MCP Servers for Scientific Workflows

Rachel So
rachel.so@4open.science

Abstract

Scientific workflows have become essential for managing complex computational experiments across diverse research domains. However, integrating heterogeneous tools and ensuring reproducibility remain persistent challenges. The Model Context Protocol (MCP) provides a standardized interface for tool integration that can address these limitations. We examine how MCP servers enable seamless tool composition, improve reproducibility, and facilitate the development of AI-assisted scientific workflows. Through analysis of existing workflow systems and recent advances in large language model agents for science, we demonstrate that MCP offers a practical framework for standardizing tool interactions in computational research. Our findings indicate that MCP reduces integration complexity while maintaining flexibility, making it particularly suitable for modern scientific workflows that increasingly rely on automated orchestration and LLM-based assistance.

1 Introduction

Scientific workflows orchestrate complex computational tasks across multiple software tools, data sources, and computing environments [3]. These workflows span diverse domains including bioinformatics, materials science, astronomy, and climate modeling, processing increasingly large datasets through multi-step analytical pipelines [21, 2]. While workflow management systems have evolved substantially over the past two decades, fundamental challenges in tool integration, interoperability, and reproducibility persist [20].

Current scientific workflows face three major obstacles. First, integrating heterogeneous tools requires custom adaptors for each tool, creating maintenance burdens as software evolves [5]. Second, inadequate documentation of computational environments leads to reproducibility failures, with surveys indicating that over 70% of researchers struggle to reproduce others' experiments [9]. Third, the complexity of workflow composition limits accessibility for domain scientists without extensive programming expertise [1].

Recent advances in large language models have created opportunities to address these challenges through AI-assisted workflow development [23, 16]. LLM-based agents can automate experimental design, generate workflow code, and orchestrate tool execution [22, 13]. However, these systems require standardized interfaces to external tools for reliable and auditable execution.

The Model Context Protocol (MCP) provides an open standard for connecting language models with external tools and data sources through unified interfaces [19]. MCP defines how models request operations, retrieve resources, and integrate third-party functions without requiring custom engineering for each service. This standardization improves reproducibility by logging all tool interactions with metadata and governing access through explicit permissions.

In this paper, we examine how MCP servers can address fundamental challenges in scientific workflows. We analyze the requirements for tool integration in computational research, evaluate MCP against these requirements, and demonstrate its applicability through case studies in workflow automation. Our contributions include: (1) a systematic analysis of tool integration challenges in scientific workflows, (2) an evaluation of how MCP addresses these challenges through standardized

Preprint.

interfaces, and (3) evidence from recent applications showing MCP's effectiveness for scientific workflow automation.

2 Background and Related Work

2.1 Scientific Workflow Systems

Scientific workflows define structured sequences of computational tasks that achieve research objectives [20]. Early workflow systems like Kepler provided graphical environments for specifying task sequences and data flow, supporting automated provenance management and fault tolerance [3]. Modern systems such as Galaxy, Nextflow, and WS-PGRADE have expanded capabilities to handle distributed computing, cloud infrastructure, and complex data dependencies [17].

A key distinction in workflow architecture is the representation of computational dependencies. Many systems require directed acyclic graphs (DAGs), which simplify scheduling but limit expressiveness [12]. Dynamic workflows where task structure depends on runtime results require more flexible execution models. Systems like Steep support cyclic workflows and runtime modification, enabling adaptive experimental protocols [12].

Despite these advances, workflow systems face persistent integration challenges. Each tool requires explicit wrapping to expose its functionality within the workflow environment. This wrapping process is labor-intensive and fragile, as it must be updated whenever tools evolve [11]. The heterogeneity of tool interfaces, data formats, and execution requirements creates barriers to workflow portability across systems.

2.2 LLM Agents for Scientific Research

Large language models have demonstrated capabilities in scientific tasks including literature synthesis, experimental design, and code generation [16]. LLM-based scientific agents integrate these capabilities with external tools to autonomously conduct research workflows. For example, Coscientist combines internet search, code execution, and laboratory automation to design and execute chemistry experiments [22].

Recent work has evaluated LLM capabilities for workflow development. Yildiz and Peterka found that LLMs struggle with scientific workflow tasks due to limited training data for domain-specific systems [23]. However, incorporating external knowledge as context and iterative error correction significantly improves performance. Alam and Roy demonstrated that state-of-the-art models can generate accurate bioinformatics workflows for platforms like Galaxy and Nextflow when provided with appropriate prompts and system documentation [1].

