Agentic Workflow Architecture for Environmental Remote Sensing Analytics

Evgenij Shapovalov[1,], Artiom Hovhannisyan[1], Valentas Gruzauskas[1]*

[1] Artificial Intelligence Methods Lab, Institute of Computer Science, Faculty of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania, evgenij.shapovalov@mif.vu.lt, artiom.hovhannisyan@mif.vu.lt, valentas.gruzauskas@mif.vu.lt

* corresponding author

doi: 10.5281/zenodo.20407492

Abstract: Environmental remote sensing analysis requires complex workflows and domain expertise that many potential users lack. We present Terra AI, an agentic system in which a large language model orchestrates remote sensing tools and machine learning services, translating natural-language queries into executable multi-step workflows. The system integrates Google Earth Engine operations with independently deployed ML models — an algal bloom classifier and a peat moisture estimator — exposed as Model Context Protocol (MCP) servers. Each MCP server carries its own prompt instructions that encode domain-specific workflow rules, while a core system prompt provides general orchestration guidance. We evaluate orchestration reliability using a benchmark adapted from TaskBench, comparing a minimal and an optimal prompt configuration across 20 test cases. The optimal configuration improves tool selection F1 from 0.71 to 0.89 and tool ordering F1 from 0.79 to 0.99. Results indicate that explicit workflow rules are the dominant factor in reliable tool chaining, and that independently developed ML models can be made accessible to non-specialists through standardized tool interfaces.

Keywords: agentic GIS; remote sensing; tool-augmented LLM; prompt engineering.

1 Introduction

Remote sensing analysis underpins environmental monitoring and land management, yet even routine tasks — computing vegetation indices, classifying land cover, detecting algal blooms — require familiarity with sensor characteristics, cloud masking, and scripting interfaces to platforms such as Google Earth Engine (Gorelick et al. 2017). This creates a gap between users who need analytical products and the technical expertise required to produce them.

Recent work has shown that large language models (LLMs) can act as orchestrators of external tools rather than domain solvers (Schick et al. 2023, Yao et al. 2022). Akinboyewa et al. (2025) demonstrated an LLM-driven GIS Copilot within QGIS that translates natural language into geoprocessing workflows. However, such systems operate inside a single platform and do not address integrating independently developed ML models with heterogeneous interfaces.

In this paper, we address two questions: (1) whether an LLM-based agent can reliably orchestrate multi-step remote sensing workflows involving heterogeneous tools and independently developed ML models, and (2) what role domain-specific prompt engineering plays in achieving reliable orchestration.

2 Materials and methods

To address these questions, we developed Terra AI, a system that orchestrates heterogeneous remote sensing services through a unified natural-language interface. We evaluate orchestration reliability by comparing two prompt configurations using a benchmark adapted from TaskBench (Shen et al. 2024). The following subsections describe the system architecture, integrated models, prompt design, and evaluation framework.

2.1 System architecture

The system has three main parts (Figure 1). The Terra AI core is a web application (React, MapLibre GL) that combines a chat panel, an interactive satellite map, a map layer system for rendering analysis results, and the core system prompt. Users draw and name polygons on the map and reference them in chat with a backslash prefix (e.g., \LakeRėkyva). The frontend resolves names to GeoJSON coordinates, and dedicated result handlers decode each tool's output into the appropriate map layer.

The LLM (GPT-4o-mini via Vercel AI SDK) receives the user message together with polygon coordinates, tool schemas, conversation history, and the core system prompt, and returns tool call decisions. The Terra AI backend dispatches these tool calls to the appropriate MCP server. The LLM never performs geospatial computation — all domain logic resides in the tools.

All tools are organized as Model Context Protocol (MCP) servers (Anthropic 2025), listed in Table 1. A generic geospatial server wraps Google Earth Engine operations. Two additional servers wrap independently developed ML models, each backed by its own prediction service. Each MCP server carries its own prompt instructions for its tools, alongside the core system prompt held by Terra AI.

2.2 Prompt architecture

Prompt instructions are distributed across two levels. A core system prompt held by the Terra AI backend provides general orchestration guidance: the agent's role, polygon referencing semantics, date defaults, and interaction constraints. Each MCP server additionally carries its own prompt instructions that encode domain-specific workflow rules for its tools — for example, that a prerequisite extraction step must precede a dependent analysis, or that certain indices should not be computed over unsuitable land cover types (see Table 1 for the full tool inventory).

