Future Generation Computer Systems 27 (2011) 737–742

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Special Section: The third provenance challenge on using the open provenance model for interoperability

Yogesh Simmhan[a,*], Paul Groth[b], Luc Moreau[c]

[a] University of Southern California, Computer Engineering, 3740 McClintock Avenue, Los Angeles, CA 90089, United States
[b] VU University Amsterdam, Department of Computer Science, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
[c] University of Southampton, School of Electronics & Computer Science, Southampton SO17 1BJ, United Kingdom

ARTICLE INFO

Article history:
Received 31 December 2009
Received in revised form
21 October 2010
Accepted 29 November 2010
Available online 17 December 2010

Keywords:
Provenance challenge
Editorial
Open provenance model

ABSTRACT

The third provenance challenge was organized to evaluate the efficacy of the Open Provenance Model (OPM) in representing and sharing provenance with the goal of improving the specification. A data loading scientific workflow that ingests data files into a relational database for the Pan-STARRS sky survey project was selected as a candidate for collecting provenance. Challenge participants record provenance, run queries over it, and export/import provenance as OPM documents with other teams to verify interoperability. Fourteen teams participated in the challenge that concluded at a workshop in June 2009 in Amsterdam. The experiences of several participating teams are included in this special issue. In this editorial, we describe the challenge in detail, review its outcome, and introduce articles included in this special issue.

1. Introduction

Provenance has been described as the derivation history of data [1], documentation of processes that affect a digital object [2], the origin, context or pedigree of data [3,4], or just the origin of something [2]. It is represented variously as annotations and acyclic graphs, mapped to XML/RDF metadata and relational tuples. There have been several surveys, documented in [2], that have shown the importance of provenance to the quality of scientific data derived from workflows [1], reliability of web pages and documents [3], to show why, where and how a data came to be in databases [5], and for reproducibility of computational tasks [4]. The importance of recording provenance is inarguable for many applications, as is the pressing need to be able to share it across organizational and technological boundaries instead of having its usefulness curtailed within silos.

Different aspects of provenance have been increasingly investigated since the Data Derivation and Provenance workshop in 2002 [6] – one of the first workshops dealing with data provenance – paved the way for the International Provenance and Annotation Workshop (IPAW) series [7], currently in its third installment. The provenance challenge [8] was conceived at the first IPAW workshop as a means to build consensus around what provenance means and to identify common ways to represent, collect, share and query it across provenance systems. The experience gained from the first two provenance challenges in 2006 and 2007 helped evolve a preliminary specification for modeling and sharing provenance documents. This draft Open Provenance Model (OPM) v1.0 specification [9] introduced three entities: digital or physical *artifacts, processes* that operate on artifacts, and *agents* that control processes, and identified causal relationships between them. The draft specification was discussed at the second IPAW workshop and community feedback incorporated into v1.01 [10]. The need was felt to test the efficacy of the updated specification in representing causality of real world applications and, importantly, its usefulness for interoperating across systems. Other peripheral goals were to build momentum around the proposed standard, integrate support for OPM within existing provenance systems, and spawn novel OPM-based tools for the community. These formed the basis for organizing the third provenance challenge discussed below. One key outcome of the collaborative challenge has been the release of the Open Provenance Model Core Specification v1.1 [11], which is part of this special issue, and a community process to update and maintain it. In this editorial, we (1) specify the goals and describe the problem used for the third provenance challenge, (2) review the participation and outcome of the challenge for various teams, (3) summarize progress on the OPM specification as part of the challenge workshop, and (4) introduce articles in this special issue that originated from both the challenge and other subsequent efforts on interoperability using the Open Provenance Model.

[*] Corresponding author.
E-mail address: simmhan@usc.edu (Y. Simmhan).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.11.020

2. The provenance challenge

Definition of the third provenance challenge (PC3) was initiated by soliciting proposals for workflows that are used in real world applications while also being suitable for flexing the different features of the Open Provenance Model. Groups were invited to submit a candidate workflow with English and graphical descriptions, and highlight its novelty in going beyond the fMRI workflow used for prior challenges. They were also to specify the availability of component tasks in the workflow that other teams could reuse or implement, and the reference input and intermediate data for the workflow.

