PROV-AGENT: Unified Provenance for Tracking AI Agent Interactions in Agentic Workflows

Renan Souza*, Amal Gueroudji†, Stephen DeWitt‡, Daniel Rosendo*, Tirthankar Ghosal*,
Robert Ross†, Prasanna Balaprakash§, Rafael Ferreira da Silva*
*National Center for Computational Sciences, Oak Ridge National Lab, Oak Ridge, TN, USA
†Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL, USA
‡Computational Sciences and Engineering Division, Oak Ridge National Lab, Oak Ridge, TN, USA
§Computer Science and Mathematics Division, Oak Ridge National Lab, Oak Ridge, TN, USA

***Abstract*—Foundation models, such as Large Language Models (LLMs), are increasingly used as core components of AI agents in complex, large-scale workflows across federated and heterogeneous environments. In *agentic workflows*, autonomous agents plan tasks, interact with humans and peers, and shape scientific outcomes. This makes transparency, traceability, reproducibility, and reliability essential. However, AI-based agents can hallucinate or reason incorrectly, and their decisions may propagate errors through the workflow, especially when one agent's output feeds into another's input. Therefore, fine-grained provenance is essential to link agent decisions, their end-to-end context, and downstream impacts. While provenance techniques have long supported reproducibility and workflow data understanding, they fail to capture and relate agent-centric metadata (prompts, responses, and decisions) with the rest of the workflow. In this paper, we introduce PROV-AGENT, a provenance model that extends W3C PROV and leverages the Model Context Protocol (MCP) to integrate agent interactions into end-to-end workflow provenance. Our contributions include: (1) a provenance model tailored for agentic workflows, (2) a near real-time, open-source system for capturing agentic provenance, and (3) a cross-facility evaluation spanning edge, cloud, and HPC environments, demonstrating support for critical provenance queries and agent reliability analysis.**

***Index Terms*—Workflows, Agentic Workflows, Provenance, Lineage, Responsible AI, LLM**

I. Introduction

The integration of foundation models, often referred to as Large "X" Models (LxMs), into computational workflows is rapidly advancing across scientific and industrial domains [1]. These models excel in language, vision, time-series, and robotics tasks, driving innovation in genomics, chemistry, and manufacturing. This shift has driven the emergence of *agentic workflows*, where autonomous agents make decisions, plan tasks, and coordinate with humans and other agents. These agents operate in dynamic environments across heterogeneous computing platforms, including edge devices, cloud systems, and high-performance computing (HPC). Unlike traditional workflows with static, deterministic paths [2], agentic workflows are non-deterministic, shaped by near real-time data, adaptive decisions, and evolving interactions [3], [4]. They often display dynamic, cyclic behavior, where agent outputs inform subsequent decisions or feedback loops.

Although Artificial Intelligence (AI) agents offer capabilities for automating complex processes, they introduce new challenges for transparency, reproducibility, and reliability. They may generate hallucinated or incorrect outputs, especially when relying on generative models, which can propagate through the workflow, compounding errors and making it difficult to assess the correctness of the results [5]. The risks are amplified in workflows where agent decisions influence other agents or downstream tasks, potentially affecting scientific conclusions or operational outcomes. Provenance data management has long played a central role in providing for such transparency, reproducibility, and reliability in computational workflows [6]. However, traditional provenance approaches are not designed to capture the intrinsic dynamics of modern AI agents. Provenance data must not only capture the data flow and task execution history but also represent the reasoning processes, model invocations, and contextual information that drive agent decisions. This level of detail enables rigorous root cause analysis when unexpected or erroneous behavior occurs. For example, understanding how a surprising result was produced requires tracing back through multiple agent interactions, prompts, responses, and intermediate computations.

A unified provenance graph that considers AI agent actions as first-class components, on par with traditional workflow tasks, enables comprehensive traceability and analysis. This structure supports critical queries such as: (1) *What specific input data led an agent to make a particular decision?* (2) *How did an agent's decision influence the control or data flow within the workflow?* (3) *Which downstream outputs were affected by a specific agent interaction?* (4) *Where did erroneous data originate, and through which agents decisions or workflow tasks did it propagate?* These questions are essential for interpreting results, debugging workflows, and improving agent performance through better prompts and model tuning.

