Understanding Collaborative Studies through Interoperable Workflow Provenance

Ilkay Altintas[1,2], Manish Kumar Anand[1], Daniel Crawl[1], Shawn Bowers[3], Adam Belloum[2], Paolo Missier[4], Bertram Ludäscher[5], Carole A. Goble[4], and Peter M.A. Sloot[2]

[1] San Diego Supercomputer Center, University of California, San Diego, USA
{altintas,mkanand,crawl}@sdsc.edu
[2] Computational Science, University of Amsterdam, The Netherlands
{A.S.Z.Belloum,p.m.a.sloot}@uva.nl
[3] Department of Computer Science, Gonzaga University
bowers@gonzaga.edu
[4] School of Computer Science, University of Manchester, Manchester, UK
{pmissier,carole.goble}@cs.man.ac.uk
[5] UC Davis Genome Center, University of California, Davis
ludaesch@ucdavis.edu

Abstract. The provenance of a data product contains information about how the product was derived, and is crucial for enabling scientists to easily understand, reproduce, and verify scientific results. Currently, most provenance models are designed to capture the provenance related to a single run, and mostly executed by a single user. However, a scientific discovery is often the result of methodical execution of many scientific workflows with many datasets produced at different times by one or more users. Further, to promote and facilitate exchange of information between multiple workflow systems supporting provenance, the Open Provenance Model (OPM) has been proposed by the scientific workflow community. In this paper, we describe a new query model that captures implicit user collaborations. We show how this model maps to OPM and helps to answer collaborative queries, e.g., identifying combined workflows and contributions of users collaborating on a project based on the records of previous workflow executions. We also adopt and extend the high-level Query Language for Provenance (QLP) with additional constructs, and show how these extensions allow non-expert users to express collaborative provenance queries against this model easily and concisely. Furthermore, we adopt the Provenance Challenge 3 (PC3) workflows as a collaborative and interoperable usecase scenario, where different stages of the workflow are executed in three different workflow environments - Kepler, Taverna, and WSVLAM. Through this usecase, we demonstrate how we can establish and understand collaborative studies through interoperable workflow provenance.

1 Introduction

As scientific knowledge grows and the number of studies that require access to knowledge from multiple scientific disciplines increase, the complexity of scientific problems

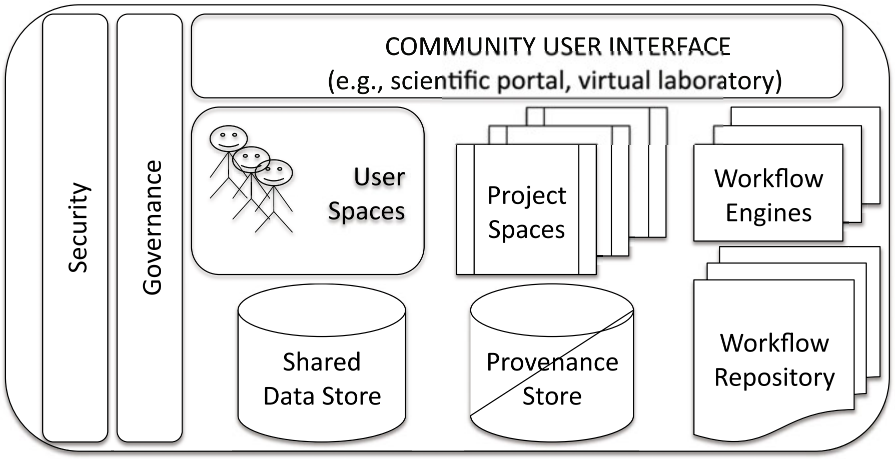
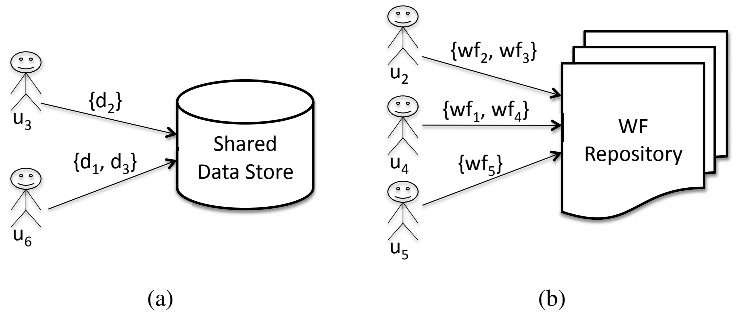


Fig. 1. Component architecture of a typical scientific research infrastructure (external data, service and computational infrastructure not shown)

amplifies. To cope with this complexity, scientists use computational methods that are evolving almost daily. However, the basic scientific method remains the same while being continuously transformed from manual to automated with the advances in computer science and technology. This gradual shift from manual process execution to automation of repeatable patterns resulted in the creation of scientific workflow systems.

Scientific workflow management systems [1,2,3,4] are critical to the way a modern scientist conducts studies today by making technological advances more approachable through integrative interfaces and abstractions for underlying computational and data resources. Scientific workflow systems allow scientists to develop formal, customizable, reusable and extensible definitions for all or part of a scientific process and execute them efficiently. In addition, using scientific workflows to perform computational experiments on data unleashes the possibility to maintain its provenance [5]. Typically designed iteratively by a user and ran multiple times by one or more users, the provenance of a scientific workflow provides a rich source for conducting similar future scientific studies [6]. However, this is still only a partial solution to the modern scientific process that relies on multi-disciplinary collaborative teams working on different parts of scientific studies. Currently, provenance support in workflow systems are mostly designed to capture the information related to a single workflow run by a user. On the other hand, the collaborative process often involves design and execution of multiple workflows [7] where different members of a team conceptualize their contribution as workflows and make it available through a common infrastructure. A scientific discovery is the result of methodical execution of many of these workflows with many datasets at different times by one or more users.