Ten workflow submissions were received for consideration. These workflows spanned across domains such as *bioinformatics* (Paolo Missier, University of Manchester), *brain imaging* (Julian Freire and Erik Anderson, University of Utah), *astronomy* (Yogesh Simmhan and Roger Barga, Microsoft Research), *oceanography* (SatyaSahoo, Wright State University; Roger Barga, Microsoft Research), and *software build system* (Luc Moreau, University of Southampton; Paul Groth, Information Sciences Institute). The candidate workflows were submitted by October 2008 and discussed during a session at the IEEE eScience conference in December 10–12, 2008 at Indianapolis.From them, the astronomy database loading workflow was chosen through community participation and further refined. The challenge definition, sample code and data, and provenance queries were published on the PC3 wiki site [12] and the challenge opened for participation on March 2, 2009. Teams participating in the challenge catalogued their workflow graphs, OPM provenance documents, and provenance query results in the PC3 wiki, and presented them at the Third Provenance Challenge workshop on June 10–11, 2009 in Amsterdam.

2.1. Guidelines for the challenge

The challenge had the following specific goals it expected to accomplish:

1. Identify strengths and weaknesses of the Open Provenance Model (OPM) v1.01 draft specification to enable a final specification to be produced.
2. Encourage the development of concrete bindings for OPM in various languages and platforms used by the challenge participants.
3. Determine the expressivity of OPM for different *process* and *artifact* technologies, such as workflows, SQL, files and databases.
4. Verify the efficacy of OPM to enable interoperability across heterogeneous applications, providing the ability to reconstruct and interpret provenance documents produced by them.
5. Finally, continue with the process of community building to further discuss interoperability mechanisms for provenance systems.

The challenge involved collecting provenance for a *workflow*, a series of tasks that take input data and produce output data. In earlier challenges, the tasks and data were agnostic to technology and were pure data flows. However, given the goals of this challenge to try different technologies, an effort was made to incorporate data present as text files, relational databases, relational tables, tuples, and collections. Similarly, the workflow tasks included applications, file operations, and relational database queries. In addition, the workflow incorporated control flow statements, such as iterations and conditional branching, to reflect more complex constructs than earlier challenges.

The workflow was described both in text and graphically, and accompanied by source code for the workflow tasks. Tasks were available in C# and in Java, and SQL database tasks could run on Microsoft SQL Server and Apache Derby databases. This enabled running the workflow on diverse OS platforms. Participants were invited to map the workflow, optionally using the implementations provided, to a workflow system of their choice (or to simulate its execution) in order to generate provenance for the workflow. The provenance collected was shared as an Open Provenance Model document. Since a serialization of OPM was not part of the specification, participants could either export the OPM document in sample XML and RDF serializations that were available, or in a serialization of their choosing.

The teams were expected to run several provenance queries that were specified on the provenance that they collected from their workflow, as well as the provenance that they imported from other teams as OPM documents. The queries tested the expressivity of the OPM specification for the given workflow and also identified (potential) information loss when exporting and importing OPM documents between provenance systems due to vagueness or shortcomings in the specification. There were two sets of queries: *core queries* that were required to be attempted by all teams and *optional queries* that could be suggested by the participants to highlight specific features or pitfalls.

Participants were asked to share their workflows, exported OPM graphs, and results of running the queries on provenance that they collected and OPM documents they imported on the PC3 wiki site [12] and present their results at the challenge workshop.

2.2. The Pan-STARRS load workflow

Panoramic Sky Survey and Rapid Response System (Pan-STARRS) [13] is a next generation sky survey project that continuously scans the visible sky once a week and builds a time series of data that catalogs the solar system and can detect Earth impacting objects. Pan-STARRS uses several workflows for its data loading pipeline [14] that convert raw image data from the digital telescope into queryable observations and metadata that are stored in relational databases. One of these data loading workflows was selected and adapted for the provenance challenge. Fig. 1 shows the data and control flows in the Pan-STARRS *load workflow* used in the challenge.

The *load workflow* imports observations of space objects arriving in the form of Comma Separated Value (CSV) files into tables in a relational database. There are two inputs to the workflow: (1) the *directory location* of a set of related CSV files from a single telescope image that need to be loaded into a new database, and (2) a *JobID* that indicates the name of the database to be created to hold the data. The input directory contains a "CSV Ready" *manifest* file, listing the CSV files to be loaded, and three CSV files – *detection, frame meta*, and *image meta* – that each correspond to a table in the database. The workflow then performs several tasks that verify the existence of the input CSV files, create a database to load the files, iterate through each input CSV file to validate and load it into a distinct table, update computed columns in the table, perform post load validations, and finally compact the database once all files are successfully loaded. Tasks that fail to validate input data cause the workflow to halt. The load workflow tasks are listed in Table 1.