In this paper, we build on foundational efforts in workflow provenance research to introduce a framework that captures both traditional workflow metadata and AI agent interac-

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid up, irrevocable, worldwide license to publish or reproduce the published form of the manuscript, or allow others to do so, for U.S. Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

tions. Our contributions are threefold: (1) **PROV-AGENT**, a provenance model that extends the W3C PROV [7] standard and incorporates concepts from the Model Context Protocol (MCP) [8] to represent agent actions and their connections to data and workflow tasks; (2) an open-source system [9] for capturing agentic provenance during execution; and (3) a preliminary evaluation with a cross-facility agentic workflow involving edge devices, cloud services, and HPC systems.

II. Background and Related Work

A. Provenance for Tracking AI Agents in Dynamic Cross-Facility Workflows

Agentic workflows are emerging as a new paradigm in scientific computing, where autonomous AI agents are integrated into complex, multi-step processes. These agents, often powered by foundation models such as LLMs, take on responsibilities traditionally handled by humans or static scripts. They interpret data, make decisions, and adaptively steer workflow execution. To support the development and orchestration of such agentic workflows, a variety of frameworks have emerged. For instance, LangChain [10], [11], AutoGen [12], LangGraph [13], Academy [3], and CrewAI [14] support multi-agent systems that interact through prompt exchanges, calls to foundation models typically hosted by AI service providers in the cloud (e.g., OpenAI, SambaNova), and shared context. These frameworks support MCP [8], which is emerging as a standard in academia and industry. MCP defines core agentic AI development concepts, including tools, prompts, resources, context management, and agent–client architecture that can communicate with external sources, such as knowledge bases or web pages, for Retrieval-Augmented Generation (RAG) [15] to dynamically augment prompts.

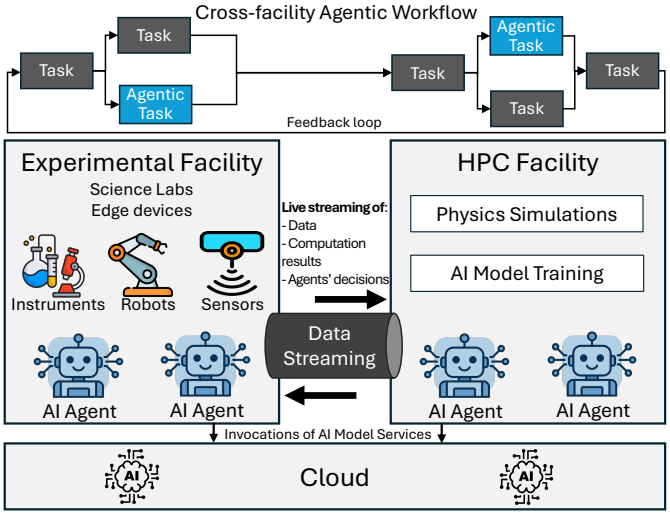

Fig. 1: Agentic workflow spanning an edge-cloud-HPC continuum. Data stream in near real time from the experimental facility to HPC systems, with results feeding back into upstream tasks. Agentic tasks (tools) run alongside traditional ones, making provenance critical to trace potential hallucinations or errors that may propagate through the entire workflow.

A growing challenge in these workflows involves managing execution across physically and logically distributed facilities that include edge devices, cloud services, and HPC systems [16]–[19] (Fig. 1). Scientific experiments may be conducted in external laboratories or at the edge, generating data in near real-time. These data must be immediately transmitted to an HPC system, where they feed into simulations, analytics pipelines, or machine learning (ML) training processes. This tight integration requires not only reliable data movement across sites but also a coherent understanding of how AI agents interact with this data across systems.

While some MCP-based agent frameworks record prompts, responses, and AI service invocations, these data are typically isolated from the rest of the workflow. This disconnection hinders the contextualization of agent interactions or understanding their downstream impact. Existing provenance techniques lack explicit representations of key agent artifacts and their integration with the workflow. They typically model workflows as static graphs, missing the semantics needed to capture agentic behavior, dynamic decisions, and model-driven reasoning. We argue that *agentic provenance*, i.e., provenance data that track tasks executed by AI agents and their influence on downstream non-agentic tasks and data in the workflows, provides the glue power needed to unify these elements into a single, queryable graph. This enables traceability, root cause analysis, and continuous agent improvement, such as refining prompts or tuning model parameters to reduce hallucinations, which are essential in agentic workflows to support responsible, reproducible, and trustworthy AI-driven decisions [6].

B. W3C PROV and Extensions for Workflows, AI, and Agents

The W3C PROV standard [7] is a widely adopted representation model for provenance, providing a structured way to describe how data were produced, by whom, and through which processes. Fortunately, the W3C PROV standard already defines **Agent**, the central abstraction in this work, as one of its three core classes, alongside Entity (data) and Activity (process), with agents representing either software or human actors responsible for activities. Fig. 2 shows these core classes and their main relationships among them.

