PROCEEDINGS OF SCIENCE

Exploiting the Model Context Protocol for Earth Observation Data Access and Workflow Orchestration

Claudio Pisa,[a,*] **Mohanad Albughdadi,**[a] **Marica Antonacci,**[a,b] **Vasileios Baousis,**[a] **Federico Fornari**[a] **and Tolga Kaprol**[a]

[a] *European Centre for Medium-Range Weather Forecasts (ECMWF), Shinfield Road, Reading RG2 9AX, United Kingdom*

[b] *INFN Sezione di Bari, Via Orabona 4, 70125 Bari, Italy*

E-mail: claudio.pisa@ecmwf.int, mohanad.albughdadi@ecmwf.int, marica.antonacci@ecmwf.int, vasileios.baousis@ecmwf.int, federico.fornari@ecmwf.int, tolga.kaprol@ecmwf.int

The EO4EU project democratises access to Earth Observation (EO) data by providing a comprehensive platform for a wide spectrum of stakeholders, from researchers to policymakers. The platform facilitates the retrieval of EO data and the orchestration of complex computational and machine learning workflows. To this end, it features a semantic knowledge graph to enhance EO metadata discoverability, supports data fusion across heterogeneous sources, and enables advanced visualisation, including immersive VR/XR environments. The Model Context Protocol (MCP) is a lightweight, standards-based protocol built on JSON-RPC that supports the dynamic coupling of Large Language Models (LLMs) with external data and services. MCP introduces a set of abstractions tailored to the operational logic of LLMs, namely tools, resources, and prompts, enabling the integration of existing platforms with LLMs regardless of their accessibility model. As part of the EO4EU project, we have developed a prototype MCP server that encapsulates the EO4EU REST APIs into MCP-compliant tools. Each tool performs a goal-oriented operation, such as data discovery or workflow management, exposing an LLM-interpretable and actionable description to MCP clients. Lessons learned include the importance of managing the LLM context window: verbose API responses can quickly exhaust it, but the MCP server can mitigate this by filtering out irrelevant information and by mapping each MCP tool to multiple API endpoints. Nevertheless, several challenges remain, including LLM hallucinations that may lead to infinite MCP tool-invocation loops.

International Symposium on Grids and Clouds (ISGC2026)
15–20, March, 2026
Academia Sinica, Taipei, Taiwan

*Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0) All rights for text and data mining, AI training, and similar technologies for commercial purposes, are reserved. ISSN 1824-8039 . Published by SISSA Medialab.

PoS(ISGC2026)003

1. Introduction

Earth Observation (EO) data has become indispensable for addressing global challenges ranging from climate change monitoring to disaster response and sustainable development. The volume and variety of EO data continue to grow at an unprecedented rate, driven by constellations of satellites such as the Copernicus Sentinels [1], commercial satellite operators, and ground-based sensor networks. While the availability of these data represents a major opportunity for science and policy, the complexity of discovering, accessing, and processing heterogeneous EO datasets remains a significant barrier for many user communities.

The EO4EU project [2] aims to lower these barriers by providing a unified platform for EO data access, processing, and visualisation. The EO4EU platform features a semantic knowledge graph for metadata-driven data discovery, a Domain-Specific Language (DSL) workflow engine for computational and machine learning pipelines, and a range of options for exploring and visualising data and processing outputs. The platform serves a broad spectrum of users, from domain scientists who require tailored datasets to policymakers who need derived information products.

At the same time, Language Models (LMs) and especially Large Language Models (LLMs) have emerged as powerful interfaces for human-computer interaction. Their ability to interpret natural language instructions and orchestrate complex tasks through tool use has opened new avenues for making computational platforms more accessible. Rather than requiring users to learn platform-specific APIs or graphical interfaces, a chat-based front-end backed by an LLM can allow them to express their data needs and processing requirements in plain natural language.

