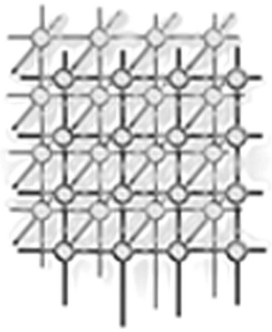
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2000; **00**:1–7 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Addressing the Provenance Challenge using ZOOM



Sarah Cohen-Boulakia*,†, Olivier Biton,
Shirley Cohen, and Susan Davidson

Department of Computer and Information Science,
University of Pennsylvania,
Levine Hall, 3330 Walnut St,
Philadelphia, PA-19104, USA

SUMMARY

ZOOM*UserViews presents a model of provenance for scientific workflows that is simple, generic, and yet sufficiently expressive to answer questions of data and step provenance that have been encountered in a large variety of scientific case studies. In addition, ZOOM builds on the concept of composite step-classes – or sub-workflows – which is present in many scientific workflow systems to develop a notion of *user views*. This paper discusses the design and implementation of ZOOM in the context of the queries posed by the provenance challenge, and shows how user views affect the level of granularity at which provenance information can be seen and reasoned about.

KEY WORDS: User views; Multi-level of granularity for provenance; Scientific workflows

1. Introduction

Scientific workflow management systems (e.g. [17], [15], [10], [9]) have become increasingly popular as a way of specifying and implementing large-scale in-silico experiments. Such systems differ from business-oriented workflow systems in the focus on data – e.g. sequences, phylogenetic trees, and proteins – and its transformation into hypotheses and knowledge [18]. In most scientific workflow systems, a workflow can be graphically designed by chaining together bioinformatics *tasks* (e.g. downloading sequences, aligning them). Tasks can also be grouped together to form *composite* tasks. Composite tasks are an important mechanism for abstraction, privacy, and reuse between workflows.

†E-mail: sarahcb@seas.upenn.edu
Contract/grant sponsor: NSF grants. (Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.); contract/grant number: Grants No. 0513778, 0415810, and 0612177.

Received 15 November 2006
Revised

Due to the large amount of intermediate and final data products generated in scientific workflows and the need for reproducible results, provenance has become of paramount importance, as evidenced by recent specialized workshops and surveys [18] on the subject. It has also received attention within the database community, and several models of reasoning (e.g. "why" and "where") over algebraic query expressions have been proposed [5, 6, 20]. However, tracking provenance in scientific workflows raises new and different challenges. First, since the operators in workflows are complex bioinformatic tasks, they are "black boxes"; fine grained algebraic reasoning cannot be performed. Second, composite tasks in scientific workflows may allow users to zoom into a box to see a sub-workflow. Composite tasks can therefore be used to vary the granularity at which provenance can be seen and reasoned about.

To address the problem of data provenance, most scientific workflow systems record metadata related to provenance [22] and/or develop a logging mechanism for tracking information and dependencies between data [4]. However, no generic model has been developed to define the meaning of provenance and take into account the composite structure of scientific workflows. In ZOOM*UserViews (ZOOM, for short), we develop a simple model that is sufficiently expressive to answer queries of provenance encountered in case studies, and which develops a notion of user views based on composite tasks.

In this paper, we discuss the design and implementation of ZOOM in the context of the queries posed by the provenance challenge (section 2). Section 3 demonstrates how ZOOM user views can be applied to a variation of the provenance challenge workflow to reason about provenance at different levels of abstraction.

2. Modeling the challenge workflow and answering queries in ZOOM

Our prototype is implemented in Oracle 10.g using an object layer together with a user interface (Java/JDBC component). In the following, we describe how we stored simulated traces of the challenge workflow run in our model. An ER diagram of the relational data model is available at the challenge web site [1].

We distinguish between the (static) specification of a workflow and its (dynamic) execution. A workflow specification consists of step-classes that represent the bioinformatics tasks to be performed, as well as data and control flow between those classes (which we will ignore for now). In contrast, an execution of a workflow is a partial order of steps (also called events), each of which is an instance of a step-class and has a set of input and output data objects. As an example, in the challenge workflow the "reslice" step-class has four instances which are the steps "5.reslice" ... "8.reslice". We make the assumption that in each workflow, each relevant object for provenance (e.g. step-classes, steps, and data items) has a unique identifier.