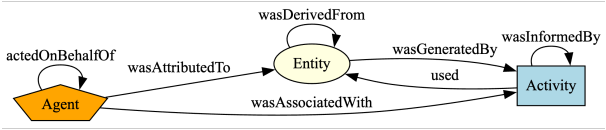

Fig. 2: The W3C PROV Provenance Model [7].

PROV supports domain- and application-specific extensions and underpins many workflow provenance systems requiring standardized, interoperable representations of complex processes. For instance, PROV-DfA [20] extends PROV to capture human actions in human-steered workflows, while ProvONE [21] adds workflow-specific metadata and aims at supporting existing workflow management systems. For AI/ML workflows, PROV-ML [22] combines general workflow concepts with ML-specific artifacts, especially for model

training and evaluation. FAIR4ML [23] adopts a model-centric approach to support the Findability, Accessibility, Interoperability, and Reproducibility (FAIR) principles. These works are orthogonal to ours, as they define complementary concepts rather than representing AI agents that steer workflows. Although the W3C PROV has been extended for agents and multi-agent systems [24], [25], these earlier efforts predate agentic workflows, lacking support for core agentic AI concepts [26] and how they relate to broader workflow.

III. A Provenance Model for Agentic Workflows

PROV-AGENT is a provenance model for representing AI agent interactions, model invocations, and their relationships to non-agentic tasks and data in agentic workflows (Fig. 3). It extends W3C PROV and incorporates MCP concepts to unify agents and traditional components as first-class elements.

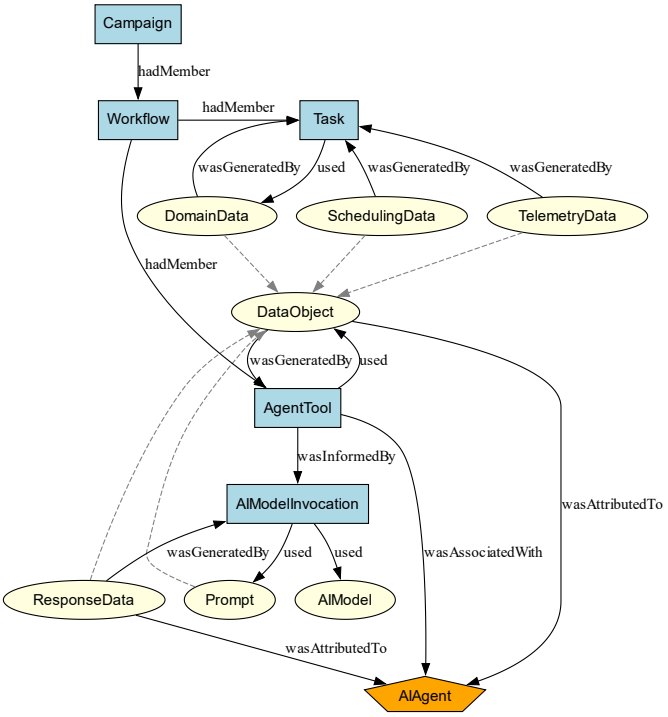

Fig. 3: PROV-AGENT: A W3C PROV Extension for Agentic Workflows. Dashed arrows represent *subClassOf*.

At its core, the model includes standard workflow structures such as `Campaign`, `Workflow`, and *Task*, modeled as subclasses of PROV Activities. Campaigns are associated with Person or Organization agents via *wasAssociatedWith*, omitted from the figure to reduce clutter. Tasks consume (PROV *used*) and produce (PROV *generated*) domain-specific data objects (`DomainData`). Typically, in a provenance graph, they contain parameters, arguments, KPIs, QoIs, and pointers to domain data files or data objects stored elsewhere [27]. Tasks also generate two additional types of metadata: `SchedulingData` and `TelemetryData`. `SchedulingData` contains where the task ran, including details such as compute node, CPU core, or GPU ID. `TelemetryData` contains runtime metrics such as CPU and GPU usage, and disk usage. These data, modeled as subclasses of `DataObject`, which is a subclass of PROV Entity, enrich the provenance graph with infrastructure-level context needed for traceability and performance analysis.

We extend the abstract W3C PROV **Agent** by modeling `AIAgent` as its subclass, enabling a natural integration of agent actions and interactions into the broader workflow provenance graph. This modeling is not constrained to single-agent scenarios. Multi-agentic workflows, with other AI agents, each with their own tools and reasoning paths, can be instantiated within the same provenance graph, enabling the representation of collaborative or parallel agent behaviors within a workflow.

