

# Provenance Information Model of Karma Version 3

Bin Cao, Beth Plale, Girish Subramanian, and Ed  
Robertson  
Department of Computer Science  
Indiana University  
Bloomington, IN 47405  
{bincao, plale, subramag, edrbtsn}@cs.indiana.edu

Yogesh Simmhan  
Microsoft Research  
One Microsoft Way  
Redmond, WA 98052  
yoges@microsoft.com

**Abstract**— Provenance that captures e-Science activity has long term value only if the right amount and kind of information is collected. In this paper, we propose a two-layer model for representing provenance information capable of representing both execution information and higher level process details. The information model forms the basis for efficient relational database storage and query, and sets the stage for investigation of the necessary and sufficient information for long-term preservation.

**Keywords**- *provenance; workflow; information model; metadata; provenance storage*

## I. INTRODUCTION

Workflows are an increasingly accepted approach for constructing computational scientific experiments. The importance of provenance in computational science investigation is becoming clearer to domain researchers as they share results and consider the long term usability of the digital data products generated in their work. Provenance captures a derivation history (trace, lineage) of data products [7], and is essential to the long-term preservation and reuse of the data, and to making determinations of its quality.

The Karma [8] provenance collection and management system captures provenance in user-driven workflow systems through modular instrumentation, which makes it useable in diverse workflow architectures composed of, say, Axis 2 web services, Java classes, and message bus listeners. In this paper, we focus our attention on the representation of provenance data, in particular, focusing on collection of the right kinds of provenance data to promote the long-term preservation of scientific digital data objects. The proposed information model contains two layers: a lower level *execution level* and a higher level *registry level*. The execution level captures method invocations, production and consumption of specific data products, parameters used in a particular invocation of a service, and so on. The registry level captures abstract information about services and data products, and valid invocation relationships between computational entities (i.e., processes, services, tasks, clients).

Our abstract model of provenance has its roots in the three-level ANSI/SPARC external/conceptual/internal hierarchy [2]. Conceptual metadata occurs within entity relation models when occurrences of one entity capture metadata about occurrences of another. At the base level is

the data described by the metadata. An example of this situation occurs in a laboratory setting [5]. The laboratory runs a variety of tests; a particular case would be a hospital laboratory which runs tests on blood, tissue, and other samples. Each type of test has a particular set of properties which apply to all tests runs of that type. These properties include inputs and readings, required steps, and specific operating instructions. Each run of a test derives properties from its test type, for example the need for calibration and a scale for those calibrations; but each run has its own unique calibration settings. Thus we could have an entity *Test Type* that has meta-information about entity *Test Run*.

The remainder of the paper is organized as follows. Section II briefly reviews related work. Section III defines the workflow model from which provenance is collected. Section IV introduces our two-layer provenance information model. Section V discusses the database schema, the Open Provenance Model, and the event model, which can be built on top of the information model. Section VI concludes the paper.

## II. RELATED WORK

Provenance data can be represented and stored using different technologies, including relational databases (e.g., REDUX [1]), semantic web technologies RDF and OWL (e.g., myGrid [10]), and internal private formats (e.g., PASS [3]). Relational databases together with XML views are also used (e.g., VisTrails [6]).

Provenance data can be organized in layers for efficient storage. REDUX [1] stores the provenance data in a multilayer model with increasingly detailed and instantiated workflow information from abstract representations down to fully instantiated and executed workflows. The VisTrails [6] provenance information is organized into three layers: the workflow evolution layer, which captures the relationships among the series of workflows created in an exploratory task; the workflow layer, which consists of specification of individual workflows; and the execution layer, which stores run-time information about the execution of workflow modules. The provenance in Chimera [12] is represented in Virtual Data Language (VDL) which represents data products as abstract typed *datasets* and their materialized *replicas*. Computational process templates, called *transformations*, are scripts in the file system. The parameterized instance of the transformations, called *derivations*, can be connected to form workflows that