However, integrating LLMs with existing platforms is far from straightforward. Directly exposing REST API specifications to an LLM results in poor tool selection: language models may struggle to choose from large sets of undifferentiated API endpoints whose descriptions are written for developers rather than for AI agents [3]. A mediation layer is needed that presents platform capabilities in terms that are interpretable and actionable by LLMs.

The Model Context Protocol (MCP) [4] addresses this need. MCP is an open, lightweight protocol built on JSON-RPC that defines a standard way for LLM-based applications to discover and invoke external tools, access data resources, and receive contextual prompts. MCP introduces three core abstractions: *tools*, which represent executable operations; *resources*, which encapsulate data or metadata; and *prompts*, which provide structured guidance for task execution. This design enables any platform with a REST or programmatic API to be made accessible to LLMs through an MCP server, regardless of the LLM's deployment or licensing model: local or remote, proprietary or open source/open weight.

This paper presents our work on developing a prototype MCP server for the EO4EU platform. We describe the architecture and implementation of the server, the design decisions that shaped it, and the lessons learned from integrating it with different LLMs. The work is situated within the broader context of the DeployAI project [5] and the European AI-on-Demand (AIoD) platform [6], which aim to provide accessible, ethical AI solutions and integrate with Common European Data Spaces [7].

The remainder of this paper is organised as follows. Section 2 provides background on the EO4EU platform, the MCP protocol, AIoD, and the broader European ecosystem. Section 3 describes the design and implementation of the EO4EU MCP server. Section 4 discusses the

PoS(ISGC2026)003

integration with AI tools and the demonstration setup. Section 5 presents the lessons learned and discusses open challenges. Section 6 concludes the paper.

2. Background and related work

2.1 The EO4EU platform

EO4EU [2] is a Horizon Europe-funded project, involving 19 organisations from 10 countries, aimed at making EO data more accessible and usable for environmental, governmental, and business applications by leveraging artificial intelligence (AI) tools and extended reality interfaces. The platform architecture, shown in Figure 1, comprises several interconnected components.

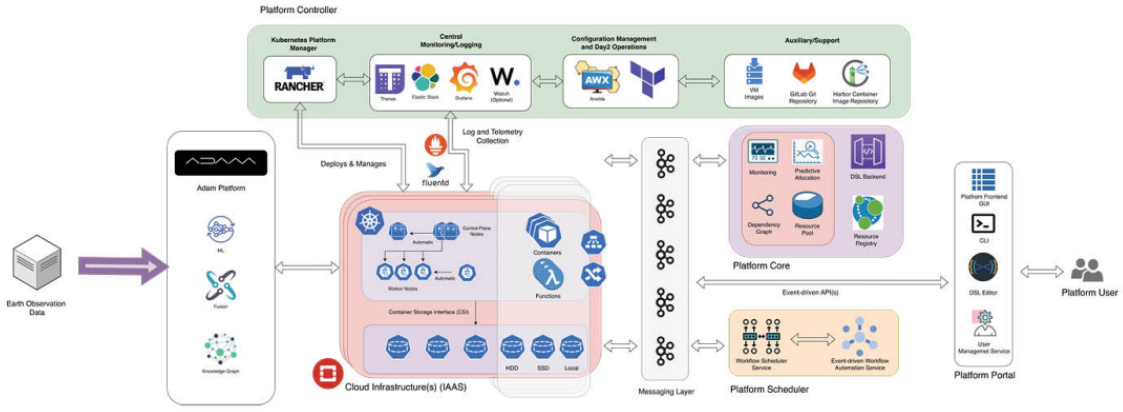

Figure 1: The EO4EU platform architecture, featuring Knowledge Graph, DSL workflow engine, and advanced visualisation interfaces. EO4EU relies on a distributed multi-cloud infrastructure, based on OpenStack and Kubernetes, on top of which cloud-native tools and EO4EU platform components are deployed. Communication between components, observability, orchestration of workflows and machine learning play key roles behind the scenes.