Following the MCP terminology, an AI agent can be associated with one or many tool executions (`AgentTool`) and each tool may be informed by (PROV *wasInformedBy*) one or many `AIModelInvocations`. Each `AIModelInvocation` uses a `Prompt` and a specific `AIModel`, which holds model metadata, including its name, type, provider, temperature, and other parameters, and *generates* a `ResponseData` object, which is *attributedTo* the corresponding agent. Although LLMs are more common, the PROV-AGENT is designed to be modality-agnostic and supports other foundation models, such as those for vision, audio, or multimodal reasoning, as long as they follow a prompt-invocation-response interaction model.

The data used or generated by agents, including prompts, responses, are represented as subclasses of the `DataObject` Entity. This allows agents to consume and produce not only `DomainData`, but also system-level and contextual data such as `SchedulingData` and `TelemetryData`. When instances of subclasses of `DataObject` are generated by agent tools, they are attributed to (*wasAttributedTo*) the instance of the agent. The additional data types can be used by the agent as part of reasoning or planning, for example through RAG strategies to enhance prompts with relevant contextual knowledge. Since the relationships are explicitly modeled using standard PROV constructs such as *used*, *wasGeneratedBy*, *wasAssociatedWith*, and *wasInformedBy*, the resulting graph is fully connected and queryable. This enables users to trace a final output or decision all the way back through agent reasoning, prompts, input data, system context, and execution metadata, addressing the key challenge of capturing agentic behavior as part of end-to-end workflows.

IV. System Implementation and Evaluation

A. Implementation

Rather than building a new provenance system from scratch, we extend *Flowcept* [9], an open-source distributed provenance framework designed for complex, heterogeneous workflows spanning experimental facilities at the edge, cloud platforms, and HPC environments. Flowcept uses a federated, broker-based model where raw provenance data, which may come in varied formats and schemas, can be streamed from instrumented scripts, observability hooks in workflow tools (e.g., Dask, MLflow), and from data streaming services and

storage layers such as Redis, Kafka, SQLite, file systems, and object stores while the workflows run [18]. A central consolidation service unifies, curates this data into a persistent provenance database while applying a W3C PROV-extended model, making it well-suited for capturing and contextualizing provenance in end-to-end agentic workflows.

Building on the MCP concepts, when the MCP server is initialized, we begin by creating a new instance of `AIAgent`, assigning it an identifier and name so it can be properly associated with the tools it executes. Flowcept supports several instrumentation methods, but for MCP-based tools, the most straightforward method is via decorators. In generic Python functions, applying the `@flowcept_task` decorator ensures that, upon execution, the function's inputs, outputs, and any generated telemetry or scheduling data are automatically captured. Follow this approach, since MCP tools have well-defined input arguments and return values, we introduce a new decorator, `@flowcept_agent_tool`, which creates a corresponding `AgentTool` execution activity for each tool execution. This activity is associated with the executing agent and linked to its inputs and outputs using the PROV relationships defined in the PROV-AGENT model.

Tool executions are often informed by one or more `AIModel` calls. Given our driving use cases and that most agentic workflow users employ LLMs for their agents, this first implementation of PROV-AGENT focuses on supporting LLMs by providing a generic wrapper for abstract LLM objects, compatible with models from popular LLM interfaces, including CrewAI, LangChain, and OpenAI. Whenever a prompt is sent to an LLM service provider in the cloud (e.g., OpenAI, SambaNova, Azure), the wrapper captures the prompt, response, model metadata (e.g., provider, name, and parameters like temperature), and optional telemetry such as response time. Fig. 4 shows an MCP tool example annotated with the decorator and using the wrapper `FowceptLLM`. Model metadata are recorded within an instance of `AIModel` and each invocation is recorded as an `AIModelInvocation` activity and linked to the model, prompt, and response, according to the defined relationships. When a tool depends on LLM results, Flowcept establishes a *wasInformedBy* relationship from the `AgentTool` to the relevant `AIModelInvocation` activities. While this implementation records only the agent's ID and name, the model supports extended metadata, such as model and tools' version control state, and further configuration parameters.

Flowcept also provides an MCP agent with a Streamlit GUI that enables users to interact with the provenance database through natural language queries at runtime. While the details of this agent are beyond the scope of this paper, in the next section we highlight how it helps users to query and explore the provenance data captured using PROV-AGENT.