Our model stores both the specification and the executions of a workflow, as well as the relationship between those two. Workflow execution information can be extracted from the log of a workflow run. In particular, *input(step, dataId, ts)* gives information about the input to each step as well as the timestamp of input (read) events; *output(step, dataId, ts)* stores the same information about output (write) events. For instance, *input(2,3,8/8/2006)*, *input(2,4,8/8/2006)*, *input(2,9,8/8/2006)*, and *input(2,10,8/8/2006)* state that step 2 (2.align_warp) took as input data whose ids are 3 (Anatomy Image2), 4 (Anatomy Header2),

9 (Reference Image), and 10 (Reference Header). Each input in the example was taken on the same date, 8/8/2006*. Also, step 10 (10.slicer) produced one output with a dataId of 25 (Atlas X Slice) on 08/17/2006 (*output(10,25,08/17/2006)*). Details of which step-class a step execution is an instance of and the time at which it began are stored in *instanceOf (step, step-class, ts)*. For example, steps 1 to 4 are executions of the step-class "align_warp". Note that the integer ids of steps do not indicate precedence; in particular, steps 1-4 are executed in parallel.

Metadata and annotations on data and parameters are also stored. For example, table *data(dataId, name, type)* relates the identifier of a data item to its name and its type, e.g. *data(1,Anatomy Image1,Anatomy Image)* states that data with id 1 is named "Anatomy Image1" and has a type "Anatomy Image". Annotations are stored as attribute-value pairs associated with data items in table *dataAttributes*, e.g. *dataAttributes(1, center, UChicago)* states that the attribute "center" has value "UChicago" for the data item whose id is 1. The association between each step and its parameters as well as annotations of these parameters are stored using table *stepParam*; for example, it is possible to record that the step whose id is 1 takes a nonlinear (linear=false), 12th order (order=12) parameter, whose model is 1365 (model=1365).

To answer the challenge queries, we also introduced table *stageInstance(step, stage)* which gives all the steps that ran for a given stage (e.g., steps 1 to 4 are in stage 1). This table is the only one dedicated to the challenge; all the other tables are part of the generic ZOOM model.

To simplify the provenance queries, we created a view *Process*, which represents details of each execution including the steps, their corresponding step-class, inputs, outputs, and starting time. For example, *Process(13, convert, 25, Atlas X Slice, 28, Atlas X Graphic, 8/20/2006)* states that step 13 is an execution of the step-class convert, takes as input data with id 25 (Atlas X Slice), was executed on 8/20/2006, and produces as output data with id 28 (Atlas X Graphic). Note that if a given step takes several inputs and/or produces several outputs then the *Process* view will contain several rows having the same step id. Output data depends only on data read by the step before the write event occurs. For example, if O is a data item produced by step S at time $t2$, then O depends only on data items i such that *input(S,i,t1)*, *output(S,O,t2)* and $t1 \leq t2$.

Using these tables, we expressed the challenge queries. Code for all queries is available at the challenge web site [1]; most of the queries use an extension of the SQL-Oracle transitive closure operator that will be explained shortly.

Query 1 can be expressed as follows; the result is visualized as shown in Figure 1:

```
SELECT DISTINCT step, step-class, input, output, MARKDATA(step, input)
FROM Process
    START WITH output = (SELECT dataId FROM data WHERE name = 'Atlas X Graphic')
    CONNECT BY PRIOR input = output AND ISMARKED(step, input)=0
ORDER BY step;
```

*For conciseness, we use dates without minutes and seconds in our simulation of the workflow; we could also have used full timestamps.