In Fig. 1, solid lines denote the data passed from one task to another. These may be string paths and URIs that reference files, tables, and databases, or simple parameters used by the task. The C# and Java implementations use class objects and collections to represent some of these parameters, though all of them can be formatted as strings, basic types and their collections. The workflow itself can be represented as a pure SQL script as many of the C# or Java tasks just wrap SQL update statements. The workflow also incorporates control flows in the form of conditions and iterations, represented by dotted lines, which allow for different execution paths to be taken.

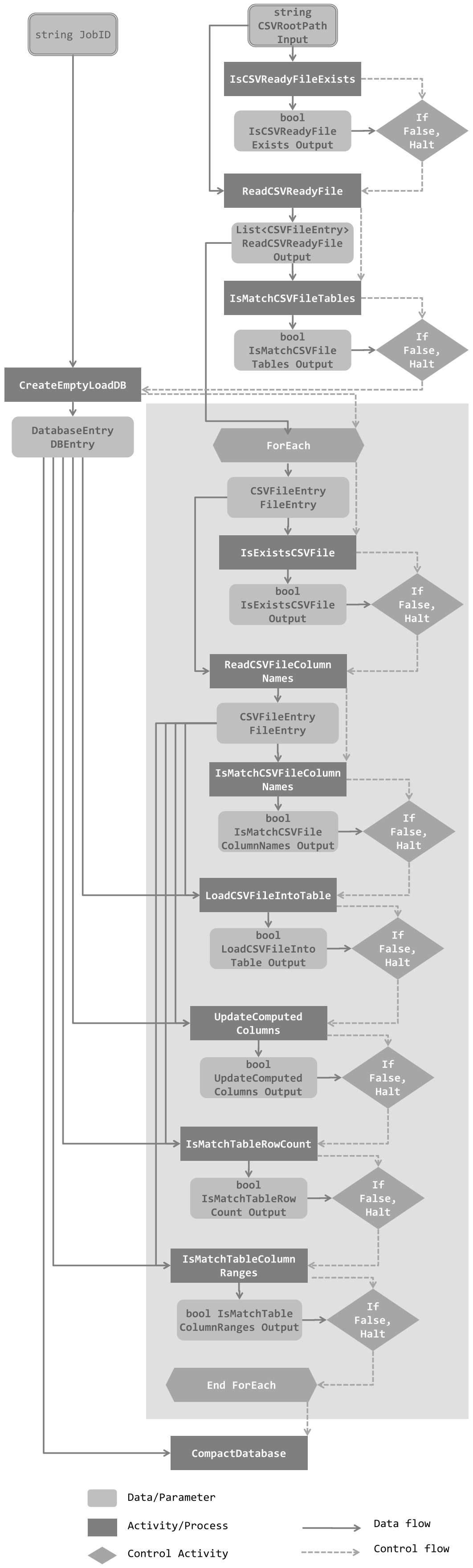

Fig. 1. Pan-STARRS load workflow used in the provenance challenge.

2.3. Provenance challenge core queries

Queries over the provenance graphs help to evaluate the expressivity of the provenance collected. The query results also validate the accuracy of provenance that is collected by a team by their own workflow implementation and the OPM documents imported from other teams. The three core queries defined for this challenge were:

Q1. For a given detection, which CSV files contributed to it?

This should return the CSV file containing the *Detection* table for a simple answer. A more detailed analysis of the provenance returns the CSV file containing the *Detection* table, CSV file containing the *Image* table (as the image is an attribute of the detection), and CSV file containing the *FrameMetadata* table (as the frame metadata is an attribute of the image).

Q2. The user considers a table to contain values they do not expect. Was the range check (IsMatchTableColumnRanges) performed for this table?

This must return 'Yes' if the range check task was performed before resulting in the values. 'No' if it was not performed.

Q3. Which operation executions were strictly necessary for the Image table to contain a particular (non-computed) value?

This query would return the execution of *ReadCSVReadyFile* and *CreateEmptyLoadDB*, and the invocation of *ReadCSVFileColumnNames* and *LoadCSVFileIntoTable* during the second iteration of the *forloop*—because *Image* is loaded in the second iteration. The results exclude the invocation of the various condition checks since they do not change the data, the invocation of *UpdatedComputedColumns* because the *Image* table data has no derived columns to update, and the invocation of *CompactDatabase* because it does not affect the value.

Besides the three core queries, there were 13 other optional queries that the teams could attempt. These are omitted for brevity and can be found in the provenance challenge wiki [12].