B. Preliminary Evaluation

In this section, we evaluate PROV-AGENT and its implementation by demonstrating how agent decisions, LLM interactions, and workflow tasks are unified in a single provenance graph, enabling users to trace erroneous outputs back to their upstream prompts, inputs, and prior decisions.

```python
from langchain_openai import ChatOpenAI
from flowcept import FlowceptLLM, flowcept_agent_tool

@mcp.tool()
@flowcept_agent_tool
def evaluate_scores(layer, result, scores):
    ...
    prompt = get_prompt(layer, result, scores)
    llm = FlowceptLLM(ChatOpenAI(model="gpt-4o"))
    response = llm.invoke(prompt)
    ...
    return ...
```

Fig. 4: MCP agent tool that invokes an LLM to assess physics model outputs. With the decorator `@agent_flowcept_task` and `FlowceptLLM` wrapper, agent tool and LLM invocation provenance are captured.

Use case. We employ PROV-AGENT in an autonomous additive manufacturing workflow being developed at Oak Ridge National Laboratory (ORNL) [28]. This envisioned workflow integrates a metal 3D printer at ORNL's Manufacturing Demonstration Facility (MDF) on the Edge with an HPC system at the ORNL Leadership Computing Facility (OLCF), streaming sensor data in near real-time to HPC simulations, illustrating a concrete case of the generic workflow in Fig. 1. Although the direct live data connection between the sensors and simulation is still under development, our implementation already applies to the agentic control loop and distributed facilities at ORNL. At MDF, sensor drivers collect data layer by layer as a metal component is fabricated. This layer-specific data are used to estimate the current state of the system. Using the approach of model predictive control [29], a forward-looking physics-based model explores the downstream consequences of decisions for upcoming layers. Each prospective decision for the upcoming layer is scored using an analysis routine. Researchers are investigating the benefits of using AI-driven decision-making via Analysis Agent tools invoking an LLM (*gpt-4o*) service hosted in the cloud. The agents use structured prompts to decide which control result is best for print control based on their scores and other data in the agent context, such as previous decisions and user guidance. Thus, the decision made for each layer informs the decision logic in the next, enabling the system to learn over the course of a print. However, because the agent relies on an LLM, there is a risk of hallucinated or incorrect outputs. Since each decision influences the next in this iterative loop, a single error may propagate across layers, potentially compromising downstream outputs, thus making provenance tracking essential.

End-to-end Provenance Graph. Figure 5-A shows how PROV-AGENT would function in the additive manufacturing use case. After the scientist inputs the experiment setup, the driver (`Sensor_Driver_i`) iteratively triggers the sensors for each printed layer i. The resulting `Sensor_Data_i` is streamed to the HPC system for processing by a physics-based model (`Physics_Model_i`) and evaluation