Community portals [8], virtual laboratories [9], and Web2.0-based social networking and sharing environments [10] are popular platforms to establish a common infrastructure where community members can contribute to data, workflows and projects through



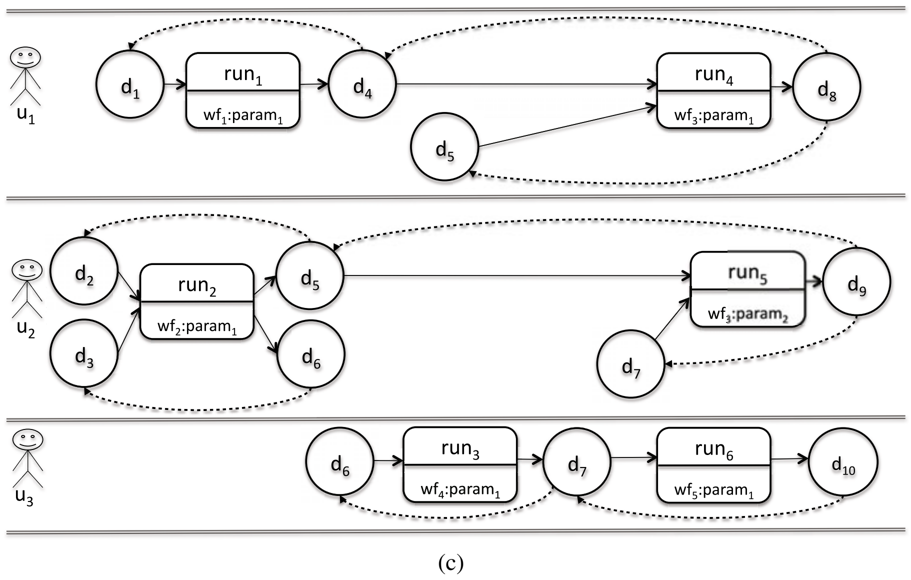
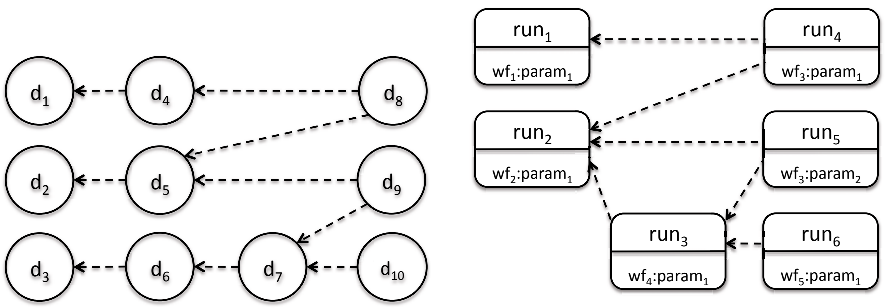


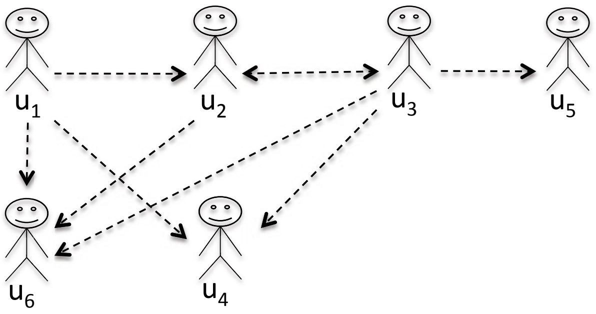
Fig. 2. Different observables of shared data, workflows and their runs in a typical scientific research project: (a) data ($\{d_1, d_2, d_3\}$) published by users in $\{u_1, u_6\}$; (b) ready to run workflows ($\{wf_1 .. wf_5\}$) published by users in $\{u_2, u_4, u_5\}$; (c) flow graph for published workflow runs (customized through their parameters) and related data ($\{d_1 .. d_{10}\}$) in user spaces ($\{u_1, u_2, u_3\}$), separated by horizontal dashed lines

their user spaces under generic governance rules. Workflows could be executed multiple times by one or more scientists, potentially from an end-user interface that combines several workflows. In addition, the executed workflows use data from external data resources and the scientific outputs are saved in data repositories, optionally along with intermediate results and the process provenance. A typical set of components for such an infrastructure is illustrated in Fig. 1.

The discussion in this paper lies in the heart of these sharing and execution practices in e-science projects where the overall execution of a set of workflows can result in an overarching model of the scientific process leading to data artifacts. All the observables



(a) (b)



(c)

Fig. 3. Different views generated by modeling and analysis of runs in a typical scientific research project: (a) data dependency view; (b) run dependency view; (c) overall non-transitive directed implicit user ($\{u_1 .. u_6\}$) collaboration view

in such a project within a three- dimensional space of users, workflows and data are illustrated in Fig. 2. The history of workflow runs in different user spaces $\{u_1, u_2, u_3\}$ depicted in Fig. 2(c) shows the usage of published data in Fig. 2(a) and workflows in Fig. 2(b). Users who performed the workflow runs or used published data start an "implicit collaboration". In Fig. 2(c), a run node identifies the provenance of a previous workflow run and the fine-grained data dependencies are shown by dashed links between data nodes. One can identify the flow of workflow executions leading to a data artifact that is published as a "scientific discovery" by chaining together the runs performed by users. This chaining happens through data artifacts consumed and produced by workflow runs, e.g., in Fig. 2(c), d_5 produced by run_2 of u_2 is consumed by run_4 performed by u_1, creating a link between run_2 and run_4.

A goal of this approach is to extend the current single-workflow and single-user targeted provenance approach to a number of workflow runs within a controlled environment such as a website community portal for sharing data and workflows. In this paper, we assume that the data store is publicly shared between users. Using this extended information, one can generate views of data dependencies, related workflow executions and user collaborations, as seen in Fig. 3(a), Fig. 3(b) and Fig. 3(c) respectively. In addition, it becomes possible to answer queries for potential acknowledgements of a scientific result and the correction trail of a faulty data item. Another important goal is to propose and demonstrate an architecture that facilitates the interoperability of different workflow systems through provenance of workflow runs and related data. We assume the model of provenance is shared between different workflow systems and provides a global repository of data artifact identifiers, i.e., an artifact produced by one workflow system and consumed by another can be uniquely identified. The design of this repository is not in the scope of this paper. This new approach puts user actions and collaborations in the center of the conducted research independent of computational technologies used to generate results.