3. Summary of team contributions

The provenance challenge saw the participation of fourteen teams [15] at the time of the provenance challenge workshop in Amsterdam. Their results are posted on the challenge wiki website [12] and the details of the teams, their approaches to provenance storage and modeling, and the results of their interoperability are summarized in Table 2. From the outset, the provenance challenge series was intended to be *informative* rather than *competitive*; so there are no specific victors of the challenge.

There were several interesting approaches taken by the teams to address challenges posed by the interoperability problem. One was in dealing with control-flows in the workflow, which was not seen in the previous challenges. Some teams modeled the control-flow activities as processes that trigger subsequent processes, while others modeled them as explicit data-flows that pass the boolean value. Teams that used pure dataflow-based systems did not even support tracking of control-flows. Many teams also unrolled the iterations in the workflow, optionally keeping track of the iteration step count. Validation errors (e.g. *IsMatchtableCount*) were also handled differently, with certain teams able to halt workflow execution as expected while others let them proceed but use an error flag to suppress the application logic for subsequent workflow activities from executing. All of these techniques have a corresponding impact on the provenance collected and its interoperability.

Teams also encountered difficulties due to different approaches taken for recording process and artifact identifiers in OPM since the model does not prescribe any naming scheme. Also, OPM makes a distinction between the artifact instance at the moment it is

Table 1
Description of tasks used in the workflow, with their inputs and outputs.

Workflow task	Input data	Output data
Pre-Load Tasks		
IsCSVReadyFileExists: Checks for existence of CSV root directory & manifest file.	*string* **CSVRootPathInput**: Path to root directory of CSV batch	*bool* **IsCSVReadyFileExistsOutput**: Returns true if the CSV root directory and csv_ready.csv manifest file it contains exists in the file system. False otherwise.
ReadCSVReadyFile: Reads the contents of the csv_ready.csv manifest file and creates a *CSVFileEntry* to hold metadata on each CSV file listed in it.	*string* **CSVRootPathInput**: Path to root directory of CSV batch	*List* ⟨*CSVFileEntry*⟩ **ReadCSVReadyFileOutput**: List of *CSVFileEntry*s read from the manifest file. Each contains the FilePath of a CSV file to load, the HeaderPath to a header file listing data columns, the RowCount of rows in the file, the TargetTable name to load into, and the Checksum for the file.
IsMatchCSVFileTables: Checks if all tables to be loaded have corresponding CSV data files in the manifest.	*List* ⟨*CSVFileEntry*⟩ **FileEntriesInput**: List of *CSVFileEntry*s read from the manifest file.	*bool* **IsMatchCSVFileTablesOutput**: Returns true if all tables have matching CSV files. False otherwise.
IsExistsCSVFile: Checks for existence of CSV data file and Header file listed in the manifest.	*CSVFileEntry* **FileEntryInput**: A *CSVFileEntry* read from the manifest file.	*bool* **IsExistsCSVFileOutput**: Returns true if the CSV data file and Header files exist in file system. False otherwise.
ReadCSVFileColumnNames: Reads list of column names present in the CSV data file from the Header file.	*CSVFileEntry* **FileEntryInput**: A *CSVFileEntry* read from the manifest file.	*CSVFileEntry* **FileEntryOutput**: The input *CSVFileEntry* updated with the ColumnsNames field populated from the Header file.
IsMatchCSVFileColumnNames: Checks if all columns expected for a target table are present in the corresponding CSV data file.	*CSVFileEntry* **FileEntryInput**: A *CSVFileEntry* read from the manifest file with column names populated.	*bool* **IsMatchCSVFileColumnNamesOutput**: Returns true if column names present in CSV data files match expected column names for the target table to load the CSV file into. False otherwise.
Load Tasks		
CreateEmptyLoadDB: Creates an empty 'Load' database with empty tables to load.	*string* **JobID**: Unique identifier for the CSV batch being loaded.	*DatabaseEntry* **CreateEmptyLoadDBOutput**: A *DatabaseEntry* with the DBName, unique DBGuid and ConnectionString for the created database.
LoadCSVFileIntoTable: Load a single CSV data file into corresponding table in the Load database.	*DatabaseEntry* **DBEntry**: A *DatabaseEntry* with the target table to load the CSV data file. *CSVFileEntry* **FileEntry**: A *CSVFileEntry* to load into the corresponding target table in the database.	*bool* **LoadCSVFileIntoTableOutput**: Returns true if the CSV data file was successfully loaded into the target table. False otherwise.
UpdateComputedColumns: Updates the computed columns in the target table that was loaded. These derived columns would have been "empty" (-999) in the CSV data file.	*DatabaseEntry* **DBEntry**: A *DatabaseEntry* with the target table already loaded from the CSV file. *CSVFileEntry* **FileEntry**: A *CSVFileEntry* containing the name of target table to update in database.	*bool* **UpdateComputedColumnsOutput**: Returns true if the derived columns were successfully updated from existing columns. False otherwise.
Post-Load Tasks		
IsMatchTableRowCount: Checks if number of rows loaded into table matches expected number of rows in CSV data file.	*DatabaseEntry* **DBEntry**: A *DatabaseEntry* with the target table loaded and updated. *CSVFileEntry* **FileEntry**: A *CSVFileEntry* containing the expected number of rows in the CSV data file and the target table name.	*bool* **IsMatchTableRowCountOutput**: Returns true if the number of rows in the target table equals the expected number of rows in the CSV data file.
IsMatchTableColumnRanges: Checks if the data loaded into table columns fall within the range of values expected for the column.	*DatabaseEntry* **DBEntry**: A *DatabaseEntry* with the target table loaded and updated. *CSVFileEntry* **FileEntry**: A *CSVFileEntry* containing the name of target table in the database to validate column ranges.	*bool* **IsMatchTableColumnRangesOutput**: Returns true if the data values of columns in the target table fall within expected range. False otherwise.
CompactDatabase: Shrinks the database after all write operations complete.	*DatabaseEntry* **DBEntry**: A *DatabaseEntry* with all tables loaded and validated.	*void*