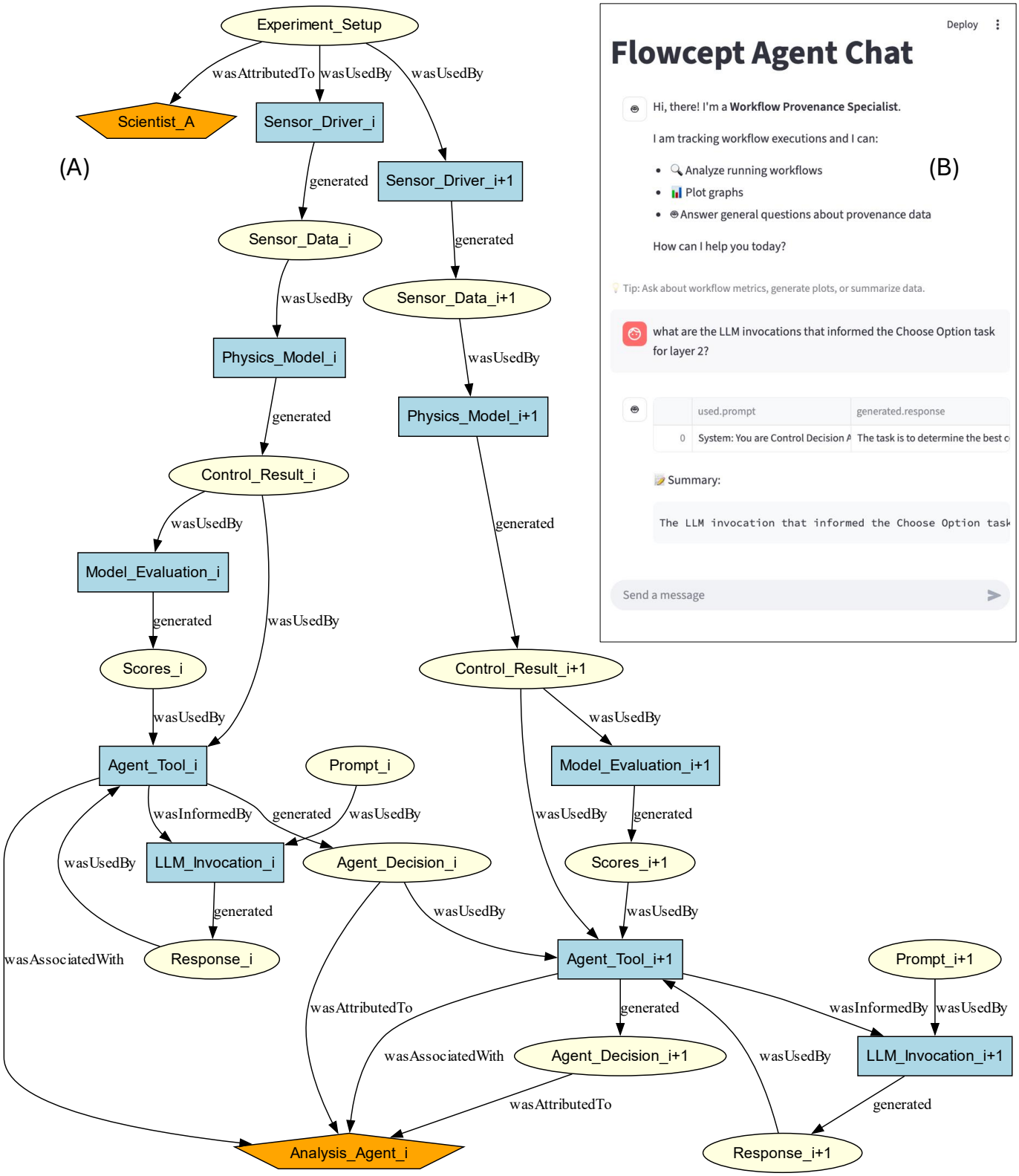

Fig. 5: (A) Instantiation of the unified provenance graph using PROV-AGENT for an additive manufacturing workflow. Sensor drivers run on edge devices, while agents and physics models run on HPC. Sensor data are generated layer by layer and used by the simulation model to assess print quality. An AI agent analyzes results and makes layer-specific decisions, where decisions at iteration i influence iteration $i{+}1$. Only two iterations are shown, though typical workflows span up to thousands. Arrows reverse standard W3C PROV directions (*used*, *wasGeneratedBy*) for top-down clarity.
(B) Chat window showing a natural language query to the Flowcept Agent for Query 3.

task (Model_Evaluation_i), producing control results and scores. Experiment_Setup, Sensor_Data_i, Control_Result_i, and Scores_i are modeled as DomainData, and their linked activities are Tasks.

For every layer i, an agent decision-making tool is executed to assess the scores for physics model predictions, creating an instance (Agent_Tool_i) of the class AgentTool, and is linked to an instance of the AIAgent, the Analysis_Agent_i. Every tool execution for layer i uses the outputs of the physics model and their evaluation scores, and invokes the cloud-based LLM models via LLM_Invocation_i, which are instances of AIModelInvocation. LLM invocations are explicitly connected to the used Prompt_i and generated Response_i instances, where the responses are attributed to the agent instance. The resulting agent decisions (Agent_Decision_i), which are instances of DomainData, are also attributed to the agent, completing each agentic reasoning cycle.

Several model components from the PROV-AGENT schema are intentionally omitted from the figure, including Campaign, Workflow, TelemetryData, and SchedulingData. These classes are recorded in the underlying provenance database but excluded from the visual representation to reduce clutter. Activities, e.g., workflow tasks and agent tools, are linked to location metadata indicating where they ran (e.g., edge, HPC, or cloud). Though omitted in the figure, these PROV Location entities help map execution across the Edge–Cloud–HPC continuum.

Query examples enabled by PROV-AGENT. With PROV-AGENT, several new queries are enabled to support agent accountability and tracing back when errors/hallucinations happen. Below we show a few examples of queries using a distributed Edge-Cloud-HPC workflow that mimics the agentic additive manufacturing workflow under development.