Contributions. We investigate the implicit user collaborations in a QLP-based [11] query model that maps to OPM [12] using observables in an e-science infrastructure (Fig. 2) and for generating views on top of them (Fig. 3). This approach links OPM graphs for workflow runs that have an input or output data dependency and helps to answer queries such as identifying data connections between workflow runs and contributions of users collaborating on a project based on the records of past executions. We adopt and extend a high-level query language for provenance, QLP, to express complex collaborative provenance queries. We also establish a mapping between QLP and OPM. Furthermore, through the PC3 (http://twiki.ipaw.info/bin/view/Challenge/) usecase scenario, we demonstrate the feasibility of how our approach will lead to development of systems that increase *interoperability* and *reusability* of workflow results by integrating provenance coming out of *different* workflow systems and, in turn, enhancing efficiency in modern collaborative research.

Outline. The organization of this paper is as follows. In Section 2, we introduce the concept of collaborative views and queries over interoperable provenance data. Section 3 explains QLP and the extensions we build on top of QLP that map to OPM constructs along with the QLP expressions of queries defined in Section 2. A feasibility study for the explained techniques is provided in Section 4, based on PC3 workflows. We review background work in Section 5 and conclude in Section 6.

2 Building Collaborative Views

The lifecycle of scientific workflows—which includes the design, execution, sharing, and management of data and provenance products—depends not only on the workflow itself, but also the overall scientific research infrastructure and scientific collaborations within which scientists use these workflows. In this section, we introduce the concept of collaborative views based on the provenance of workflows and user actions within a scientific infrastructure (see Fig. 2). We also present example queries that are enabled by our provenance model, including those that allow scientists to determine implicit collaborative relationships. The basic relations we use to develop collaborative views are shown in Fig. 4. We first describe these relations, and then show how they enable the construction of both standard and collaborative provenance views.

The relation $\mathsf{Run}(r, w)$ states that r is a run (i.e., execution) of a workflow w (shown using rounded boxes in Fig. 4). We assume every run is of exactly one workflow. Each run

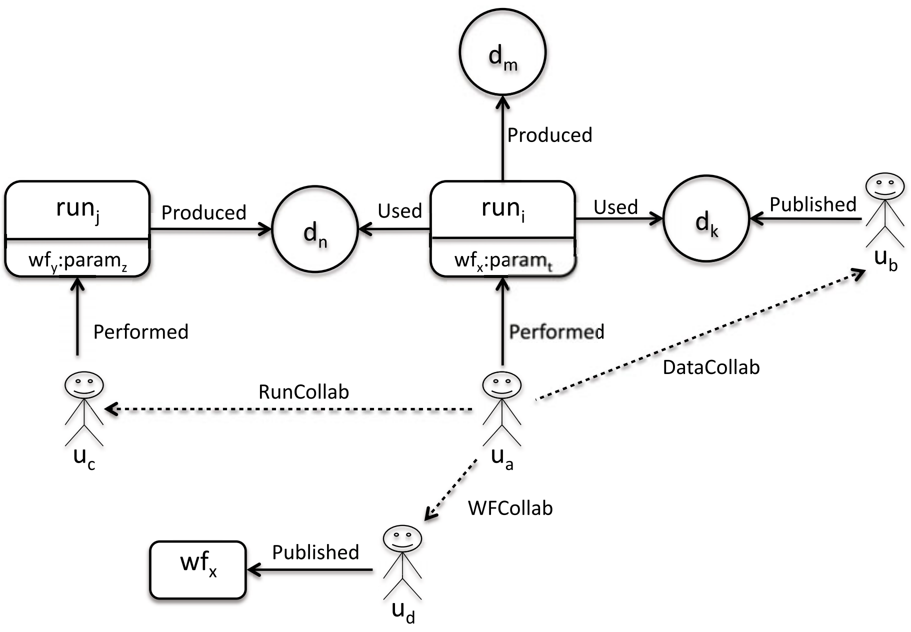


Fig. 4. The main entities and edges of the collaborative provenance model

r can take zero or more data artifacts d_{in} as input according to the relation $\mathsf{Input}(r, d_{in})$, which states that d_{in} was input to r. Each run r can also have zero or more data artifacts d_{out} as outputs according to the relation $\mathsf{Output}(r, d_{out})$, which states that d_{out} was an output of r. A data artifact can be an output of at most one run, but can be used as an input to zero or more runs.

Although not shown directly in Fig. 4, we assume a relation $\mathsf{DerivedFrom}(d_{out}, r, d_{in})$ for capturing causal dependencies between input and output data items of a run r. Given a fact $\mathsf{DerivedFrom}(d_{out}, r, d_{in})$, we say that d_{out} was derived from r using d_{in}. Each derivation also implies that d_{in} was an input to r, i.e., $\mathsf{Input}(r, d_{in})$, and d_{out} was an output of r, i.e., $\mathsf{Output}(r, d_{out})$. This constraint is captured in first-order logic (FO) as

$$\forall d_{in}, r, d_{out}\ (\mathsf{DerivedFrom}(d_{out}, r, d_{in}) \rightarrow \mathsf{Input}(r, d_{in}) \wedge \mathsf{Output}(r, d_{out})).$$

We define the relation $\mathsf{DDep}(d_{out}, d_{in})$ as the set of all immediate data dependencies given by $\mathsf{DerivedFrom}$, where d_{out} is said to depend on d_{in}. We can easily compute DDep using the following Datalog rule.

$$\mathsf{DDep}(d_{out}, d_{in})\ \ \text{:-}\ \ \mathsf{DerivedFrom}(d_{out}, r, d_{in}).$$

We write DDep^* to denote the transitive closure of the DDep relation.

The Used and $\mathsf{Produced}$ relations (as shown in Fig. 4) are variants of Input and Output that additionally imply a derivation relationship. These relations are defined as views over Input and Output using the following Datalog rules.

$$\begin{aligned}\mathsf{Used}(r,d_{in}) &\;:\!-\; \mathsf{DerivedFrom}(d_{out},r,d_{in}).\\ \mathsf{Produced}(r,d_{out}) &\;:\!-\; \mathsf{DerivedFrom}(d_{out},r,d_{in}).\end{aligned}$$

The first rule states that a data artifact d_{in} was used by a run r if d_{in} derived an output d_{out} of r. The second rule states that a data artifact d_{out} was produced by a run r if it was derived by an input d_{in} of r. Note that these relations do not explicitly link the inputs and outputs of a derivation, which is only done through the DerivedFrom relation.