The **Knowledge Graph** (KG) layer provides semantic enrichment of EO metadata, enabling users to discover datasets through keyword-based queries that go beyond simple catalogue searches. The KG integrates metadata from diverse sources, including Copernicus, Sentinel satellites, EUMETSAT, CMCC, and the INSPIRE directive [8], linking datasets through semantic relationships.

The **DSL Backend** implements a workflow engine based on a domain-specific language for defining computational pipelines. Users can create, parameterise, execute, and monitor workflows that perform data retrieval, processing (including machine learning inference), and result generation. Workflows are deployed on cloud infrastructure managed by Kubernetes and can access GPU-accelerated resources.

The platform exposes its functionality through REST APIs, a web-based user interface, a command-line interface, and a DSL editor. It also supports advanced visualisation, including VR/XR environments [2].

2.2 The Model Context Protocol

The Model Context Protocol (MCP) [4] is an open protocol originally developed by Anthropic and now adopted by a growing ecosystem, including Docker's MCP Catalog [9]. MCP is designed

PoS(ISGC2026)003

to standardise the way LLM-based applications interact with external systems.

MCP is built on JSON-RPC 2.0 [10], a stateless, transport-agnostic remote procedure call protocol. On top of this foundation, MCP defines three categories of server-side abstractions:

- **Tools**: executable operations that an LLM can invoke. Each tool has a name, a natural-language description (written for the LLM, not for developers), and a typed parameter schema. The LLM selects and invokes tools based on the user's intent.

- **Resources**: read-only data endpoints identified by URIs. Resources can provide access to API specifications, configuration data, or dynamic content such as workflow status.

- **Prompts**: parameterised prompt templates that guide the LLM in performing multi-step tasks. Prompts encode domain expertise as reusable interaction patterns.

The MCP architecture follows a client-server model. An *MCP host* (typically an LLM-powered application) contains an *MCP client* that connects to one or more *MCP servers*, each exposing tools, resources, and prompts for a specific domain. The transport layer supports both local (standard I/O) and remote (HTTP-based) communication.

A key insight behind MCP is that exposing raw API endpoints to an LLM is ineffective. API documentation is written for human developers and often describes implementation details rather than user-facing goals [11]. MCP tools, by contrast, are described in terms of the goals they achieve, allowing the LLM to reason about which tool to use based on the user's intent rather than on API semantics.

2.3 The DeployAI and AI-on-Demand ecosystem

The EO4EU MCP server is being developed within the broader context of the DeployAI project [5] and the European AI-on-Demand (AIoD) platform [6]. DeployAI's mission is to deliver a production-grade AI-on-Demand platform and marketplace that offers DevOps tools, AI models, real-world use cases, and access to cloud and HPC infrastructures.

A central component of the AIoD platform is the **AI Builder** [12], a visual workflow editor that allows users to compose AI pipelines from reusable modules and deploy them on various execution environments, including cloud infrastructure and HPC clusters. The AIoD platform also integrates with Common European Data Spaces [7], providing access to shared data ecosystems across sectors including the Green Deal Data Space.

As part of this integration, ECMWF has developed a **Copernicus Climate Data Store (CDS) connector** [13] that provides Copernicus ERA5 reanalysis data through the AI Builder. This connector, implemented as a gRPC-based service, allows AI pipelines to ingest climate and weather data directly.

The EO4EU MCP server complements these efforts by enabling LLM-based interaction with the EO4EU platform, which itself aggregates data from multiple sources including Copernicus, Destination Earth [14], and other European and international EO data providers.

PoS(ISGC2026)003

2.4 Related work

The integration of LLMs with domain-specific platforms is an active area of research and development. Tool-augmented LLMs [15] have demonstrated the ability to select and invoke external functions to overcome the limitations of parametric knowledge. Frameworks such as LangChain [16] and LlamaIndex [17] provide abstractions for building LLM-powered applications that interact with external data and tools. However, these frameworks typically require tight coupling between the application and the tools, making it difficult to reuse tool implementations across different LLM hosts.