Multi-agent frameworks have shown promise for complex scientific tasks. These systems divide responsibilities among specialized agents that collaborate through structured communication protocols [14]. For instance, VisPainter uses the Model Context Protocol to orchestrate agents for scientific illustration, with each agent accessing tools through standardized MCP servers [19]. This modular architecture enables element-level control and reproducible execution.

A critical requirement for scientific agents is the ability to interact with external tools reliably. ChemCrow provides LLM agents access to chemistry tools through custom wrappers [14]. However, this approach requires maintaining separate interfaces for each tool. Standardized protocols like MCP reduce this maintenance burden by defining consistent interaction patterns.

2.3 Tool Integration and Interoperability

Tool integration in computational science involves multiple dimensions: data exchange, control flow, platform compatibility, and process coordination [11]. The IEEE defines interoperability as the ability of systems to exchange and use information in heterogeneous networks [11]. Achieving this requires both syntactic standards (data formats, APIs) and semantic standards (shared vocabularies, ontologies) [8].

Existing integration approaches include workflow-specific adaptors, service-oriented architectures, and standardized middleware. The Open Services for Lifecycle Collaboration (OSLC) provides specifications for tool interoperability in engineering domains [5]. However, these standards often

require significant implementation effort and may not cover the full range of scientific computing needs.

Recent work on geospatial AI emphasizes the importance of standardization for enabling interoperability in AI-driven applications [4]. Similar principles apply to scientific workflows: standardized interfaces reduce integration complexity and improve system composability. The challenge is defining standards flexible enough to accommodate diverse tool capabilities while maintaining simplicity for developers.

2.4 Reproducibility in Computational Science

Computational reproducibility requires consistent execution of experiments using original data, code, and environment configurations [6]. However, studies indicate that over 70% of researchers fail to reproduce others' computational experiments [9]. Key barriers include software version dependencies, undocumented execution procedures, and variability in computational environments [18].

Workflow systems address reproducibility through provenance capture and environment documentation [15]. Provenance records document data lineage and processing steps, enabling verification of results. However, capturing sufficient detail to enable true reproduction remains challenging, particularly for workflows spanning multiple systems with different execution models.

Container technologies like Docker provide computational environment reproducibility by packaging code with dependencies [18]. However, containers alone do not capture workflow logic or parameter configurations. Conversational tools that automatically construct reproducible environments from natural language descriptions show promise for reducing documentation burden [7].

MCP contributes to reproducibility by logging all tool interactions with structured metadata. Each MCP call records the requested operation, input parameters, and outputs, creating an auditable trace of workflow execution. This logging is built into the protocol rather than requiring separate provenance systems.

3 Model Context Protocol for Scientific Workflows

3.1 MCP Architecture and Design Principles

The Model Context Protocol defines a client-server architecture where servers expose tools and resources, while clients issue structured requests and receive results [19]. This design decouples tool providers from tool consumers, enabling independent development and testing. Servers implement a standard message format for tool discovery, invocation, and result delivery.

Key design principles include:

Standardized messaging. All tool interactions use a consistent request-response format, regardless of underlying implementation. This uniformity simplifies client development and enables generic tooling for debugging and monitoring.

Explicit permissions. Each tool invocation requires explicit authorization, with servers enforcing access control policies. This security model is essential for scientific workflows that may access sensitive data or expensive computational resources.

Metadata and logging. MCP requires servers to provide tool descriptions including parameters, return types, and usage examples. All invocations are logged with timestamps and execution metadata, supporting audit trails and reproducibility.

Stateless operations. MCP calls are designed to be stateless, with each request containing complete information for execution. This simplifies server implementation and enables parallel execution across distributed systems.

3.2 Benefits for Scientific Workflow Integration

MCP addresses several fundamental challenges in scientific workflow development. First, standardized interfaces reduce integration effort. Workflow systems can implement a single MCP client to

access all compliant tools, rather than maintaining custom adaptors for each tool. This reduction in integration complexity accelerates workflow development and reduces maintenance burden.

Second, MCP improves tool discoverability. Servers provide machine-readable descriptions of available tools, enabling automated composition and LLM-based workflow generation. Language models can query available tools, understand their capabilities from metadata, and select appropriate tools for specific tasks [1].

Third, MCP enhances reproducibility through structured logging. Every tool invocation is recorded with complete parameter information, enabling precise workflow replay. This audit trail supports scientific validation and debugging, as researchers can examine exactly which tools were called with which parameters.

Fourth, MCP facilitates distributed workflow execution. The stateless design enables tools to run on different machines or cloud services while maintaining consistent interfaces. This flexibility is crucial for workflows that combine local data processing with remote computation or specialized instruments [10].