- **Q1. Given an agent decision, what was the complete lineage until the first input data?**
  Given an agent decision Agent_Decision_i, the query traverses to its generating Agent_Tool_i, then to the inputs it used: Scores_i, Control_Result_i, and Agent_Decision_i-1. These are traced back through Model_Evaluation_i and Physics_Model_i to the original Sensor_Data_i that was generated by the driver for layer i when utilized the recorded Experiment_Setup.
- **Q2. When printing layer 2, what was the agent decision, the available score options, and the reasoning behind the decision?**
  Starting at Agent_Decision_2, we trace back to Agent_Tool_2 and inspect the input Scores_2, Control_Result_2, and the Response_2 from the LLM_Invocation_2 to understand the reasoning context.
- **Q3. What was the LLM prompt and response when a surprising agent decision was identified?**
  Given that a hallucination was identified when the agent was deciding on the scores for layer 2, after identifying the unexpected Agent_Decision_2, the query traces back to Agent_Tool_2 and its linked LLM_Invocation_2 to retrieve the corresponding Prompt_2 and Response_2. This query is illustrated in the Streamlit chat GUI of Flowcept agent in Figure 5-B.
- **Q4. How did an agent decision influence subsequent workflow activities?**
  Given that an agent decision Agent_Decision_i is used by another Agent_Decision_i+1, the query recursively navigates on the used/wasGeneratedBy relationships in the path between the Agent_Decision_i and the Agent_Decision in the last layer.
- **Q5. Where did erroneous data originate, and how did it propagate?**
  After identifying a faulty Agent_Decision_i, the query traces backward through the tool, LLM response, model outputs, and Sensor_Data_i to find the cause, and forward to identify affected downstream results.

These queries demonstrate how PROV-AGENT enables end-to-end analysis of agent behavior within workflows, supporting accountability, debugging, and iterative improvement.

V. Conclusion

As AI agents become core components of workflows, ensuring transparency, accountability, and reproducibility is critical, especially given their non-deterministic behavior and potential to hallucinate and propagate errors across data and tasks in the workflows. PROV-AGENT addresses this need by extending the W3C PROV standard and leveraging the Model Context Protocol (MCP) to capture fine-grained agentic provenance. We build on established provenance foundations to add key AI agent artifacts, including tools, prompts, responses, and model invocations, and integrate them to the non-agentic tasks and data in the workflow. This unified approach not only supports traceability and root cause analysis but also enables continuous agent improvement through the comparison of decisions across prompt engineering refinements or fine-tuning of model parameters. We demonstrated our open-source implementation in an agentic workflow use case running on distributed facilities with feedback loops and AI agents steering execution. To the best of our knowledge, this is the first provenance framework designed to track agent actions and decisions in agentic workflows. While this is an early step, it establishes a foundation that researchers and practitioners can build on, not only to enable root-cause analysis, interpretability, and model and prompt fine-tuning, but also to explore new techniques for detecting and ultimately remediating hallucinations in AI-driven decisions.

Acknowledgments. The authors would like to thank the ORNL team: Miaosen Chai, Timothy Poteet, Phillipe Austria, Marshall McDonnell, Ross Miller, A.J. Ruckman, Tyler Skluzacek, Feiyi Wang, Sarp Oral, Arjun Shankar for their assistance and guidance with the use case development. ChatGPT-4o was used to help polish writing, improve conciseness, and check grammar across the sections of the paper. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES

[1] P. Fettke, H.-G. Fill, and J. Köpke, "LLM, LAM, LxM Agent: From Talking to Acting Machines: Insights from the Perspective of Conceptual Modeling," *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, vol. 20, 2025.

[2] F. Suter, T. Coleman, İ. Altintaş, R. M. Badia, B. Balis, K. Chard, I. Colonnelli, E. Deelman, P. Di Tommaso, T. Fahringer *et al.*, "A terminology for scientific workflow systems," *Future Generation Computer Systems*, p. 107974, 2025.

[3] J. G. Pauloski, Y. Babuji, R. Chard, M. Sakarvadia, K. Chard, and I. Foster, "Empowering Scientific Workflows with Federated Agents," *arXiv preprint arXiv:2505.05428*, 2025.

[4] R. Ferreira da Silva, M. Abolhasani, D. A. Antonopoulos, L. Biven, R. Coffee, I. T. Foster, L. Hamilton, S. Jha, T. Mayer, B. Mintz *et al.*, "A Grassroots Network and Community Roadmap for Interconnected Autonomous Science Laboratories for Accelerated Discovery," *arXiv preprint arXiv:2506.17510*, 2025.

[5] X. Gu, X. Zheng, T. Pang, C. Du, Q. Liu, Y. Wang, J. Jiang, and M. Lin, "Agent smith: a single image can jailbreak one million multimodal llm agents exponentially fast," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[6] R. Souza, S. Caino-Lores, M. Coletti, T. J. Skluzacek, A. Costan, F. Suter, M. Mattoso, and R. F. Da Silva, "Workflow provenance in the computing continuum for responsible, trustworthy, and energy-efficient ai," in *2024 IEEE 20th International Conference on e-Science (e-Science)*, 2024.