We define the relation $\mathsf{RDep}(r_2,r_1)$ as the set of all immediate run dependencies, where r_2 is said to depend on r_1. The RDep relation is defined as the following view in Datalog.

$$\mathsf{RDep}(r_2,r_1) \;:\!-\; \mathsf{Output}(r_1,d_1),\mathsf{Used}(r_2,d_1).$$

Specifically, a run dependency is established between a run r_2 and r_1 whenever the output of r_1 is used by r_2 to derive a data artifact.

We assume the relation $\mathsf{Published}(u,w)$ records the case when a user u published a workflow w to the workflow repository; and similarly, $\mathsf{Published}(u,d)$ records the case when u published a data artifact d to the shared data store (see Fig. 2). A user u may also perform, i.e., execute and then publish, a workflow run r, which is captured by the relation $\mathsf{Performed}(u,r)$. When a user performs a run, we assume all outputs of the run are published to the data store, which is captured by the following FO constraint.

$$\forall u,r,d\; (\mathsf{Performed}(u,r) \wedge \mathsf{Output}(r,d) \rightarrow \mathsf{Published}(u,d)).$$

As shown in Fig. 4, we consider three variants of collaboration, which we define as views using the following Datalog rules.

$$\begin{aligned}\mathsf{WFCollab}(u_2,u_1) &\;:\!-\; \mathsf{Published}(u_1,w),\mathsf{Run}(r,w),\mathsf{Performed}(u_2,r).\\ \mathsf{DataCollab}(u_2,u_1) &\;:\!-\; \mathsf{Published}(u_1,d_1),\mathsf{Used}(r,d_1),\mathsf{Performed}(u_2,r).\\ \mathsf{RunCollab}(u_2,u_1) &\;:\!-\; \mathsf{Performed}(u_1,r_1),\mathsf{Output}(r_1,d_1),\mathsf{Used}(r_2,d_1),\\ &\qquad \mathsf{Performed}(u_2,r_2).\end{aligned}$$

The first rule states that a *workflow collaboration* (WFCollab) is established between two users whenever the second user executes a workflow that is publishes by the first user. The second rule states that a *data collaboration* (DataCollab) is established between two users whenever a data artifact published by the first user was used as an input by a run that is performed by the second user. The third rule states that a *run collaboration* (RunCollab) is established between two users whenever a run performed by the second user uses the output of a run by the first user.

A collaboration dependency $\mathsf{CDep}(u_2,u_1)$ between two users is established whenever they participate in one of the three collaborations defined above (where u_2 depends on u_1). The CDep relation is easily defined in Datalog as follows.

$$\begin{aligned}\mathsf{CDep}(u_2,u_1) &\;:\!-\; \mathsf{WFCollab}(u_2,u_1).\\ \mathsf{CDep}(u_2,u_1) &\;:\!-\; \mathsf{DataCollab}(u_2,u_1).\\ \mathsf{CDep}(u_2,u_1) &\;:\!-\; \mathsf{RunCollab}(u_2,u_1).\end{aligned}$$

Table 1. Example queries across workflow executions and collaborations

Q1	Which data artifacts were used explicilty or implicitly to generate data artifact d?
Q2	Which runs were used in the generation of a data artifact d?
Q3	If data artifact d is detected to be faulty, which runs were affected by d?
Q4	Which users depended on data artifact d?
Q5	Which user collaborations were involved in the derivation of artifact d_2 from artifact d_1?
Q6	Who are the potential acknowledgements for a publication of a data artifact d?

Each of the views shown in Fig. 3 can be reconstructed from the provenance model described here. The relations DDep and RDep defined above can be used to construct the standard data and run dependency graphs shown in Fig. 3(a) and 3(b), respectively. More importantly, using the CDep relation, we can also construct user collaboration views (i.e., the collaboration dependency graph) as in Fig. 3(c). With these three dependency graphs, it becomes possible to answer both standard provenance queries as well as queries that involve user collaborations. In the following section we extend the model presented here (with respect to the three dependency graphs) to addtionally support lineage-based path queries. Our approach provides a simple mechanism for filtering dependency graphs to answer provenance queries such as those in Table 1.

3 Expressing Collaborative Queries

We use QLP (the *Query Language for Provenance*) [11] for expressing lineage queries, and in particular, to filter the dependency graphs described in Section 2. In general, answering standard provenance questions (including those of Table 1) requires the generation of recursive queries over lineage graphs. Such queries are often complex to express and expensive to evaluate [12,13,14,15]. QLP provides a simple, declarative, path-based language (similar, e.g., to XPath) for expressing such queries, and optimization techniques have been developed that make answering QLP queries over large provenance repositories feasible [16]. QLP queries work over sets of lineage edges, e.g., represented by the DerivedFrom relation. A QLP path query p can be viewed as a filter that selects matching paths within the lineage graph induced by the underlying edges. The result of a QLP query is the set of edges along matching paths of the induced graph. Thus, QLP is a closed language that returns a subset of a given set of lineage edges. Closed languages such as QLP have a number of benefits including the ability to construct views, "incremental" querying, and visualization [17,18].

QLP queries are expressed and evaluated against a selected provenance view, which can be a single workflow run, the entire repository of runs, or the provenance view resulting from a previous query. In the collaborative provenance query scenario, users can use QLP expressions to filter the various dependency views of Fig. 3. Below we present the basic constructs of QLP and show how QLP can be used to filter dependency graphs (and subsequently answer the queries of Table 1; see Table 2).