MCP addresses this interoperability gap by providing a protocol-level standard. Since its release, MCP has gained traction in the developer ecosystem, with MCP servers available for databases, file systems, version control, and various web services [9]. To our knowledge, the work presented here is among the first to apply MCP to Earth Observation data platforms, bridging the gap between the EO and AI communities at the protocol level.

In the geospatial domain, efforts such as the Open Geospatial Consortium (OGC) API standards [18] and OpenEO [19] have established interoperability at the API level for EO data access and processing. Our MCP server builds on top of the EO4EU platform's existing OpenEO-compliant APIs and adds an LLM-accessible layer, complementing rather than replacing these standards.

3. The EO4EU MCP server

3.1 Architecture and design

The EO4EU MCP server is implemented in Python and follows the standard MCP client-server architecture, as illustrated in Figure 2. The server acts as a bridge between MCP clients (and their associated LLMs) and the EO4EU platform APIs.

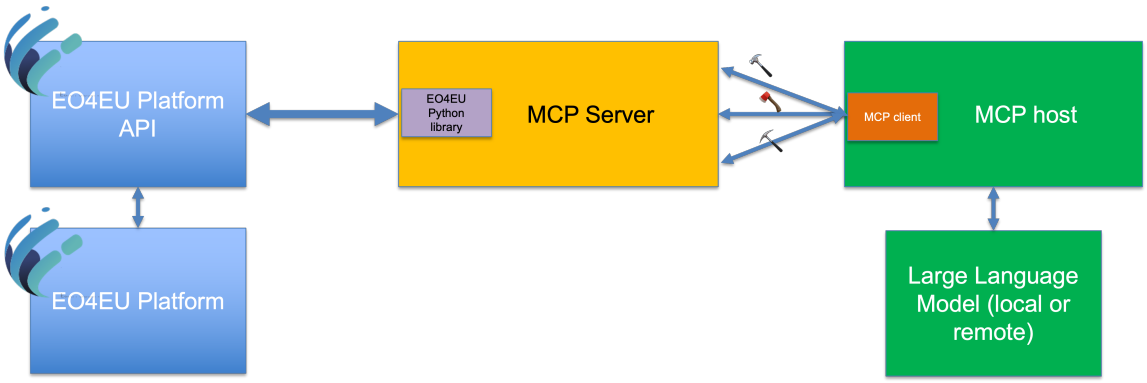

Figure 2: Integration architecture showing the EO4EU MCP server mediating between the EO4EU platform APIs and an MCP host with an LLM. The MCP server encapsulates the EO4EU Python client library and exposes platform capabilities as MCP tools, resources, and prompts.

The server is built using the `FastMCP` class from the Python MCP SDK [20], on top of an EO4EU Python client library that wraps the platform's REST APIs into a typed Python interface with sub-clients for the Knowledge Graph, the DSL workflow backend, and S3 storage. Authentication is handled by a dedicated `EO4EUAuth` class with automatic token refresh, ensuring uninterrupted

PoS(ISGC2026)003

operation during long-running sessions. Configuration is managed through Pydantic models loaded from YAML, with environment variable overrides for runtime credential injection.

3.2 MCP tools

The EO4EU MCP server exposes tools organised into three categories: Knowledge Graph tools, DSL workflow tools and S3 storage tools. Each tool is registered with the MCP server using Python's decorator pattern (`@server.tool()`) and includes a natural-language description written for LLM consumption. A non-exhaustive list of these tools is provided hereafter.

3.2.1 Knowledge Graph tools

Two tools provide access to the EO4EU semantic knowledge graph:

- `search_datasets_tool`: accepts a list of keywords and queries the knowledge graph to return matching datasets. Internally, it calls `client.kg.execute_query()` to build a query, then `client.kg.search_datasets()` to retrieve results.
- `get_dataset_details_tool`: retrieves detailed metadata for a given dataset identifier, including title, description, and persistent identifier.