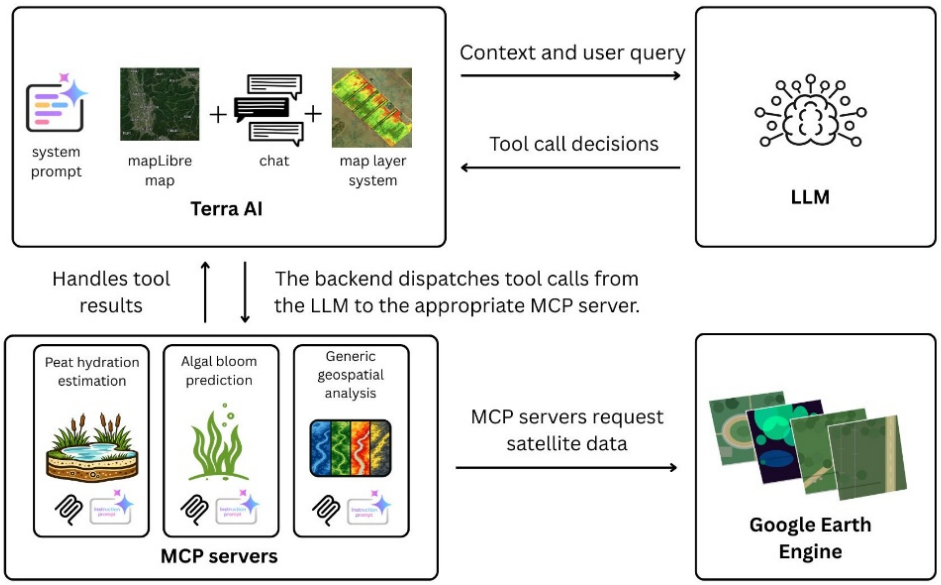

Figure 1. Terra AI system architecture. The Terra AI core hosts the chat interface, map, and core system prompt. The LLM receives context and returns tool call decisions. The backend dispatches tool calls to MCP servers, each of which carries its own prompt instructions and may request satellite data from Google Earth Engine.

This separation allows new tools to be added with their own workflow rules without modifying the core prompt.

We compared two configurations: a minimal one with only the core prompt and basic tool descriptions, and an optimal one that includes full per-server workflow rules, precondition checks, negative constraints, and in-context examples. All other components were held constant (same LLM, same tools, same test queries).

Table 1. Tool inventory.

Tool	Function	MCP server
classify_land_use	Land cover distribution (Dynamic World)	Geospatial
calculate_ndvi	Vegetation health statistics + tile URL	Geospatial
extract_class_polygons	Sub-polygons of a specific land class	Geospatial
calculate_algal_bloom	Bloom probability per pixel over water	Algal bloom
calculate_peat_hydration	Spatial peat moisture estimate	Peat hydration

2.3 Integrated ML models

Algal bloom detection wraps the classifier from Grendaitė and Petkevičius (2024), trained on 742 lake monitoring sites across the Baltic States (2017–2021) with 71.6% accuracy at a 10 mg/m^3 chlorophyll-a threshold. The service classifies each water pixel as bloom or non-bloom using Sentinel-2 spectral features. The agent's workflow extracts water polygons first, then runs bloom analysis per water body, returning coverage statistics rendered as a heatmap (Figure 2).

Peat hydration estimation wraps a 3D convolutional model from Komurcu et al. (submitted for publication), developed at Vilnius University under an ESA-funded project. The model fuses multi-temporal Sentinel-1 SAR with static site features and weather data. Inference is performed at known sample points within the user-defined polygon — locations where static features are available — and results are spatially interpolated to produce a continuous moisture map (Figure 3).

From the orchestration perspective both models function identically: the agent provides coordinates and a date, and the MCP server returns a structured result. The integration architecture is agnostic to model performance.