In the scenario illustrated by Fig. 2, the lineage information is recorded at a "coarse-grain" level, where only the lineage relationships between inputs and outputs of a run

Lineage-preserving path queries (examples)	
$*$ .. e_n	Lineage graph that resulted in nodes in e_n.
e_n .. $*$	Lineage graph for nodes derived from nodes in e_n
e_{n_1} .. e_{n_2}	Lineage graph for paths from nodes in e_{n_1} to nodes in e_{n_2}
e_{n_1} .. r_i .. e_{n_2}	Lineage graph for paths from nodes in e_{n_1} to e_{n_2} passing through run r_i
Functions over lineage path queries	
exists(p)	True if the selected view contains a path defined by path query p
runs(p)	The runs of the lineage graph returned by path query p
workflows(p)	The workflows of the lineage graph returned by path query p
artifacts(p)	The data nodes of the lineage graph returned by path query p
inputs(p)	The source nodes of the lineage graph returned by path query p
outputs(p)	The sink nodes of the lineage graph returned by path query p
Views over lineage path queries	
DATA-DEP(p)	Data dependencies (Fig. 3(a)) of the lineage graph selected by path query p
RUN-DEP(p)	Run dependencies (Fig. 3(b)) of the lineage graph selected by path query p
COLLAB-DEP(p)	Collaborations (Fig. 3(c)) of the lineage graph selected by path query p

Fig. 5. Basic QLP constructs and functions, where e_n is a node expression comprised of either a data artifact identifier, a run identifier, a data artifact type (denoting the set of artifacts having that type), or a workflow (denoting the set of runs of the workflow). We use p to denote a QLP path query, and r_i to denote a run.

are stored. In the following, we restrict the underlying lineage model of QLP to be over workflow runs, as opposed to the standard use of QLP that supports queries over indidvidual processes within runs (thus modeling lineage at a "fine-grain" level).

Table 5 introduces some of the basic constructs and functions of QLP, together with the extensions described here, including the DATA-DEP, RUN-DEP, and COLLAB-DEP functions. As a simple example of a QLP path query, the expression "$*$ .. d_7" returns lineage edges denoting paths starting from any node in the lineage graph and ending at node d_7. Similarly, the query "d_2 .. $*$" returns lineage edges denoting paths starting at node d_2 and ending at any node in the lineage graph. Both "ends" of a path can be fixed in QLP, e.g., the query "d_5 .. d_9" returns all edges on paths in the lineage graph that start at d_5 and end at d_9. QLP queries can restrict paths to include intermediate objects, e.g., the query "#r_2 .. d_6 .. #r_5 .. $*$" returns the set of lineage edges denoting paths that start at run r_2, go through artifact d_6 followed by (via one or more lineage edges) run r_5, and end at any node.

3.1 Filtering Dependency Views Using QLP

The DATA-DEP, RUN-DEP, and COLLAB-DEP functions construct data, run, and collaboration dependency graphs, respectively, that result from evaluating a QLP query over the current provenance view. Thus, these functions, unlike the DDep, RDep, and CDep relations defined in Section 2, create views purely out of lineage relations.

Filtering Data Dependency Views. We write $v(p)$ to denote the set of lineage edges of the form $\langle d_2, r, d_1 \rangle \in L$ returned after evaluating a QLP path query p over a set of lineage edges L [16]. We directly use this evaluation to define the DATA-DEP function as follows.

$$\text{DATA-DEP}(p) := \{\langle d_2, d_1\rangle \mid \exists r : \langle d_2, r, d_1\rangle \in v(p)\}$$

As shown in Table 2, we can use the DATA-DEP function to answer Q1 of Table 1, which returns the subset of the data-dependency graph that ends at artifact d. Note that the DATA-DEP function computes a subset of the DDep relation restricted to lineage edges.

Filtering Run Dependency Views. Similarly, to construct a filtered run-dependency graph, we again use the evaluation function as follows.

$$\text{RUN-DEP}(p) := \{\langle r_2, r_1\rangle \mid \exists d_1, d_2, d_3 : \langle d_3, r_2, d_2\rangle \in v(p) \wedge \langle d_2, r_1, d_1\rangle \in v(p)\}$$

Note that each output of a run within a lineage graph returned by a QLP query is required to be dependent on some input (since only derivation edges are considered). Thus, the run dependencies returned by the RUN-DEP function have the additional constraint that each output is dependent on some input (within the query result) of the run. This can be viewed as restricting the RDep relation to only selecting from Produced edges instead of Output edges. We can use the RUN-DEP function to answer Q2 and Q3 of Table 1, as shown in Table 2.

Filtering User Collaboration Views. Let DATA-DEP$^*(p)$ be the set of edges of the transitive closure of the edges returned by DATA-DEP(p). We define the COLLAB-DEP function as:

$$\text{COLLAB-DEP}(p) := \text{C-DEP}_{WF}(p) \cup \text{C-DEP}_{DATA}(p) \cup \text{C-DEP}_{RUN}(p),$$

where the functions C-DEP$_{WF}$, C-DEP$_{DATA}$, and C-DEP$_{RUN}$ are defined as follows.

$$\begin{aligned}
\text{C-DEP}_{WF}(p) &:= \{\langle u_2, u_1\rangle \mid \exists r, w : \text{Run}(r, w) \wedge \text{Published}(u_1, w) \wedge \\
&\qquad \text{Performed}(u_2, r)\} \\
\text{C-DEP}_{Data}(p) &:= \{\langle u_2, u_1\rangle \mid \exists d_1, d_2, r : \text{Published}(u_1, d_1) \wedge \text{Performed}(u_2, r) \wedge \\
&\qquad \langle d_2, r, d_1\rangle \in v(p)\} \\
\text{C-DEP}_{Run}(p) &:= \{\langle u_2, u_1\rangle \mid \exists d_0, d_1, d_2, r_1, r_2 : \text{Performed}(u_1, r_1) \wedge \\
&\qquad \langle d_1, r_1, d_0\rangle \in v(p) \wedge \langle d_2, r_2, d_1\rangle \in v(p) \wedge \text{Performed}(u_2, r_2)\}
\end{aligned}$$

The COLLAB-DEP function can be used to answer queries Q4-Q6 of Table 1, as shown in Table 2.

3.2 Relation between the Collaborative Model and OPM

The Open Provenance Model (OPM) [12] has emerged from the e-science community, and has evolved as a standard representation to facilitate the exchange of information between multiple provenance systems. OPM is based on a model and set of inference rules for directed acyclic provenance graphs, which represent casual dependencies between data products and processes. OPM defines three primary entities (nodes): (1)

Table 2. Example queries expressed using the dependency functions defined for QLP

Q1	DATA-DEP(* .. d)
Q2	RUN-DEP(* .. d)
Q3	RUN-DEP(d .. *)
Q4	COLLAB-DEP(d .. *)
Q5	COLLAB-DEP(d_1.. d_2)
Q6	COLLAB-DEP(* .. d)

Artifacts: immutable piece of data; (2) *Processes*: actions or series of actions performed on or caused by artifacts; and (3) *Agents*: entities that enable, facilitate, control, or affect execution of processes. OPM also defines five primary types of causal dependencies (edges) that comprise provenance graphs: (1) *used*: a process used artifact(s); (2) *wasGeneratedBy*: an artifact was generated by a process; (3) *wasTriggeredBy*: a process was triggered by another process(es); (4) *wasDerivedFrom*: an artifact was derived from another artifact(s); and (5) *wasControlledBy*: a process was controlled by an agent.