3.2.2 DSL workflow tools

Tools to manage the workflow lifecycle:

- `list_workflows_tool`: lists workflows filtered by status (e.g., `DRAFT`, `RUNNING`, `ERROR`), also gathering workflow details.
- `start_workflow_tool`: initiates workflow execution.
- `stop_workflow_tool`: stops a running workflow.
- `delete_workflow_tool`: removes a workflow.
- `create_stress_test_workflow_tool`: creates a stress test workflow for platform performance evaluation.

3.2.3 S3 storage tools

Tools which provide access to workflow results:

- `list_s3_buckets_tool`: lists available S3 buckets.
- `list_s3_objects_tool`: lists objects within a bucket, with optional prefix filtering.
- `download_s3_object_tool`: downloads a specific object to a local path.
- `download_workflow_results_tool`: downloads all results from a completed workflow, with optional file pattern filtering (e.g., `*.tif`).

PoS(ISGC2026)003

3.2.4 Tool description design

A critical design decision was the writing of tool descriptions. Each tool's description is formulated in terms of user-facing goals rather than API operations. For example, the description of `get_dataset_details_tool` reads: "*Get, from the EO4EU platform, detailed dataset information, i.e. dataset details, for a given dataset ID. Call this tool when you have already obtained the dataset id (e.g. from the user, from previous conversations or from the search_datasets_tool).*" This description not only explains what the tool does but also provides guidance on when to use it, helping the LLM make appropriate tool selections.

3.3 MCP resources

The server exposes one MCP resource, `eo4eu://openapi/main`, which provides the main EO4EU OpenAPI specification. This allows the LLM to inspect the platform's API structure without invoking tools.

3.4 MCP prompts

MCP prompts encode domain expertise as reusable interaction templates. The server provides two prompts: `find_and_process_data_prompt`, for data discovery and processing guidance; and `troubleshoot_workflow_errors_prompt`, for systematic error diagnosis covering script, data access, resource, and system errors. These prompts serve as structured plans that the LLM can follow step by step, reducing the likelihood of missing critical steps or invoking tools in an incorrect order.

4. Integration with AI tools

4.1 Integration with the AIoD ecosystem

The EO4EU MCP server is designed to operate within the broader AI-on-Demand (AIoD) ecosystem. As shown in Figure 2, the MCP server connects the EO4EU platform (and, through it, multiple EO data sources including Copernicus, Destination Earth, EUMETSAT, and Sentinel) with AI tools available on the AIoD platform.

This integration complements the CDS connector, which provides Copernicus ERA5 climate data to AI Builder pipelines via gRPC. While the CDS connector operates at the data pipeline level, the MCP server operates at the interaction level, enabling users to explore, query, and orchestrate EO4EU capabilities through natural language.

Together, these integration mechanisms are devised to allow the AIoD platform to access EO data from the Green Deal Data Space sources [7] through multiple channels suited to different user interaction models: programmatic pipelines (via AI Builder and the CDS connector) and conversational interfaces (via the MCP server and an LLM).

4.2 Demonstration setup

We demonstrated the EO4EU MCP server using LM Studio [21] as the MCP host, configured with a locally deployed Qwen3-8B model [22]. The setup consists of:

PoS(ISGC2026)003

1. The EO4EU platform running its production APIs.
2. The EO4EU MCP server running locally, connected to the platform via HTTPS.
3. LM Studio acting as the MCP host, connecting to the MCP server via HTTP.
4. The Qwen3-8B LLM running locally on consumer hardware.

Through this setup, users interact with the EO4EU platform entirely through a chat interface. The LLM interprets user requests (e.g., "My use case is preventing fires in Crete. Please suggest 3 datasets from EO4EU."), selects the appropriate MCP tools, invokes them, and presents the results in natural language. Multi-step workflows, such as searching for datasets, starting a processing workflow, and downloading results, are handled through a sequence of tool invocations orchestrated by the LLM.

A key advantage of the MCP architecture is model independence. The same MCP server can be used with any LLM that supports tool use, whether proprietary or open source/open weight, deployed locally or accessed via API. This flexibility allows users to choose the LLM that best fits their privacy, performance, and cost requirements.

