are injected into the PyPSA model at regular intervals.

Synchronization runs in the main simulation loop: SUMO steps at 0.1s; EV load aggregation and PyPSA DC power flow update every 5s; V2G sessions update every 1s; traffic-light phases update on a 60s cycle. UI state is broadcast over WebSocket roughly every 0.5s at 1x simulation speed (configurable), keeping the demo responsive while preserving cross-domain causality.

Monitoring and safeguards. Each substation is monitored by a local utilization tracker. Warning, critical, and overload states are triggered at 85%, 95%, and 105% capacity, respectively; overload conditions sustained for 30 seconds automatically trip the substation to emulate protection logic. These events are logged and surfaced in the UI, enabling users to observe how load-driven failures propagate into traffic signals and charging stations.

Scenario controller. Time-of-day and temperature are managed by a scenario controller that updates load models and provides auto-advance time for repeatable demonstrations. This enables consistent test conditions (e.g., morning rush, heatwave) and supports structured evaluation of command sequences in the demo video.

5 Conversational Control and Transparency

User input is sent to a backend agent that uses OpenAI-compatible function calling to select and execute tools. The tool set contains 31 actions spanning substation control, simulation control, EV configuration, V2G activation, map focus, and layer toggling. Each tool execution emits progress events to the UI, and the resulting action list is displayed in

the chat panel ("Agent Actions") so users can verify what was executed.

```
{
  "name": "fail_substation",
  "parameters": {
    "substation_id": "string"
  }
}
```

Typed schemas constrain the LLM and ensure deterministic backend execution.

If the LLM is unavailable, the system falls back to a deterministic parser with fuzzy matching (Levenshtein distance) and explicit pronoun resolution, ensuring the demo still functions without external model access. For destructive or multi-step requests (e.g., blackout, resilience tests), the chat UI supports confirmation prompts and exposes each executed tool.

Tools are defined with typed parameters and return structured JSON, enabling reliable orchestration. The agent receives live system state each turn (substation loads, EV statistics, V2G status, time, and temperature), which supports context-aware follow-up requests such as "fail the most vulnerable substation" or "report current risk." Map-view actions are also exposed to the UI to keep camera changes visible and reproducible.

The tool layer prevents arbitrary code execution by restricting actions to predefined handlers. Destructive commands require explicit confirmation, and all tool calls are logged with parameters and timestamps for auditability.

6 Tool Coverage and Interaction Design

The tool inventory is organized around four interaction loops: (i) *grid operations* (fail/restore substations, blackout, load queries), (ii) *traffic and EV control* (start/stop simulation, spawn vehicles, EV configuration), (iii) *V2G management* (enable/disable V2G, resilience tests, session status), and (iv) *visual analytics* (map focus, layer toggles, report generation). Each tool returns a success flag and a structured response; optional map actions trigger explicit UI updates (e.g., fly-to a substation).

Multi-step requests are handled by macro tools that chain actions in a verifiable order. For example, a "prepare for a heatwave" command sets time and temperature, configures EVs, and starts or augments the simulation; a vulnerability analysis scans substation load ratios and ranks risk. This design keeps the LLM role narrow (choose tools and order) while keeping execution deterministic and observable in the UI.

7 Demo Walkthrough (Video)

The demo video follows a scripted sequence of natural-language commands aligned with the UI controls:

- **Basic control.** Start simulation with a chosen vehicle count and EV percentage; set time of day and temperature; focus the map on a specific substation.
- **Multi-step orchestration.** Issue a macro command (e.g., "prepare for a heatwave") that sets time/temperature and spawns vehicles.
- **Crisis management.** Trigger a blackout, enable V2G for a failed substation, and restore substations.
- **Analysis/reporting.** Request vulnerability analysis or generate a PDF situation report; tool results and report links appear in the chat panel.

8 Implementation and Availability

The demo is implemented in Python with Flask, SUMO, and PyPSA, and uses Mapbox GL JS for 3D visualization. The project includes an installation guide, scripted scenarios, and API endpoints for programmatic control.

The LLM interface supports both OpenAI-hosted models and local OpenAI-compatible servers via a configurable base URL; the model name is selected by configuration. This design allows the demo to run online or offline while keeping the same tool API and UI behavior.

9 Demo Setup and Reproducibility

The system runs as a single web application. A typical setup installs SUMO and Python dependencies, configures a Mapbox access token for the frontend, launches the Flask backend, and opens the dashboard in a browser. When an LLM is available, the backend uses `OPENAI_API_KEY` (cloud) or `OPENAI_BASE_URL` (local) and `LLM_MODEL` to select the model; otherwise, the deterministic fallback parser provides the same tool interface. The demo state can be reset via tool actions (e.g., restore all substations, stop simulation), enabling repeatable video segments and live demonstrations.

10 Extensibility and Deployment Considerations

The architecture is modular: the tool executor isolates simulator control, and the UI consumes a single WebSocket stream. New tools are added via typed schemas and handlers, which keeps side effects auditable. The same API supports manual and LLM-driven actions, enabling integration with other simulators without changing the interaction layer.

11 Intended Use and Evaluation Goals

The demo targets researchers, educators, and practitioners who need to explore cross-domain effects without writing simulator scripts. We focus on *interaction fidelity*, *scenario repeatability*, and *explainability of state changes*, emphasizing consistency between simulated state, LLM-visible context, and map visualization. The same tool interface supports controlled "what-if" experiments for teaching and rapid prototyping.

12 Limitations

LLM-to-Map focuses on interactive control rather than exhaustive validation. The LLM layer may misinterpret ambiguous commands, so tool execution is exposed to the user and destructive actions can be confirmed. The demo configuration is a synthetic midtown Manhattan model rather than a calibrated city-scale digital twin.

Ethical Statement

The demo uses public map data and synthetic load models; it does not ingest personal or user-specific data. Because LLM outputs can be incorrect or ambiguous, we prioritize transparency (tool logs) and explicit confirmation for destructive actions. The system is intended for research and education rather than real-world grid operations. The tool layer restricts execution to predefined schemas and prevents arbitrary code execution.

LLM-use disclosure. Large language models were used for language editing; all technical content was verified by the authors.

References

[Brown *et al.*, 2018] Tom Brown, Jonas Hörsch, and David Schlachtberger. PyPSA: Python for power system analysis. *Journal of Open Research Software*, 6(1):4, 2018.

[Chen *et al.*, 2024] Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee Wong. MapGPT: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9796–9810, 2024.

[Lopez *et al.*, 2018] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Lars Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using SUMO. In *Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.

[Zhang *et al.*, 2023] Yifan Zhang, Cheng Wei, Shangyou Wu, Zhengting He, and Wenhao Yu. GeoGPT: Understanding and processing geospatial tasks through an autonomous GPT. *arXiv preprint arXiv:2307.07930*, 2023.

[Zhang *et al.*, 2024] Mike Zhang, Kaixian Qu, Vaishakh Patil, Cesar Cadena, and Marco Hutter. Tag Map: A text-based map for spatial reasoning and navigation with large language models. *arXiv preprint arXiv:2409.15451*, 2024.