[7] P. Groth and L. Moreau. (2013) W3C PROV: an overview of the prov family of documents. [Online]. Available: https://www.w3.org/TR/prov-overview

[8] "Model context protocol," https://modelcontextprotocol.io/introduction, 2025.

[9] "Flowcept code repository," https://github.com/ORNL/flowcept, 2025.

[10] O. Topsakal and T. C. Akinci, "Creating large language model applications utilizing langchain: A primer on developing llm apps fast," in *International Conference on Applied Engineering and Natural Sciences*, vol. 1, no. 1, 2023, pp. 1050–1056.

[11] B. Auffarth, *Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT, and other LLMs*. Packt Publishing Ltd, 2023.

[12] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu *et al.*, "Autogen: Enabling next-gen llm applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.

[13] J. Wang and Z. Duan, "Agent AI with langgraph: A modular framework for enhancing machine translation using large language models," *arXiv preprint arXiv:2412.03801*, 2024.

[14] "CrewAI," https://www.crewai.com/, 2025.

[15] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, 2023.

[16] S. Murugesan, "The rise of agentic AI: implications, concerns, and the path forward," *IEEE Intelligent Systems*, vol. 40, no. 2, pp. 8–14, 2025.

[17] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey," *IEEE Access*, 2025.

[18] R. Souza, T. J. Skluzacek, S. R. Wilkinson, M. Ziatdinov, and R. F. da Silva, "Towards lightweight data integration using multi-workflow provenance and data observability," in *IEEE International Conference on e-Science*, 2023.

[19] R. Ferreira da Silva, D. Bard, K. Chard, d. W. Shaun, I. T. Foster, T. Gibbs, C. Goble, W. Godoy, J. Gustafsson, U.-U. Haus, S. Hudson, S. Jha, L. Los, D. Paine, F. Suter *et al.*, "Workflows community summit 2024: Future trends and challenges in scientific workflows," 2024.

[20] R. Souza and M. Mattoso, "Provenance of dynamic adaptations in user-steered dataflows," in *Provenance and Annotation of Data and Processes (IPAW)*. Cham: Springer International Publishing, 2018, pp. 16–29.

[21] Y. Cao, C. Jones, V. Cuevas, M. Jones, B. Ludäscher, T. M. McPhillips, P. Missier, C. R. Schwalm, P. Slaughter, D. Vieglais, L. Walker, and Y. Wei, "Provone: extending prov to support the dataone scientific community."

[22] R. Souza, L. G. Azevedo, V. Lourenço, E. Soares, R. Thiago, R. Brandão, D. Civitarese, E. Vital Brazil, M. Moreno, P. Valduriez, and M. Mattoso, "Workflow provenance in the lifecycle of scientific machine learning," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 14, p. e6544, 2022.

[23] L.-J. Castro, D. Garijo, D. Rebholz-Schuhmann, D. Solanki, J. T. Ciuciu-Kiss, D. Katz, L. Eklund, and G. Bharathy, "FAIR4ML Metadata Schema," https://w3id.org/fair4ml, 2025.

[24] D. B. Davis, J. Featherston, M. Fukuda, and H. U. Asuncion, "Data provenance for multi-agent models," in *2017 IEEE 13th International Conference on e-Science (e-Science)*. IEEE, 2017, pp. 39–48.

[25] S. Friedman, J. Rye, D. LaVergne, D. Thomsen, M. Allen, and K. Tunis, "Provenance-based interpretation of multi-agent information analysis," *arXiv preprint arXiv:2011.04016*, 2020.

[26] R. Sapkota, K. I. Roumeliotis, and M. Karkee, "AI agents vs. Agentic AI: A conceptual taxonomy, applications and challenges," *arXiv preprint arXiv:2505.10468*, 2025.

[27] R. Souza, L. Azevedo, R. Thiago, E. Soares, M. Nery, M. A. S. Netto, E. V. Brazil, R. Cerqueira, P. Valduriez, and M. Mattoso, "Efficient runtime capture of multiworkflow data using provenance," in *IEEE eScience*, 2019.

[28] https://intersect-architecture.readthedocs.io/en/latest/examples/aam/index.html.

[29] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: an engineering perspective," *The International Journal of Advanced Manufacturing Technology*, vol. 117, no. 5, pp. 1327–1349, 2021. [Online]. Available: https://doi.org/10.1007/s00170-021-07682-3