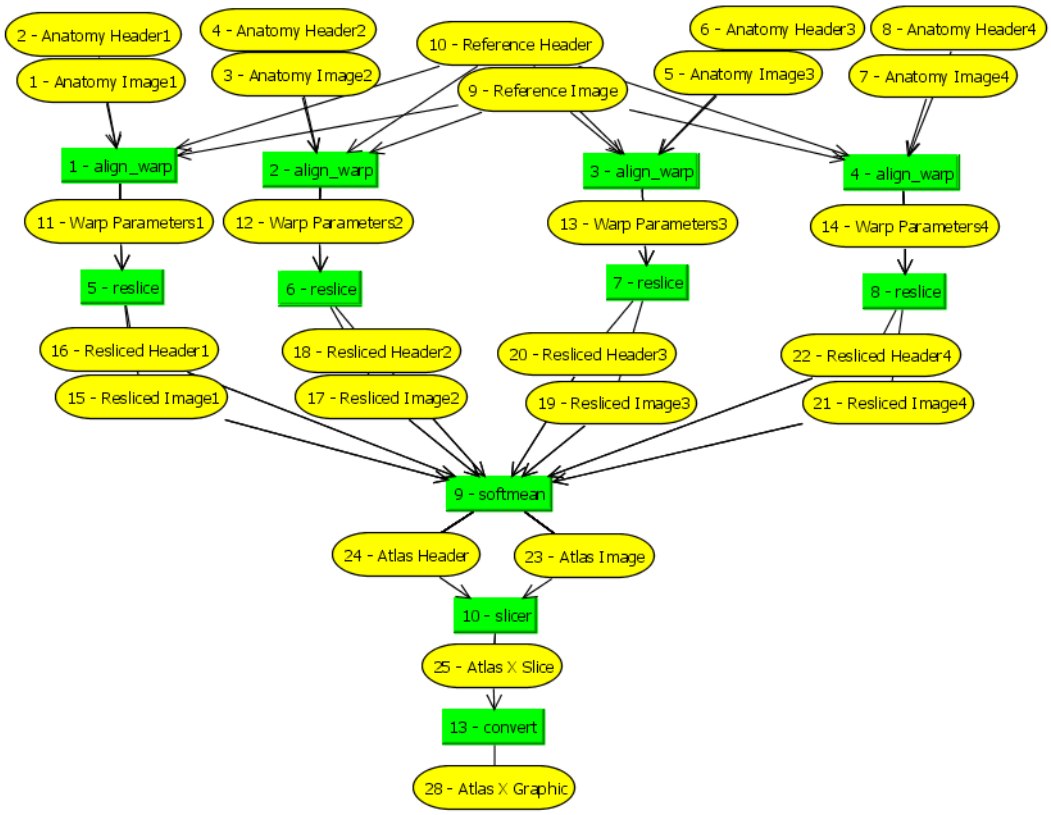


Figure 1. Results obtained for Query 1: the resulting provenance information is shown as a graph. Squared shapes represent steps; rounded shapes represent data.

This SQL query focuses on the *Process* table. The subquery first searches for the id of the data item named "Atlas X Graphic" (Where clause). This id is then the entry point (Start With) of the main recursive query that will search for the data used as input to produce it. These inputs are then considered as outputs for which input are searched (CONNECT BY PRIOR input = output). The recursive procedure ends when the data id considered is a user input (i.e. a data that was not produced using any other data).

Since the transitive closure operator, CONNECT BY PRIOR, provided by Oracle is developed for hierarchies and does not work for arbitrary DAGs, we extended this operator by creating two stored procedures: MARKDATA to mark each node seen in the recursion, and ISMARKED to stop the recursion if the node has already been seen (e.g. two steps with a common step ancestor).

Query 2 is similar, but uses MINUS to restrict provenance information to occur after given step (softmean); Query 3 makes use of the *stageInstance* table to focus on stages 3, 4 and 5 of the process; query 4 uses a join between *Process* and *stepParam*, while queries 5, 8, and 9 use joins between *Process* and *dataAttributes*.

The semantics of Query 7, which asks for the difference between two workflow executions, is a topic for future research. For the challenge workflow run in which only one step-class has been modified and the workflow specification appears linear, the query can be answered by simply using MINUS. However, it is not clear what the question means absent a workflow specification and in the presence of cycles in the specification.