5. Lessons learned and discussion

The development and testing of the EO4EU MCP server yielded several practical insights about integrating LLMs with platform APIs through MCP.

5.1 Context window management

The most significant challenge was managing the LLM's context window. EO platform APIs often return verbose JSON responses with extensive metadata and nested structures irrelevant to the user's immediate query, which can rapidly exhaust the context window in a multi-turn conversation.

The MCP server mitigates this problem by acting as a filter: each tool extracts only the relevant fields from the API response and formats them into concise structures. For example, the `search_datasets_tool` extracts only the dataset identifier, persistent identifier, and title from the full knowledge graph response, discarding dozens of metadata fields that are not needed for the user's initial exploration.

This filtering has an architectural implication: the **one-to-many mapping** between MCP tools and API endpoints. A single MCP tool may invoke multiple API calls internally (e.g., `search_datasets_tool` first calls `execute_query` then `search_datasets`), but presents a single, focused result to the LLM. This reduces the number of tools the LLM must reason about and keeps responses compact.

5.2 LLM hallucinations and tool invocation loops

During testing, we observed that LLMs occasionally hallucinate tool parameters or misinterpret tool results, leading to incorrect subsequent actions. Tool hallucinations in agentic settings are a documented failure mode, classified into tool selection hallucinations (choosing the wrong tool) and tool usage hallucinations (fabricating or misusing parameters) [23]. A particularly problematic

PoS(ISGC2026)003

failure mode is the infinite tool invocation loop, where the LLM repeatedly invokes the same tool with slight variations of the same parameters, expecting different results.

These loops occur when the LLM does not correctly interpret an error response, or when the context window approaches its capacity, or when it lacks sufficient information to determine that a previous invocation already provided the needed result [23].

As a mitigation, we found it effective to limit the number of results returned by search tools, and in general limiting the usage of the context window by reducing the verbosity of the invoked tools. Other approaches which can be found in the literature include: providing explicit conditional guidance in tool descriptions to improve the model's reasoning about when and whether to invoke a tool [24, 25], and designing prompts that establish clear stopping conditions to prevent unbounded loops [26].

5.3 Comparison with direct API exposure

An alternative to the MCP approach would be to expose the EO4EU OpenAPI specification directly to the LLM and let it generate HTTP requests. We considered this approach but found it impractical for several reasons:

1. The EO4EU API has dozens of endpoints with complex parameter schemas. LLMs perform poorly when selecting from large, flat lists of similar-looking options [3].

2. API descriptions are written for developers, not for AI agents [25]. They describe HTTP methods and URL patterns rather than user-facing goals.

3. The absence of a mediation layer eliminates the opportunity to filter verbose responses, leading to rapid context window exhaustion.

The MCP server approach addresses all the above-mentioned issues by providing a curated, goal-oriented interface with built-in filtering and error handling.

5.4 Future work

Several directions remain for future development. First, we plan to extend the MCP server with additional tools for data visualisation, enabling the LLM to generate maps and charts from EO data products. Second, we intend to integrate the MCP server with the AIoD AI Builder, allowing composability with other tools. Finally, addressing tool invocation loops and hallucination mitigation remains an important direction.

6. Conclusions

We have presented the design, implementation, and deployment of an MCP server for the EO4EU Earth Observation platform. The server exposes several tools across three categories (Knowledge Graph, DSL workflows and S3 storage), resources, and prompts, enabling LLM-based applications to discover datasets, manage computational workflows, and retrieve processing results through natural language interaction.

PoS(ISGC2026)003

Our experience demonstrates that the MCP protocol provides an effective abstraction layer for integrating complex scientific platforms with LMs. The key design principles that emerged from this work are: (i) use goal-oriented tool descriptions rather than API-level descriptions; (ii) filter verbose API responses at the server side to manage context window consumption; and (iii) employ one-to-many mappings between MCP tools and API endpoints.