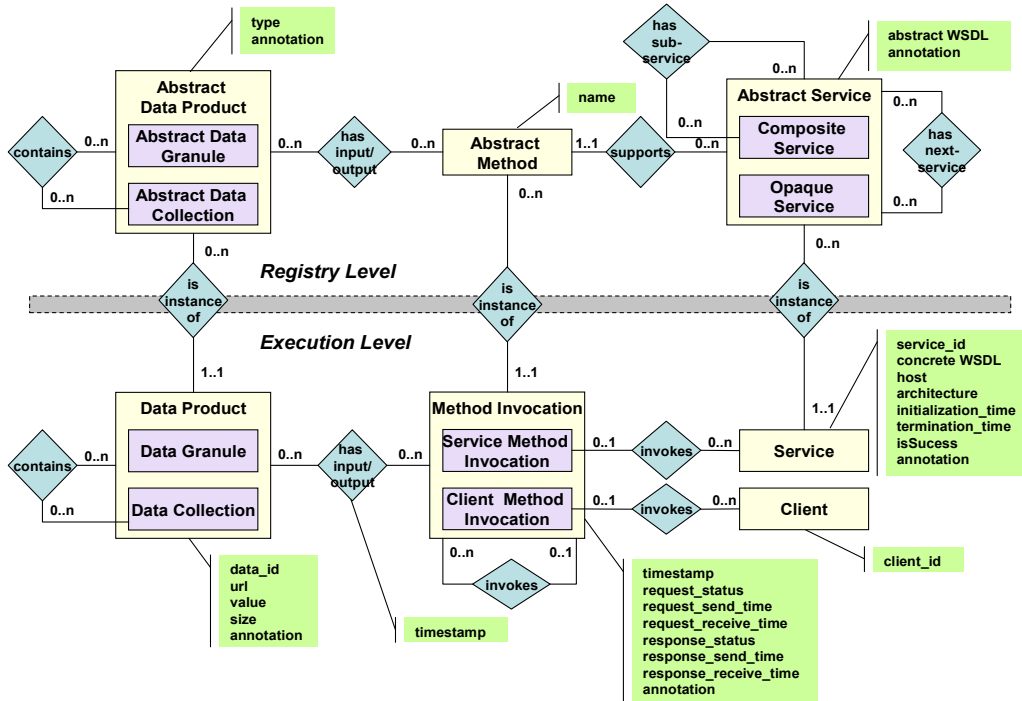


Figure 1. Information Model composed of higher level registry layer and lower level instance or execution layer.

consume and produce replicas. The provenance manager of VIEW [13] contains three layers: the provenance model layer, the relational model layer, and the model mapping layer. The provenance model layer represents scientific workflow run provenance via domain ontologies; the relational model layer stores and queries provenance metadata; and the model mapping layer performs schema mapping, data mapping, and query mapping for the other two layers.

Unlike the above systems, in which workflows have to be predefined before execution, Karma can capture and store streaming provenance information. The two-layer information model we propose contains a registry layer which represents metadata about the instance, i.e., the execution layer. This non-comprehensive discussion of related work covers only the most relevant research.

### III. WORKFLOW MODEL

Our provenance collection system views domain science interaction with cyberinfrastructure as driven either by a user-directed workflow (no orchestration tool in use) or by a workflow orchestration system. For either case, the information model is based on four principle components: *method*, *service*, *workflow*, and *data product*. A method is a function or procedure that executes a computational or data transformation task. A service consists of a group of related methods to perform a coherent task. A data product can be either a single instance of a data object (called a data granule) or a collection of data objects. A workflow can be

regarded as a directed graph where nodes are services and edges between nodes represent data flow between services or methods inside the service. A workflow itself can be considered a client that invokes a lower level or embedded workflow (hierarchical workflow composition).

### IV. INFORMATION MODEL

The provenance collected from the workflow model described in the previous section can be represented by the information model shown in Figure 1. The information model contains two levels: the registry level and the execution level. The registry level has similarities to registries as used in Service-Oriented Architectures (SOA) [11] containing information about services and data products at a sufficient level of detail to support discovery and automated decisions about whether or not to bind a particular data product or service. The execution level captures instance invocation and execution details of a particular sequence of actions. Note how the structure of the top layer mirrors the bottom. The two-layer model recognizes commonalities in workflows and stores that common information consistently and without redundancy.

In more detail, the registry level records the metadata of services and data which may be used in an execution sequence. It also serves to capture the structure of workflows that are known in advance. The registry level contains three primary entities: *abstract service*, *abstract method*, and *abstract data product*. There are two special abstract services: *composite service* and *opaque service*. A composite service represents a workflow having member services; the

relationship is represented by “has sub-service”. An opaque service represents a black-box workflow in which the internals of services involved are not known. For predefined workflows, workflow structure in terms of services is represented by the relationship “has next-service”, which means a service has another service following it. An abstract service can have zero or more abstract methods. Each abstract method has zero or more data products as input and output. The input/output parameter could be either a granule or collection. If an input/output parameter for a method has multiple components, it is represented as “abstract data collection”; otherwise it is an “abstract data granule”. In summary, the registry level captures the metadata of services (abstract WSDL), the methods inside a service, the name and type of input parameters and output results of each method, and the structure of workflows in terms of services for predefined workflows. The order of method execution is recorded in the execution level by method invocation.