3. User views

The main contribution of the ZOOM*UserViews model is to develop a notion of user views based on composite step-classes. There are several reasons why composite step-classes are useful in workflows. First, they can be used to hide complexity and allow users to focus on a higher level of abstraction. For example, users may wish to focus on biologically relevant step-classes and hide step-classes which focus on formatting. Second, composite step-classes can represent authorization levels; users without the appropriate clearance level would not be allowed to see the details of a composite step-class.

A user view is then defined as follows [8]: *Given a workflow specification S, a user view U is a partition of atomic step-classes in S.* Note that two composite step-classes (partitions) in U cannot overlap in its member atomic step-classes, and that each atomic step-class in S must be a member of some composite step-class in U.

As an example, consider Figure 2 in which three user views are indicated by placing boxes around portions of the workflow execution. User view {Box3} (which we will call "UBlackBox") sees the entire workflow as one composite step. User view {Box1, softmean, Box2} (which we will call "uBio") hides details of the align_warp and reslice steps (Box1) as well as slicer and convert (Box2); however step-class softmean can be observed. In the lowest level user view, {align_warp, reslice, softmean, slicer convert}, all step-classes can be observed; we call this user view "uAdmin".

User views affect what data a user can see, as well as the provenance of visible data. For example, data item 15 (Resliced Image 1) cannot be seen in uBlackBox since it is hidden in Box3. In uBio, the immediate data provenance of 15 will be 1 (Anatomy Image1) and immediate step provenance will be stepBox1-1 (a composite execution of Box1 induced by steps 1.align_warp and 5.reslice). In uAdmin, the immediate data provenance of 15 will be 11 (Warp Params1) and immediate step provenance 5.reslice.

To model user views, we must address their definition at the the specification level as well as the implications for provenance at the execution level. At the specification level, table *immContains* captures the immediate containment of step-classes; view *contains* is the closure of *immContains*. For example, *immContains(box1,align_warp)* and *immContains(box3,box1)* express the facts that composite step-class "box1" includes step-class align_warp, and composite step-class "box3" contains composite step-class "box1". Table *userView(usr, stepClass)* indicates the lowest step-classes that a user can see. For the example in Figure 2, we get the following:

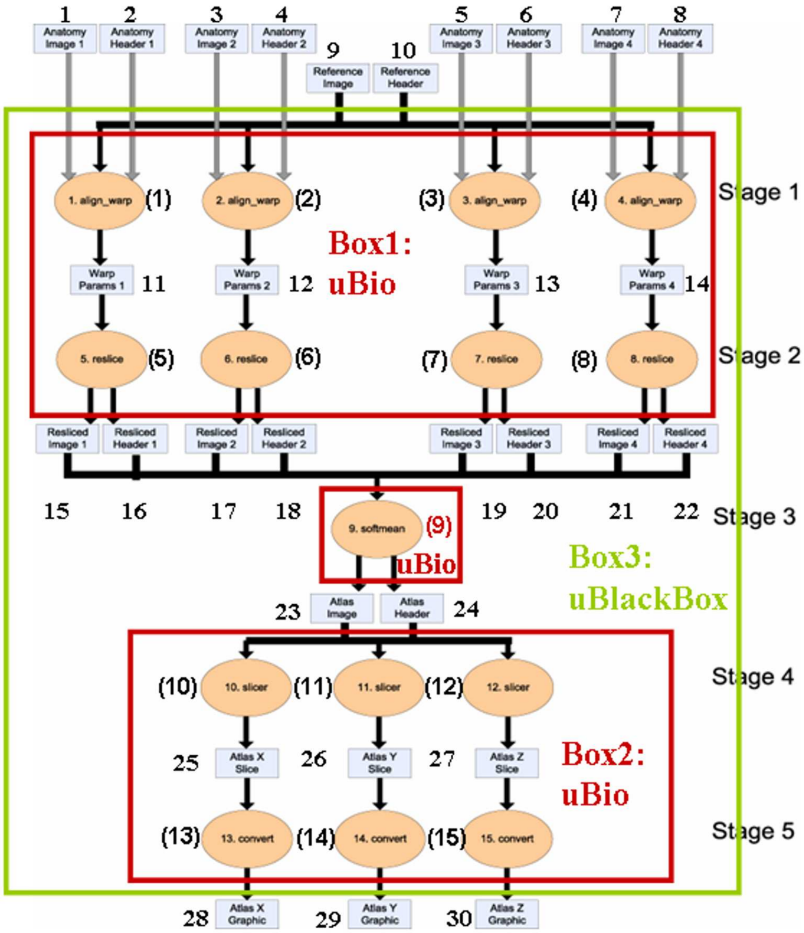


Figure 2. Variation of the challenge workflow with user views uBlackBox and UBio indicated. For ease of readability, steps ids have been indicated under brackets.

USR	STEPCLASS	USR	STEPCLASS
uBlackBox	box3	uAdmin	align_warp
uBio	box1	uAdmin	reslice
uBio	softmean	uAdmin	reslice
uBio	box2	uAdmin	softmean
uAdmin	convert	uAdmin	slicer

At the execution level, composite steps are induced for composite step-classes in each user view and input and output relations are calculated for composite steps. The input to a composite step is any input to a member step which is not the output of another member step; the output of a composite step is any output of a member step which is not input to another member step. Thus *cInput* (*cOutput*) stores information about inputs (outputs) to a composite step and the time at which the read (write) events occurred.

Intuitively, determining composite steps is done as follows: For any step S in the execution graph that is not yet in a composite step, create a composite step C (a new instance of the composite class of S) and add to C all steps S' such that S' has an edge to or from any node of C and has the same composite class of S. (The complete algorithm is beyond the scope of this paper.)

In our example, four composite steps would be created for Box1 (stepBox1-1 ... stepBox1-4), and three for Box2 (stepBox2-1 ... stepBox2-3). StepBox1-1 consists of steps 1 and 5; stepBox2-1 takes data id 23 (Atlas Image) and data id 24 (Atlas Header) as inputs (*cInput(stepBox2-1, 23, 8/17/2006)*, *cInput(stepBox2-1, 24, 8/17/2006)*) and produces data id 28 (Atlas Graphic X) on 08/20/2006 as output (*cOutput(stepBox2-1, 28, 8/20/2006)*).

Using *cInput* and *cOutput*, we define the view *uProcess* which gives details of each execution a user can see (analogous to *Process* in the previous section).

To illustrate the effect of user views on provenance, consider the following query: "What is the step and data provenance of *Resliced Image1* (id 15)?" This can be expressed as follows, where $userv can be replaced by any user view in the system:

```
SELECT upc.*, MARKDATA(step, input)
FROM uProcess  upc
WHERE usr = $userv
     START WITH outputName = 'Resliced Image1'
     CONNECT BY PRIOR upc.input = upc.output
         AND ISMARKED(step, input)=0;
```

First, consider the case where $userv="uAdmin". The answer obtained are those obtained without the benefit of user views (see Figure 3 (A)).

Now, consider the case where $userv="uBio". The following result is obtained, visualized in Figure 3 (B).

USR	STEP	STEPCLASS	INPUT	INPUTNAME	OUTPUT	OUTPUTNAME	TIME
uBio	stepBox1-1	box1	1	Anatomy Image1	15	Resliced Image1	8/7/2006
uBio	stepBox1-1	box1	10	Reference Header	15	Resliced Image1	8/7/2006
uBio	stepBox1-1	box1	2	Anatomy Header1	15	Resliced Image1	8/7/2006
uBio	stepBox1-1	box1	9	Reference Image	15	Resliced Image1	8/7/2006

Note that uAdmin can see that data id 11 (Warp Params 1) is part of the provenance while uBio cannot see it.

Finally, when $userv="uBlackBox" the result relation is empty; the user is not entitled to see any information about data that is not in his user view.