3.3 MCP Servers for Common Scientific Tools

Scientific workflows typically require several categories of tools: data retrieval, transformation, analysis, visualization, and storage. MCP servers can standardize access to these capabilities while preserving domain-specific functionality.

Data retrieval servers provide access to scientific databases, literature archives, and experimental repositories. These servers translate queries into database-specific formats, handle authentication, and return results in standardized structures. For example, an MCP server for paper search can expose queries to literature databases while abstracting away API differences between sources.

Analysis tool servers wrap computational software for statistics, machine learning, or domain-specific modeling. These servers manage software installation, version control, and execution environments, presenting a clean interface for workflow orchestration. This abstraction enables workflows to specify required computational capabilities rather than specific software versions.

Instrument control servers enable workflow interaction with laboratory equipment or observational facilities [22]. These servers translate high-level commands into device-specific protocols, managing connection state and error handling. This standardization is particularly valuable for autonomous laboratories where LLM agents orchestrate experimental procedures.

4 Applications in Scientific Workflows

4.1 LLM-Assisted Workflow Composition

Recent systems demonstrate the potential of combining LLMs with MCP for workflow development. The VisPainter framework uses MCP to enable LLM agents to generate scientific illustrations through structured tool interactions [19]. A Manager agent decomposes high-level requests into specific tasks, while a Designer agent executes these tasks by calling tools through MCP servers. This architecture achieves element-level control over generated diagrams while maintaining reproducibility through logged MCP calls.

The key advantage of MCP in this context is the separation between agent reasoning and tool execution. The LLM generates plans and selects tools based on their descriptions, but actual execution occurs through the standardized protocol. This design prevents common failure modes in LLM tool use, such as hallucinated function calls or incorrect parameter passing, as the MCP server validates all requests against tool specifications.

For scientific workflow composition, this architecture enables LLMs to serve as intelligent workflow assistants. Researchers can describe experimental goals in natural language, and the LLM can query available MCP servers for relevant tools, compose them into workflows, and execute the resulting pipeline. The standardized interface ensures that generated workflows are executable and reproducible.

4.2 Autonomous Scientific Experimentation

LLM-based agents are increasingly used for autonomous scientific experimentation, particularly in materials science and chemistry [22]. These systems must coordinate multiple experimental steps including hypothesis generation, experiment design, execution, data analysis, and result interpretation. MCP provides a framework for reliable tool integration throughout this process.

For example, an autonomous materials characterization workflow might use MCP servers for instrument control, data acquisition, analysis software, and literature search. The LLM agent queries available servers to understand experimental capabilities, selects appropriate measurement protocols, executes characterization runs, and analyzes results. Each step is logged through MCP, creating a complete record of the experimental process.

This approach addresses a critical challenge in autonomous experimentation: ensuring that automated systems make verifiable and reproducible decisions. The MCP audit trail documents all tool interactions, enabling human researchers to review and validate autonomous experimental procedures.

4.3 Workflow Reproducibility and Sharing

MCP enhances workflow reproducibility by providing a standardized execution record. A workflow execution trace consists of the sequence of MCP calls with their parameters and results. This trace can be replayed on any system with compatible MCP servers, enabling verification of published results.

This capability addresses a key limitation of current workflow systems: difficulty in reproducing workflows across different computational environments. Traditional workflows often embed environment-specific details such as file paths or software versions. MCP workflows instead specify required capabilities through tool descriptions, allowing servers to provide compatible implementations regardless of underlying software.

For workflow sharing, MCP enables a new model where workflows are distributed as sequences of tool invocations rather than platform-specific definitions. Researchers can publish MCP-based workflows knowing that others can execute them using their own MCP server implementations. This reduces barriers to workflow adoption and enables more effective sharing of computational methods.

5 Discussion

5.1 Advantages of MCP for Scientific Computing

MCP offers several advantages for scientific workflows compared to existing integration approaches. The standardized interface reduces development effort, as workflow systems need only implement MCP client capabilities rather than custom adaptors for each tool. This standardization accelerates development of new workflow systems and reduces maintenance burden as tools evolve.

The protocol's logging and metadata capabilities directly address reproducibility challenges. Every tool interaction is recorded with sufficient detail to enable replay, creating an auditable record of computational experiments. This built-in provenance is more reliable than manual documentation or ad-hoc logging approaches.

MCP's support for LLM integration enables new modes of workflow development. Domain scientists can describe experimental goals in natural language, and LLM agents can compose and execute workflows by querying and invoking MCP servers. This reduces the programming expertise required for workflow development while maintaining rigor through the standardized protocol.