2.4 Evaluation framework

We evaluate orchestration correctness, not analytical accuracy. The framework is adapted from TaskBench (Shen et al. 2024): tool invocations are modelled as directed graphs and compared against reference solutions via structured JSON matching. Four metrics are computed: Node F1 (tool selection), Edge F1 (tool ordering), Parameter Name F1, and Parameter Name–Value F1. The test suite comprises 20 synthetic cases across five categories: single-step, conditional, chained, multi-region, and underspecified queries. The results should be interpreted as a proof-of-concept demonstration rather than a comprehensive benchmark.

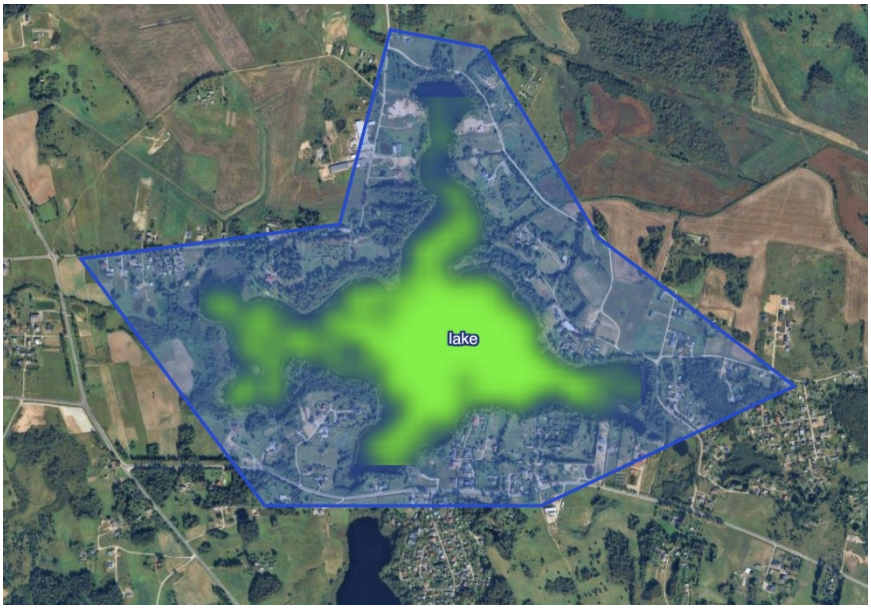

Figure 2. Algal bloom detection result. Green areas indicate predicted bloom presence over extracted water polygons. Query: "What was the state of water in terms of algal blooming in the lake in 2020-08-10?".

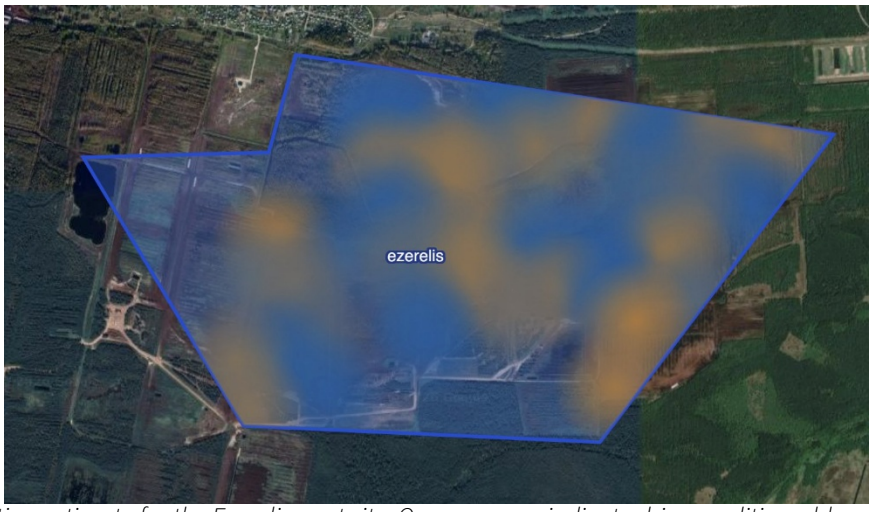

Figure 3. Peat hydration estimate for the Ezerelis peat site. Orange areas indicate drier conditions, blue areas indicate higher moisture, interpolated from Sentinel-1 SAR inference at known sample points. Query: "What level of peat hydration was in \ezerelis on 2020-07-10?".