The collaborative model, e.g., as in Fig. 4, roughly contains the same entities and five causal dependencies of OPM. A lineage (i.e., DerivedFrom) relation is of the form $\langle d_{out}, r, d_{in}\rangle$ for data nodes (artifacts in OPM) d_{in} and d_{out} and run (or processes in OPM) r. For example, in Fig. 3(b) $\langle d_4, r_1, d_1\rangle$ is a lineage relation stating that artifact d_1 was *used* by the run r_1 to produce artifact d_4 (i.e., artifact d_4 *wasDerivedFrom* artifact d_1), and artifact d_4 *wasGeneratedBy* workflow run r_1. Adjacent lineage relations, e.g., $\langle d_7, r_3, d_4\rangle$ and $\langle d_4, r_1, d_1\rangle$ state that run r_3 *wasTriggeredBy* run r_1. Similarly, users in the collaborative model can be viewed as a form of agents in OPM, where Performed edges are similar to *wasControlledBy* edges in OPM. To the best of our knowledge, OPM does not provide support for recording when users publish data and workflows, which is essential in the collaborative model proposed here for creating the various types of user collaborations described in Section 2.

4 Conceptual Interoperability Scenario

To further illustrate our approach, we describe (i) an interoperability scenario, derived from the Third Provenance Challenge (PC3), (ii) some prototypical collaborative queries, and (iii) an architecture for its implementation. A similar example based on the First Provenance Challenge is currently being implemented in the context of a DataONE[1] student project.

Usecase. The workflows selected for PC3 are part of an image-processing pipeline in the Pan-STARRS[2] project. A next generation panoramic telescope surveys the sky looking for asteroids or comets that may impact the Earth. The telescope may generate several TBs of data nightly, which must be reduced and stored into an object data management framework that is publicly accessible by astronomers. Based on this usecase,

[1] `http://dataone.org`

[2] `http://pan-starrs.ifa.hawaii.edu`

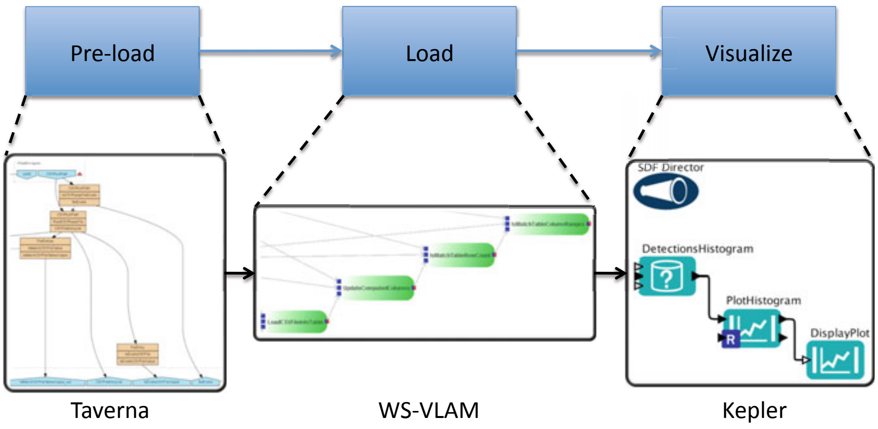


Fig. 6. Conceptual process for the Provenance Challenge 3

the main PC3 workflow ingests CSV files containing readings from the telescope into an SQL database and the plotting workflow creates histograms of the ingested data.

To build a collaborative workflow environment, we assume that we executed the fragments of these PC3 workflows in three different workflow systems as shown in Fig. 6. In this scenario, Taverna [19] performs the initialization and pre-loading checks, WS-VLAM [20] loads the CSV files into the database and updates the column counts, and Kepler [21] creates the histograms. We chose this division of the PC3 workflows to evenly and logically divide the tasks among the workflow engines.

An example history of observables and actions within this usecase is shown in Table 3. In this scenario, all three workflow engines use the same MySQL database when executing their subset of the PC3 workflows. In the pre-load tasks, Taverna verifies the contents of the input CSV files and creates the tables in the database. Next, WS-VLAM reads the contents of the CSV files into these tables, and verifies the row counts and data values. Finally, Kepler produces histograms from these data. For example, the second row refers to run_1 performed by u_2 using $wf_{preload}$ published by u_1. In run_1, u_2 used $d_{J062941}$ as an input and the run produced $d_{J062941-1}$ as its output.

Collaborative PC3 Queries. The following are example queries on Table 3 expressed using the QLP functions defined in Section 3.

Q1. What data contributed to $d_{histogram}$?
DATA-DEP($*$ .. $d_{histogram}$)

Q2. If $d_{J062942-2}$ is determined to be faulty, what other data products may be faulty based on $d_{J062942-2}$?
DATA-DEP($d_{J062942-2}$ .. $*$)

Q3. What runs contributed to the generation of $d_{J062941-2}$?
RUN-DEP($*$ .. $d_{J062941-2}$)

Q4. Which users contributed workflows that produced $d_{histogram}$?
COLLAB-DEP($*$ .. $d_{histogram}$)

Table 3. The publish and run observables in interoperable PC3 scenario. The contents of the table shall be read as follows: e.g., the second row refers to run_1 performed by u_2 using $wf_{preload}$ published by u_1. In run_1, u_2 used $d_{J062941}$ as an input and the run produced $d_{J062941-1}$ as its output.

u_1	Published			$wf_{Preload}$				
u_2	Performed	run_1	Used	$wf_{Preload}$	Used	$d_{J062941}$	Produced	$d_{J062941-1}$
u_3	Performed	run_2	Used	$wf_{Preload}$	Used	$d_{J062942}$	Produced	$d_{J062942-2}$
u_4	Published			wf_{Load}				
u_5	Published			$wf_{Visualize}$				
u_2	Performed	run_3	Used	wf_{Load}	Used	$d_{J062941-1}$	Produced	$d_{J062941-2}$
u_3	Performed	run_4	Used	$wf_{Visualize}$	Used	$d_{J062941-2}$	Produced	$d_{histogram}$
u_3	Published			$d_{histogram}$				