The EO4EU MCP server is devised to operate within the broader European AI-on-Demand ecosystem, complementing the CDS connector and the AI Builder to provide multiple interaction modalities for accessing Earth Observation data from the Green Deal Data Space. By enabling LLM-driven access to EO data, this work contributes to lowering the technical barriers for a wider community of users, from researchers to policymakers, to benefit from the growing wealth of Earth Observation data.

Acknowledgments

This work has been supported by the European Union's Horizon Europe programme through the EO4EU project (Grant Agreement No. 101059548) and the Digital Europe Programme through the DeployAI project (Grant Agreement No. 101146490).

References

[1] European Commission. Copernicus: Europe's Eyes on Earth. https://www.copernicus.eu/. accessed March 2026.

[2] EO4EU Consortium. EO4EU – Earth Observation for the European Union. https://www.eo4eu.eu/. accessed March 2026.

[3] Daniel Maninger et al. Evaluating and mitigating errors in LLM-generated web API integrations, 2025.

[4] Anthropic. Model Context Protocol specification. https://modelcontextprotocol.io/, 2024.

[5] DeployAI Consortium. DeployAI – Deploying AI-on-Demand for Europe. https://www.2030.ai/. accessed March 2026.

[6] European Commission. AI-on-Demand Platform. https://www.2030.ai/. accessed March 2026.

[7] European Commission. A European Strategy for Data. COM(2020) 66, 2020.

[8] European Commission. INSPIRE Directive: Infrastructure for Spatial Information in the European Community. Directive 2007/2/EC, 2007.

[9] Docker. Docker MCP Catalog. https://hub.docker.com/mcp. accessed March 2026.

[10] JSON-RPC Working Group. JSON-RPC 2.0 Specification. https://www.jsonrpc.org/specification, 2013.

PoS(ISGC2026)003

[11] Jayachandu Bandlamudi et al. A framework for testing and adapting REST APIs as LLM tools, 2025.

[12] Swetha Lakshmana Murthya, Sangamithra Panneer Selvama, and Martin Welß. Multi-agentic workflows and long-term memory use cases in AI-builder, 2025.

[13] ECMWF. Copernicus Climate Data Store. https://cds.climate.copernicus.eu/. accessed March 2026.

[14] European Commission. Destination Earth. https://digital-strategy.ec.europa.eu/en/policies/destination-earth. accessed March 2026.

[15] T. Schick et al. Toolformer: Language models can teach themselves to use tools. In *Proc. NeurIPS 2023*, 2023.

[16] LangChain. LangChain: Building applications with LLMs through composability. https://www.langchain.com/. accessed March 2026.

[17] LlamaIndex. LlamaIndex: Data framework for LLM applications. https://www.llamaindex.ai/. accessed March 2026.

[18] Open Geospatial Consortium. OGC API Standards. https://ogcapi.ogc.org/. accessed March 2026.

[19] M. Schramm et al. The openEO API – harmonising the use of Earth Observation cloud services using virtual data cube functionalities. *Remote Sensing*, 13(6):1125, 2021.

[20] Model Context Protocol. Python SDK for MCP. https://github.com/modelcontextprotocol/python-sdk. accessed March 2026.

[21] LM Studio. Discover, download, and run local LLMs. https://lmstudio.ai/. accessed March 2026.

[22] Alibaba Cloud. Qwen Technical Report. *arXiv preprint*, 2023. arXiv:2309.16609.

[23] Hongshen Xu et al. Reducing tool hallucination via reliability alignment, 2025.

[24] Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. EASYTOOL: Enhancing LLM-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 951–972, 2025.

[25] Mohammed Mehedi Hasan et al. Model context protocol (MCP) tool descriptions are smelly! Towards improving AI agent efficiency with augmented MCP tool descriptions, 2026.

[26] Amazon Web Services. GENCOST05-BP01: Create stopping conditions to control long-running agentic workflows. https://docs.aws.amazon.com/wellarchitected/latest/generative-ai-lens/gencost05-bp01.html, 2025.

PoS(ISGC2026)003