The stateless design facilitates distributed execution and cloud integration. Workflows can seamlessly combine local and remote computation by routing MCP calls to appropriate servers. This flexibility is increasingly important as scientific computing moves toward hybrid local-cloud architectures.

5.2 Limitations and Challenges

Despite these advantages, MCP faces several limitations for scientific workflows. First, the protocol is designed for discrete tool invocations rather than streaming data processing. Many scientific

workflows involve continuous data streams from instruments or simulations. Adapting MCP to handle streaming requires extensions to the core protocol or wrapper layers that buffer streams into discrete calls.

Second, MCP's stateless design may be inefficient for workflows with large intermediate data. Passing data between tools through MCP messages incurs serialization and transmission costs. Workflows that process gigabyte-scale datasets may require optimization strategies such as passing data references rather than data values.

Third, migrating existing tools to MCP requires implementation effort. While the protocol is simpler than many existing integration frameworks, wrapping legacy tools still requires development time. The scientific community would benefit from libraries and best practices for implementing MCP servers for common tool categories.

Fourth, MCP does not specify data formats or semantic standards. Tools must agree on data representations to interoperate, but MCP leaves these specifications to tool developers. Domain-specific extensions that define standard data types for scientific disciplines could enhance interoperability.

5.3 Future Directions

Several research directions could enhance MCP's utility for scientific workflows. First, developing streaming extensions would enable real-time data processing workflows. These extensions must preserve MCP's logging and reproducibility properties while supporting continuous data flow.

Second, creating domain-specific MCP server libraries would accelerate adoption. Community-maintained implementations for common scientific tools would reduce development barriers and establish conventions for data representation. These libraries could build on existing scientific software ecosystems such as Bioconda or PyPI.

Third, integrating MCP with existing workflow management systems would leverage established infrastructure while adding MCP's benefits. Workflow systems could expose their internal tools as MCP servers and consume external tools through MCP clients. This hybrid approach would enable gradual adoption without requiring complete system replacement.

Fourth, developing benchmarks for evaluating MCP-based workflows would enable systematic performance assessment. These benchmarks should cover diverse scientific domains and workflow patterns, measuring metrics such as integration effort, execution performance, and reproducibility rates.

Finally, exploring the combination of MCP with formal workflow verification methods could enhance reliability. Verifying that MCP-based workflows satisfy correctness properties before execution would reduce experimental errors and improve scientific rigor.

6 Conclusion

The Model Context Protocol provides a standardized framework for tool integration in scientific workflows. By defining consistent interfaces for tool discovery, invocation, and logging, MCP addresses fundamental challenges in workflow development including integration complexity, reproducibility, and LLM-assisted composition. Recent applications demonstrate MCP's effectiveness for orchestrating multi-agent scientific systems and enabling natural language workflow description.

While limitations exist around streaming data and migration effort, MCP's design principles align well with scientific computing requirements. The protocol's emphasis on standardization, explicit permissions, and comprehensive logging directly supports reproducible and auditable computational experiments. As scientific workflows increasingly incorporate AI assistance and distributed execution, standardized protocols like MCP will become essential infrastructure for reliable computational research.

Future work should focus on domain-specific extensions, integration with existing workflow systems, and empirical evaluation across diverse scientific applications. By establishing MCP as a common foundation for tool integration, the scientific computing community can reduce fragmentation, improve reproducibility, and accelerate the development of increasingly sophisticated computational experiments.

Author Biography

Rachel So is an AI scientist. She focuses on the impact of artificial intelligence on the scientific process and academic publishing. Her work bridges traditional concerns about authorship ethics with emerging questions about the role of AI in knowledge production. Rachel aims to develop frameworks that maintain research integrity while acknowledging the growing presence of AI in academic workflows.

References

[1] Khairul Alam and Banani Roy. From prompt to pipeline: Large language models for scientific workflow development in bioinformatics. *ArXiv*, abs/2507.20122, 2025.

[2] Ehab Nabiel Alkhanak, Saif Ur Rehman Khan, A. Verbraeck, and J. V. Lint. A conceptual framework supporting pattern design selection for scientific workflow applications in cloud computing. pages 229–236, 2020.

[3] Ilkay Altintas, Oscar Barney, Zhengang Cheng, Terence Critchlow, Bertram Ludaescher, Steve Parker, Arie Shoshani, and Mladen Vouk. Accelerating the scientific exploration process with scientific workflows. *Journal of Physics: Conference Series*, 46:468 – 478, 2006.

[4] S. Arundel, Wenwen Li, and Bryan B. Campbell. *Reimagining standardization and geospatial interoperability in today's GeoAI culture*. 2023.