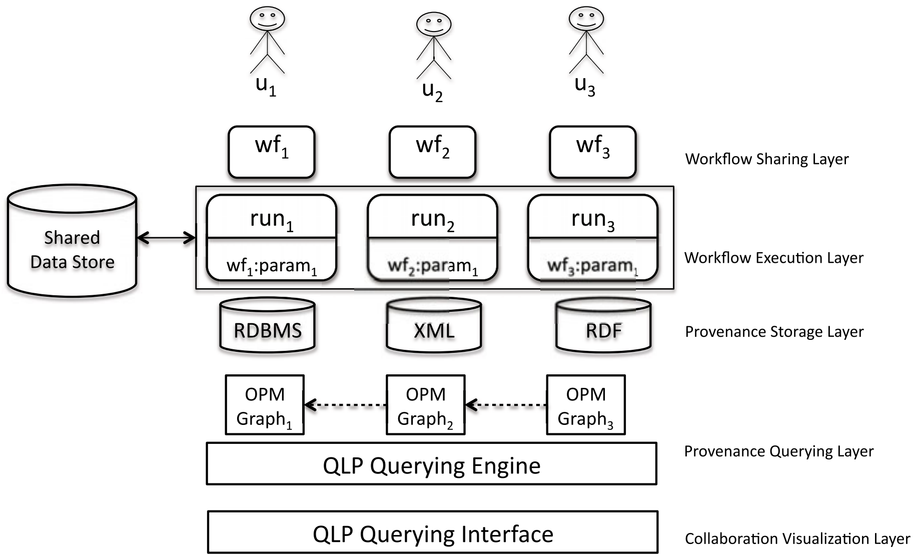


Fig. 7. Architecture for answering collaborative queries

QLP-based Interoperable Query Framework. Fig. 7 shows the design of an end-to-end framework that can be plugged into any scientific infrastructure with the ability to publish data and workflows, to execute workflows using different workflow engines, to collect workflow provenance and to express and evaluate QLP queries. In this architecture, workflows use a shared data space with common data identifiers. To generate data dependency views, using the QLP mapping to OPM, the QLP Querying Engine transforms users QLP queries into OPM queries. In addition, the same querying engine routes the mapped queries to distinct provenance stores using the developed SQL (RDBMS), XQuery (XML) and SPARQL (RDF) interfaces.

5 Related Work

The ideas presented in this paper depend on previous work in scientific workflow provenance and collaborative scientific platforms. Below we present the related work.

Provenance in Scientific Workflows. Scientific workflow systems are being used in many scientific domains, and many approaches have been proposed recently for representing and storing workflow provenance [5,6]. However, most of the existing provenance approaches store provenance for a single runs, and do not capture or maintain associations across runs [22,23,24,25]. The framework described in [7] records associations between multiple related workflow runs. Vistrails provides a spreadsheet where users can compare the results of multiple workflows, or multiple workflow runs[26]. However, our work is based on capturing associations not only across workflow runs, but also across users, where users play an active role of publishing data, or publishing workflows, or executing workflow runs. Our work captures these associations and establishes user collaboration views based on provenance.

Querying Provenance. Approaches for querying provenance are largely based on physical data representations [14], e.g., relational, XML, or RDF schemas, where users express provenance queries through corresponding query languages, i.e., SQL, XQuery, or SPARQL. Provenance queries often require computing transitive closures over dependency relations, and expressing such queries using standard approaches is typically done using recursion or stored procedures [15,16,27]. Expressing such queries is both cumbersome and error-prone, and requires considerable user expertise. High-level languages such as QLP provide a separation between the logical provenance model and its underlying physical representation, which allows for the use of different representation schemes and additional optimization techniques. Also, QLP is closed under lineage relations, where answers to lineage queries are sets of lineage dependencies (edges) forming provenance subgraphs, i.e., provenance preserving.

Collaborative Applications in E-Science. Since collaborative research studies require substantial infrastructure, we often see infrastructure projects that facilitate conducting a number of these multi- disciplinary scientific studies for a particular domain, e.g., Virolab [28], VL-e [9] and CAMERA [8]. In the VL-e project, the WFBus focuses on the execution of workflows developed in various workflow management systems. Collaborative views through provenance covers both the execution and provenance aspects of an aggregate workflow. In the context of the CAMERA project, workflow-related scientific products and their provenance are stored in data repositories that are accessible through the project portal, allowing for collaborative views and queries over these runs. In the ViroLab virtual laboratory, scientific applications are executed as scripts and their provenance is recorded by collecting events emitted by the GridSpace engine that executes the experiment scripts. Collaborative views over the provenance of these executing scripts can be captured by explicit reuse of results from previous experiments. An interesting opportunity arises from the support for Scientific Research Objects [29] by myExperiment [10]. Applications such as portal environments can deploy the content of SROs in new ways and collaborative views over them.

6 Conclusion

In this paper, we introduced the concept of collaborative views and queries over interoperable provenance data in a collaborative scientific research. We adopted and extended a high-level query language for provenance, QLP, to express complex collaborative provenance queries. We also established a mapping between QLP and OPM. Finally, we showed the feasibility of our approach on collaborative queries through PC3-inspired usecase workflows and described our planned architecture for its future implementation. The contributions of this paper tie together users actions with multiple workflow executions that create a chain of custody for data generated by collaborations.

This is the right time to introduce such provenance models and query languages as collaborative research projects are ever growing and Web2.0-oriented scientific sharing environments, e.g., myExperiment, are being introduced to allow for sharing and execution of workflows in different workflow system by groups of users. Thanks to Provenance Challenge efforts, OPM is starting to be adopted by workflow systems participating in the challenge pushing OPM as a standard for provenance data.

Future Work. In the future, we plan to publish an implementation of the QLP based collaborative query engine based on the PC3 workflow using workflows in Kepler, Taverna and WSVLAM. We are currently conducting a larger bioinformatics usecase from the CAMERA project where users share data, workflows and runs through shared stores. We also intend to work on aspects of restricted user spaces and optimization of collaborative query evaluation.