3 Results

Table 2 reports macro-averaged F1 scores for both prompt configurations. The largest improvement is in Edge F1 (+0.20): the minimal configuration's primary failure mode is misordering tool calls. The optimal configuration achieves near-perfect ordering (0.99), indicating that per-server workflow rules encode precondition logic the LLM cannot reliably infer from tool descriptions alone. Parameter Name–Value F1 remains the weakest dimension (0.58 → 0.75), reflecting the difficulty of inferring concrete values from underspecified queries. Figure 2 and figure 3 illustrate end-to-end results for the two MCP servers with ML models: algal bloom and peat hydration.

Table 2. Macro-averaged F1 scores across 20 test cases.

Metric	Minimal	Optimal	Δ
Node F1 (tool selection)	0.71	0.89	+0.18
Edge F1 (tool ordering)	0.79	0.99	+0.20
Parameter Name F1	0.66	0.84	+0.18
Parameter Name–Value F1	0.58	0.75	+0.17

4 Discussion

The high tool-selection and ordering scores (0.89 and 0.99 F1) suggest that LLM-based orchestration of remote sensing workflows is practically viable. For most tested queries — including multi-step chains such as land cover classification followed by bloom detection — the system produced correct, executable analysis plans. A user who cannot write Google Earth Engine scripts can obtain a land cover breakdown, a vegetation health assessment, or an algal bloom map by describing what they need in plain language. These results are consistent with the findings of Akinboyewa et al. (2025), who demonstrated effective LLM-driven geoprocessing within QGIS. Our work extends this to a multi-server setting where independently developed models are orchestrated through standardised interfaces.

Achieving this reliability required encoding domain-specific rules into the prompt configuration — for example, that water polygons must be extracted before bloom analysis, or that vegetation indices should not be computed over areas consisting entirely of water, built-up surfaces, or snow. While Schick et al. (2023) showed that LLMs can learn to invoke tools, our results suggest that tool descriptions alone are insufficient for reliable multi-step sequencing — explicit procedural rules encoding domain preconditions were the dominant factor, accounting for the largest improvement in Edge F1. The distributed prompt architecture, where each MCP server carries its own workflow rules, proved practical: adding a new tool with its prerequisites does not require modifying the core system prompt.

Several limitations should be noted. The test suite is synthetic and limited to 20 cases. We evaluate workflow correctness, not scientific validity of outputs — that depends on the underlying models and data. The algal bloom model assumes shallow lake type due to lack of a depth database. The peat model is limited to areas where static site features are available. No user studies have been conducted.

5 Conclusions

This paper presented Terra AI, a system that allows users to perform environmental remote sensing analyses by describing what they need in plain language. The system correctly assembles multi-step analytical workflows for most tested queries. Making this work reliably required encoding domain-specific procedural rules into the system, distributed across a core prompt and per-server instructions. The architecture shows that independently developed ML models can be plugged in through standardised interfaces without modifying the core system. Future work should validate the system with domain scientists and expand the range of supported analyses.

Acknowledgments: This research was funded by the Research Council of Lithuania under the programme Information Technologies for the Development of Science and Knowledge Society, project No. S-ITP-25-3.

6 References

Akinboyewa, T., Li, Z., Ning, H., Lessani, M.N., 2025. GIS Copilot: towards an autonomous GIS agent for spatial analysis. International Journal of Digital Earth 18(1), 2497489.

Anthropic, 2025. Model Context Protocol, https://modelcontextprotocol.io (Accessed 26 May, 2026).

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R., 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. Remote Sensing of Environment 202, 18–27.

Grendaitė, D., Petkevičius, L., 2024. Identification of Algal Blooms in Lakes in the Baltic States Using Sentinel-2 Data and Artificial Neural Networks. IEEE Access 12, 27973–27988.

Komurcu, K., Bevainis, L., Gruzauskas, V., 2026. Temporally-Aware Multi-Sensor Fusion for Peatland Moisture Retrieval: Overcoming Asynchronous Observation Gaps with Deep Learning. Ecohydrology, submitted for publication.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., et al., 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. Advances in Neural Information Processing Systems 36, 68539–68551.

Shen, Y., Song, K., Tan, X., Zhang, W., et al., 2024. TaskBench: Benchmarking Large Language Models for Task Automation.

Yao, S., Zhao, J., Yu, D., Du, N., et al., 2022. ReAct: Synergizing Reasoning and Acting in Language Models. ICLR.