The execution level models instances of the registry level and records the execution-related information of method invocations and data products used or generated by each invocation. *Service, method invocation, and data product* in the execution level are, respectively, instances of abstract service, abstract method, and abstract data product in the registry level. A method can be invoked by a service (represented by “service method invocation”), a method, or a client (represented by “client method invocation”). The client is an entity which initiates workflows or services; it could be a user or a workflow engine. The execution information of a method invocation includes request/response status and timestamp, and location, initialization and termination timestamp of a service for “service method invocation”, or client id for “client method invocation”. The values, sizes, URL (for web services), timestamp of input parameters and output results are recorded in data product. Similar to the registry level, the input/output parameter for a method could be either a granule or collection.

Compared to the layered model as used for instance by Barga et al. [1], Karma has three advantages. First, Karma can capture and store streaming provenance data, that is, the workflow can be unknown until execution. Second, for services and data products which may be used by multiple workflows, Karma stores the metadata only once in the registry level to provide uniformity and avoid redundancy. Third, for predefined workflows, Karma stores the structure of the workflow only once at the registry level and separate instances of run-time provenance at the execution level, which allows efficient storage for the repeated workflow executions.

## V. APPLICATIONS OF INFORMATION MODEL

From the information model we construct a database schema used for provenance data storage and query, the Open Provenance Model (OPM) representation for sharing provenance data with other systems, and the event model for event collection.

### A. Database Schema

The provenance database schema represents the information model with a 1:1 mapping to store and query provenance data. Following the information model, the database tables are also in two levels.

The primary tables at the registry level include *abstract service, abstract method, abstract data product*, and the *input/output* relationship between abstract data product and abstract method. There are three auxiliary tables to store *composite service, workflow structure, and data collection*.

The tables at the execution level are derived from those used in Karma2 [8], with the extension of client, method, and data collection to comply with the information model. Obviously, *service, method, and data product* store the run-time instance data. Note that although the tables at the execution level store instance data for the tables at the registry level, the primary-foreign key relationships between these two levels are not mandatory. We represent a *method invocation* in terms of *invoker* and *invokee*, where an invoker is the one who initiates the invocation and an invokee is the one who is invoked. Both invoker and invokee are *entities*. The invoker can be a client (i.e., client method invocation), or a service (i.e., service method invocation). We assign an entity id for each invoker and invokee, and these ids are used to represent the invocation relationship.

### B. OPM Model

Emerging from the e-Science provenance community, the Open Provenance Model (OPM) [9] is evolving as a standardized representation of historical provenance graphs. OPM defines a model and a set of inference rules for provenance graphs.

OPM defines three primary entities:

- *Artifacts*: immutable piece of state;
- *Processes*: actions or series of actions performed on or caused by artifacts; and
- *Agents*: contextual entities that act as a catalyst of a process.

These three OPM entities are represented in Karma as follows:

- Artifacts: Data Product (Data Granule and Data Collection).
- Process: Service (Composite Service and Opaque Service) and methods.
- Agent: Client. The “Client” may be a user or a workflow engine that initiates the “Workflow”.

OPM defines five types of causal dependencies:

- Type 1: process *used* artifact
- Type 2: artifact *was generated by* a process
- Type 3: process *was triggered by* another process
- Type 4: artifact *was generated by* another artifact
- Type 5: process *was controlled by* agent

These causality edges use the past tense referring to an execution that has completed. Karma has the following causal dependencies that map to OPM dependencies:

- Method invocation `<has input>` data granule/collection (Type 1)

- Data granule/collection <Inverse(has output)> method invocation (Type 2)
- Abstract service <has next method> another abstract service (Type 3)
- Composite service <has sub method> another abstract service (Type 3)
- Data granule 2 <inverse(has output)> method invocation <has input> data granule 1 (Type 4)
- Service Instance <Inverse(invokes)> Client (Type 5)

Since our information model conforms to OPM, it is easy to generate OPM compliant provenance “graphs” from our provenance database for facilitating sharing provenance data and interoperability with other systems. From the database point of view, the OPM model can be created as materialized views since provenance is immutable.