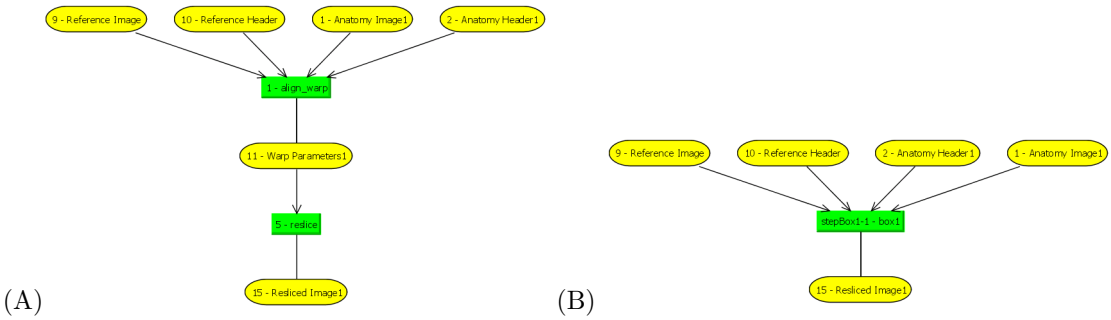


Figure 3. Results obtained for the query considering uAdmin (A) and uBio (B) (also available at the challenge web site [1]).

4. Discussion and Conclusion

This paper discusses how ZOOM addresses the Provenance challenge queries, and introduces a variation of the workflow and provenance query to illustrate user views.

The ZOOM model captures all necessary information for reasoning about provenance generated by the workflow systems which participated in the Provenance Challenge, e.g., Kepler (teams COMAD [3] and RWS [14]), Taverna (myGrid team [21]), Pegasus/Wings [12], Redux [2], Karma [19], and Chimera (team UChicago/VDL [7]). Systems like ES3 [11] and JP [13], provide approaches based on operating systems, and can record provenance at very low levels (OS-/job-oriented). Since these systems provide a log with input/output/timestamps, our model is able to take into account the provenance information they generate.

The user view approach followed in this paper can be exploited by any system which allows the user to compose tasks, as done in Kepler (*composite actors*) and Taverna/myGrid (*nested workflow processors/knowledge view* model [22]).

We summarize the key points of our system using the system matrix defined in the editorial [16].

Characteristics of the ZOOM provenance system: ZOOM provides a model of provenance for **scientific workflows**, which relies on minimal log information: start step events, input (read) events, output (write) events, and the **time** at which these events occurred. ZOOM is thus **technology independent**; no specific workflow system was used. However, the **simulated run** provided was used to illustrate the technique. Provenance is currently represented using a **relational** framework (RDBMS) and the query language is SQL extended with transitive closure. The prototype was implemented under Oracle 10.g and Java. The purpose of ZOOM is both to **store (S)** provenance information and to **query (Q)** it.

Properties of Provenance Representation in ZOOM: The causal graph illustrating the provenance of a data product is constructed using the base tables *input* and *output*, and

their relationship with other tables in the schema. We assume that **unique ids** are assigned to every provenance relevant object.

As for the queries: recursive queries expressing the transitive closure on inputs and outputs are used, and both the **flow of data and steps** (that is, both data and events) can be queried. It is worth noticing that although it is always possible to ask queries about **annotations** on data or on steps (e.g. type of parameters, metadata), these queries were not considered as core provenance queries in ZOOM.

Last but not least, ZOOM provides **user views** allowing provenance information to be calculated at different levels of granularity. Only visible and necessary data (and steps) are returned.

The model presented in this paper can manage much more complex workflows than the workflow of the first challenge, for example, workflows with cycles in their definition. In complex workflows, the benefit of user views is even more apparent: it allows the intricate relationships of data produced and consumed in cycles to be hidden, providing the user with a much simpler understanding of the execution. Part of our ongoing work on ZOOM is to demonstrate the benefit of user views in complex scientific workflows encountered in a wide variety real-world examples.

ACKNOWLEDGEMENTS

The authors wish to thank members of the Kepler group (particularly Bertram Ludäscher, Timothy McPhillips, and Shawn Bowers) for many fruitful discussions about scientific workflows.