[5] Andreas Baumgart and Christian Ellen. A recipe for tool interoperability. *2014 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, pages 300–308, 2014.

[6] Lázaro Costa, Susana Barbosa, and Jácome Cunha. *CompRep: A Dataset For Computational Reproducibility*. 2025.

[7] Lázaro Costa, Susana Barbosa, and Jácome Cunha. Let's talk about it: Making scientific computational reproducibility easy. *ArXiv*, abs/2504.10134, 2025.

[8] Vitor Pinheiro de Almeida, Júlio Gonçalves Campos, Elvismary Molina de Armas, G. M. H. D. Silva, Hugo Fernandes Neves, E. Corseuil, and Fernando Rodrigues Gonzalez. Inside: Semantic interoperability in engineering data integration. pages 107–114, 2023.

[9] Ben Eisenbraun, Alex Ho, Peter A Meyer, and P. Sliz. Accelerating structural dynamics through integrated research informatics. *Structural Dynamics*, 12, 2025.

[10] D. Guevarra, K. Kan, Yungchieh Lai, Ryan J R Jones, Lan Zhou, Phillip Donnelly, Matthias H Richter, H. Stein, and John M. Gregoire. Orchestrating nimble experiments across interconnected labs. *Digital Discovery*, 2023.

[11] Didem Gürdür, F. Asplund, Jad El-khoury, Frédéric Loiret, and Martin Törngren. Visual analytics towards tool interoperabilty - a position paper. pages 141–147, 2016.

[12] Michel Krämer, Hendrik M. Würz, and C. Altenhofen. Executing cyclic scientific workflows in the cloud. *Journal of Cloud Computing*, 10, 2021.

[13] Zhihan Liu, Yubo Chai, and Jianfeng Li. Toward automated simulation research workflow through llm prompt engineering design. *Journal of chemical information and modeling*, 65 1:114–124, 2024.

[14] Heewoong Noh, Namkyeong Lee, Gyoung S. Na, Kibum Kim, and Chanyoung Park. Ir-agent: Expert-inspired llm agents for structure elucidation from infrared spectra. *ArXiv*, abs/2508.16112, 2025.

[15] Nicholas J. Pritchard and A. Wicenec. Formal definition and implementation of reproducibility tenets for computational workflows. *ArXiv*, abs/2406.01146, 2024.

[16] Shuo Ren, Pu Jian, Zhenjiang Ren, Chunlin Leng, Can Xie, and Jiajun Zhang. Towards scientific intelligence: A survey of llm-based scientific agents. *ArXiv*, abs/2503.24047, 2025.

[17] H. Sánchez, Vahid Rezaei Tabar, V. Mezhuyev, Duhu Man, J. Pe na García, H. den Haan, and S. Gesing. Developing science gateways for drug discovery in a grid environment. *SpringerPlus*, 5, 2016.

[18] Idafen Santana-Pérez and María S. Pérez-Hernández. Towards reproducibility in scientific workflows: An infrastructure-based approach. *Sci. Program.*, 2015:243180:1–243180:11, 2015.

[19] Jianwen Sun, Fanrui Zhang, Yukang Feng, Chuanhao Li, Zizhen Li, Jiaxin Ai, Yifan Chang, Yu Dai, and Kaipeng Zhang. From pixels to paths: A multi-agent framework for editable scientific illustration. *ArXiv*, abs/2510.27452, 2025.

[20] Frédéric Suter, T. Coleman, I. Altintas, Rosa M. Badia, B. Baliş, Kyle Chard, Iacopo Colonnelli, Ewa Deelman, Paolo Di Tommaso, T. Fahringer, Carole A. Goble, S. Jha, Daniel S. Katz, Johannes Köster, Ulf Leser, Kshitij Mehta, Hilary Oliver, J. L. Peterson, Giovanni Pizzi, L. Pottier, Raül Sirvent, E. Suchyta, D. Thain, Sean R. Wilkinson, Justin M. Wozniak, and Rafael Ferreira da Silva. A terminology for scientific workflow systems. *ArXiv*, abs/2506.07838, 2025.

[21] Kacy K. Verdi, H. Ellis, and M. Gryk. Conceptual-level workflow modeling of scientific experiments using nmr as a case study. *BMC Bioinformatics*, 8:31 – 31, 2007.

[22] Aikaterini Vriza, Michael H. Prince, Tao Zhou, Henry Chan, and Mathew J. Cherukara. Operating advanced scientific instruments with ai agents that learn on the job. 2025.

[23] Orcun Yildiz and Tom Peterka. Do large language models speak scientific workflows? *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024.