Acknowledgements. The authors would like to thank for their collaboration to the ViroLab consortium and the Kepler team. This research was supported by the Dutch Bsik project VL-e: Virtual Laboratory for e-Science and the European ViroLab grant INFSO-IST-027446, NSF SDCI Award OCI-0722079 for Kepler/CORE, DOE grant DE-FC02-01ER25486 for SciDAC/SDM, NSF CEO:P Award No. DBI 0619060 for REAP, NSF IIS-0630033 for pPOD and Gordon and Betty Moore Foundation award to Calit2 at UCSD for CAMERA.

References

1. Ludäscher, B., Goble, C. (eds.) Special section on scientific workflows. ACM SIGMOD Record 34(3) (2005)
2. Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M. (eds.) Workflows for e-Science. Springer, Heidelberg (2007)
3. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. IEEE Computer 40(12), 24–32 (2007)
4. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25, 528–540 (2009)
5. Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Record 34, 31–36 (2005)
6. Freire, J., Koop, D., Santos, E., Silva, C.T.: Provenance for computational tasks: A survey. Computing in Science and Engineering 10, 11–21 (2008)

7. Bowers, S., McPhillips, T., Wu, M.W., Ludäscher, B.: Project Histories: Managing Data Provenance Across Collection-Oriented Scientific Workflow Runs. In: Cohen-Boulakia, S., Tannen, V. (eds.) DILS 2007. LNCS (LNBI), vol. 4544, pp. 122–138. Springer, Heidelberg (2007)
8. Altintas, I., Lin, A.W., Chen, J., Churas, C., Gujral, M., Sun, S., Li, W., Manansala, R., Sedova, M., Grethe, J.S., Ellisman, M.: Camera 2.0: A data-centric metagenomics community infrastructure driven by scientific workflows. In: Proceeding of The IEEE 2010 Fourth International Workshop on Scientific Workflows, Miami, Florida (2010)
9. Zhao, Z., Booms, S., Belloum, A., de Laat, C., Hertzberger, B.: Vle-wfbus: A scientific workflow bus for multi e-science domains. In: International Conference on e-Science and Grid Computing (2006)
10. Roure, D.D., Goble, C., Stevens, R.: Designing the myexperiment virtual research environment for the social sharing of workflows. In: E-SCIENCE 2007: Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, Washington, DC, USA, pp. 603–610. IEEE Computer Society Press, Los Alamitos (2007)
11. Anand, M.K., Bowers, S., Mcphillips, T., Ludäscher, B.: Exploring scientific workflow provenance using hybrid queries over nested data and lineage graphs. In: SSDBM 2009: Proceedings of the 21st International Conference on SSDM, pp. 237–254. Springer, Heidelberg (2009)
12. Moreau, L., Freire, J., Futrelle, J., McGrath, R.E., Myers, J., Paulson, P.: The Open Provenance Model: An Overview. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 323–326. Springer, Heidelberg (2008)
13. Anand, M.K., Bowers, S., Ludäscher, B.: A navigation model for exploring scientific workflow provenance graphs. In: WORKS 2009: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science, pp. 1–10. ACM, New York (2009)
14. Cohen, S., Cohen-Boulakia, S., Davidson, S.B.: Towards a model of provenance and user views in scientific workflows. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 264–279. Springer, Heidelberg (2006)
15. Heinis, T., Alonso, G.: Efficient lineage tracking for scientific workflows. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1007–1018. ACM P, New York (2008)
16. Anand, M.K., Bowers, S., Ludäscher, B.: Techniques for efficiently querying scientific workflow provenance graphs. In: EDBT 2010: Proceedings of the 13th International Conference on Extending Database Technology, pp. 287–298. ACM, New York (2010)
17. Anand, M.K., Bowers, S., Altintas, I., Ludäscher, B.: Approaches for exploring and querying scientific workflow provenance graphs. In: IPAW (2010)
18. Anand, M.K., Bowers, S., Ludäscher, B.: Provenance browser: Displaying and querying scientific workflow provenance graphs (Demo). In: 26th IEEE International Conference on Data Engineering (2010)
19. Turi, D., Missier, P., Goble, C., De Roure, D., Oinn, T.: Taverna workflows: Syntax and semantics. In: International Conference on e-Science and Grid Computing, pp. 441–448 (2007)
20. Korkhov, V., Vasyunin, D., Wibisono, A., Guevara-Masis, V., Belloum, A., de Laat, C., Adriaans, P., Hertzberger, L.: Ws-vlam: towards a scalable workflow system on the grid. In: WORKS 2007: Proceedings of the 2nd workshop on Workflows in Support of Large-scale Science, pp. 63–68. ACM, New York (2007)
21. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience. Special Issue on Scientific Workflows (2005)
22. Altintas, I., Barney, O., Jaeger-Frank, E.: Provenance collection support in the kepler scientific workflow system. In: Moreau, L., Foster, I. (eds.) IPAW 2006. LNCS, vol. 4145, pp. 118–132. Springer, Heidelberg (2006)

23. Davidson, S.B., Boulakia, S.C., Eyal, A., Ludäscher, B., McPhillips, T.M., Bowers, S., Anand, M.K., Freire, J.: Provenance in scientific workflow systems. IEEE Data Eng. Bull. 30, 44–50 (2007)
24. Bowers, S., Mcphillips, T., Riddle, S., Anand, M.K., Ludäscher, B.: Kepler/ppod: Scientific workflow and provenance support for assembling the tree of life. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 70–77. Springer, Heidelberg (2008)
25. Zhao, J., Goble, C., Stevens, R., Turi, D.: Mining taverna's semantic web of provenance. Concurrency and Computation: Practice and Experience, Special Issue on The First Provenance Challenge 20, 463–472 (2007)
26. Scheidegger, C.E., Vo, H.T., Koop, D., Freire, J., Silva, C.T.: Querying and re-using workflows with vstrails. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1251–1254. ACM, New York (2008)
27. Anand, M.K., Bowers, S., McPhillips, T., Ludäscher, B.: Efficient provenance storage over nested data collections. In: EDBT 2009: Proceedings of the 12th International Conference on Extending Database Technology, pp. 958–969. ACM, New York (2009)
28. Malawski, M., Bartynski, T., Bubak, M.: Invocation of operations from script-based grid applications. Future Generation Computer Systems 26, 138–146 (2010)
29. De Roure, D., Goble, C.: Research objects for data intensive research. In: E-Science (2009)