### C. Event Model

Unlike other provenance enabled workflow systems, Karma is not tightly coupled to a particular workflow system. This gives Karma the flexibility to capture provenance from different kinds of systems. One of the important systems we are working on is the *event processing system*. In our experience over the past six years in developing and supporting the National Science Foundation funded Linked Environments for Atmospheric Discovery (LEAD) project [14], we have seen the benefits and importance of events and event processing systems in multiple contexts.

Generally, a service generates an event and sends it to a bus (e.g., the Microsoft .NET service bus, or WS-Messenger bus), on which one or more services are listening. When an event arrives, these services pick it up and execute their own tasks based on the event. By tracking the service that generated an event, and the services that consumed an event, relationships can be established either explicitly or causally. In our information model, we store the event generated by a service as a data product. Obviously, the event model is a subset of the information model and it can be created using materialized views like the OPM model.

## VI. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce a two-layer information model for representing provenance information at a level of detail sufficient for addressing the long-term preservation of digital scientific data products. As our experience with information representation in Karma has matured, we have evolved to the need for the registry level that is more than just an abstract representation such as a workflow script. In ongoing work, we are implementing this information model in the new release of Karma (Version 3). We are applying

the new model to the provenance collection we are carrying out in the Life Science Grid (LSG) [4] project. Future work includes provenance storage over large collections, and provenance query capabilities of the new model.

Additionally we continue to extend Karma’s instrumentation support to such frameworks as ERMA (Extremely Reusable Monitoring API) [15]. Finally, we are at a relatively early stage of examining the exporting of preservation bundles.

### ACKNOWLEDGMENT

This work is funded through NSF OCI-0721674 and funding from Eli Lilly Corporation.

### REFERENCES

- [1] R. Barga, L. Digiampietri, “Automatic capture and efficient storage of e-Science experiment provenance,” *Concurrency and Computation: Practice and Experience*, 20(5): 419 – 429, 2008.
- [2] M. Brodie and J. Schmidt, “Final report of the ansi/x3/sparc dbs-sg relational database task group,” *SIGMOD Record*, 12(4): 1–62, 1982.
- [3] D. Holland, M. Seltzer, U. Braun and K. Muniswamy-Reddy, “PASSing the provenance challenge,” *Concurrency and Computation: Practice and Experience*, 20(5): 531 – 540, 2008.
- [4] Life Sciences Grid, <http://sourceforge.net/projects/lsg/>
- [5] E. Robertson. “Explicitly Modeling Metadata,” unpublished whitepaper. Computer Science Department, 2008.
- [6] C. Scheidegger, D. Koop, E. Santos, H. Vo, S. Callahan, J. Freire, and C. Silva, “Tackling the Provenance Challenge one layer at a time,” *Concurrency and Computation: Practice and Experience*, 20(5): 473 – 483, 2008.
- [7] Y. Simmhan, B. Plale, and D. Gannon, “A survey of data provenance in e-Science,” *ACM SIGMOD Record*, 34(3): 31-36, 2005.
- [8] Y. Simmhan, B. Plale, and D. Gannon, “Karma2: Provenance Management for Data-Driven Workflows,” *Int’l Journal of Web Services Research*, 5(2): 1-22, 2008.
- [9] The Open Provenance Model (v1.01), <http://eprints.ecs.soton.ac.uk/16148/1/opm-v1.01.pdf>
- [10] J. Zhao, C. Goble, R. Stevens, and D. Turi, “Mining Taverna’s semantic web of provenance,” *Concurrency and Computation: Practice and Experience*, 20(5): 463 – 472, 2008.
- [11] Effective SOA Deployment Using an SOA Registry Repository, [http://www.sun.com/products/soa/registry/soa\\_registry\\_wp.pdf](http://www.sun.com/products/soa/registry/soa_registry_wp.pdf).
- [12] I. Foster, J. Vöckler, M. Wilde, and Y. Zhao, “Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation,” in *Proceedings of SSDBM*, pp. 37-46, 2002.
- [13] C. Lin, S. Lu, Z. Lai, A. Chebotko, X. Fei, J. Hua, and F. Fotouhi, “Service-Oriented Architecture for VIEW: a Visual Scientific Workflow Management System,” in *Proceedings of SCC*, pp. 335-342, 2008.
- [14] LEAD portal, <https://portal.leadproject.org/gridsphere/gridsphere>.
- [15] ERMA: The Extremely Reusable Monitoring API, <http://erma.wikidot.com>.