REFERENCES

1. http://twiki.gridprovenance.org/bin/view/Challenge/UPenn.
2. R. S. Barga and L. A. Digiampietri. Automatic capture and efficient storage of escience experiment provenance. *Concurrency and Computation: Practice and Experience*, 2007.
3. S. Bowers, T. McPhillips, and B. Ludäscher. Provenance in collection-oriented scientific workflows. *Concurrency and Computation: Practice and Experience*, 2007.
4. S. Bowers, T. M. McPhillips, B. Ludäscher, S. Cohen, and S. B. Davidson. A model for user-oriented data provenance in pipelined scientific workflows. In *IPAW, International Provenance and Annotation Workshop*, pages 133–147, 2006.
5. P. Buneman, S. Khanna, and W. Tan. Why and where: A characterization of data provenance. In *International Conference on Database Theory (ICDT)*, pages 316–330, 2002.
6. L. Chiticariu, W. Chiew Tan, and G. Vijayvargiya. Dbnotes: a post-it system for relational databases based on provenance. In *SIGMOD Conference, International Conference on Management of Data*, pages 942–944, 2005.
7. B. Clifford, I. Foster, M. Hategan, T. Stef-Praun, M. Wilde, and Y. Zhao. Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*, 2007.
8. S. Cohen, S. Cohen-Boulakia, and S. Davidson. Towards a model of provenance and user views in scientific workflows. In *Data Integration in the Life Science*, volume 4075, pages 264–279. LNBI Springer-Verlag, 2006.
9. S. Cohen-Boulakia, S. Lair, N. Stransky, S. Graziani, F. Radvanyi, E. Barillot, and C. Froidevaux. Selecting biomedical data sources according to user preferences. *Bioinformatics*, 20:i86–i93, 2004.
10. I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. The virtual data grid: A new model and architecture for data-intensive collaboration. In *Conference on Innovative Data System Research (CIDR)*, 2003.
11. J. Frew, D. Metzger, and P. Slaughter. Automatic capture and reconstruction of computational provenance. *Concurrency and Computation: Practice and Experience*, 2007.

12. J. Kim, E. Deelman, Y. Gil, G. Mehta, and V. Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 2007.
13. A. Krenek, J. Sitera, L. Matyska, F. Dvorak, M. Mulac, M. Ruda, and Z. Salvet. glite job provenance – a job-centric view. *Concurrency and Computation: Practice and Experience*, 2007.
14. B. Ludäscher, N. Podhorszki, I. Altintas, S. Bowers, and T. McPhillips. Models of computation and provenance, and the rws approach. *Concurrency and Computation: Practice and Experience*, 2007.
15. T. M. McPhillips and S. Bowers. An approach for pipelining nested collections in scientific workflows. *SIGMOD Record*, 34(3):12–17, 2005.
16. L. Moreau, B. Ludäscher, R. S. Barga, L. Digiampietri J. Goldbeck Y. L. Simmhan, B. Plale, A. Krenek, D. Turi, C. Goble, R. Stevens, J. Zhao, J. Freire, J. Frew, D. Metzger, P. Slaughter, S. Cohen-Boulakia, S. Davidson, S. Cohen, S. Bowers, T. McPhillips, N. Podhorszki, I. Altintas, D. Holland, M. Seltzer E. Deelman, Y. Gil, J. Kim, G. Mehta, V. Ratnakar, J. Myers, J. Futrelle, S. Miles, S. Munroe, P. Groth, S. Jiang, K. Schuchardt, T. Gibson, E. Stephan, G. Chin, B. Clifford, M. Wilde, and I. Foster. The First Provenance Challenge. *Concurrency and Computation: Practice and Experience*, 2006.
17. T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.
18. Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. In *SIGMOD Record*, pages 31–36, 2005.
19. Y. L. Simmhan, B. Plale, and D. Gannon. Querying capabilities of the karma provenance framework. *Concurrency and Computation: Practice and Experience*, 2007.
20. J. Widom. Trio: A system for integrated management of data, accuracy, and lineage. In *CIDR, Conference on Innovative Data Systems Research*, pages 262–276, 2005.
21. J. Zhao, C. Goble, R. Stevens, and D. Turi. Mining taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 2007.
22. J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, and M. Greenwood. Using semantic web technologies for representing e-science provenance. In *Semantic Web Conference (ISWC2004)*, pages 92–106, 2004.