used or generated, and the logical or physical identifier for the actual artifact, which caused some challenge queries to require additional examination. Naming was particularly a problem for database artifacts like tables and tuples since several teams did not support tracking of database artifacts. Different teams were able to track database and file entities at various levels of granularities.

Challenge queries that combined provenance information with additional metadata or data values also posed difficulties for teams. Additional *ad hoc*annotations were used by certain teams to capture part of the metadata within the OPM model itself, with particular teams even encoding the entire artifact value as part of the provenance annotation.

Given that this was the first time the OPM specification was put to use, the initial interoperability challenge for teams was in mapping from their native provenance model to OPM. Most teams were successful in achieving this to answer the core queries. A number of teams also attempted, with different degrees of success, to interoperate with the OPM generated by others. More detailed experiences of some of these teams in addressing the challenge and using the Open Provenance Model are discussed in articles in this special issue.

4. Progress on the OPM specification

The Open Provenance Model OPM v1.00, originally crafted in a meeting held in Salt Lake City in August 2007, was released to the community in December 2007 [9]. The first OPM workshop in June 2008 involved some twenty participants discussing issues related to this specification, and led to a revised specification, referred to as OPM v1.01 [10].

Table 2
Details of teams participating in the challenge.

Team	Workflow/provenance system	Native provenance format	OPM binding format	Native provenance storage	PC3 query format	Teams interoperated with *(Query success status)*
NCSA	Java/Tupelo	RDF	XML, RDF		Java API	NCSA*(success)*
Univ of Chicago	Swift	Relational	XML	DBMS		UChicago *(partial)*
Microsoft Research	Trident	Relational	XML	DBMS/MS SQL server	SQL	MSR *(success)*, Soton *(fail)*, UoM *(fail)*, UCD *(partial)*, IU *(success)*, UUtah *(fail)*
UC-Davis	COMAD-Kepler	Relational	XML	DBMS/MySQL	SQL	UCD *(success)*, Soton, RPI, KCL, Harvard
Univ of Soton, USC-ISI	Java	XML	XML, RDF	File	Java API	Soton *(success)*
Univ of Manchester	Taverna	RDF	XML, RDF		SPARQL	UoM, UCD, Soton, NCSA
RPI/Tetherless	Java/ProtoProv	RDF	XML	Jena	SPARQL	RPI *(success)*
UvA/VL-e	WS-VLAM/ PLIER	Relational	XML	DBMS	SQL	UvA *(success)*
SDSC	Kepler		XML		XQuery	SDSC *(success)*
Univ of Utah	VisTrails, Python	XML	XML		XQuery	UUtah *(partial)*, SDSC *(partial)*
Kings College, London	Java		XML			KCL *(success)*
Harvard Univ	Bash, Java/PASS	Path Query Language Graphs	XML	PQL DB Store	PQL	Harvard *(success)*
Indiana Univ	ODE/Karma	Relational	XML	DBMS/MySQL	SQL	IU *(success)*
UTEP	WDO-It!	Proof Markup Language		File		

From the outset, the original authors' intent has been to define a data model that is open from an inter-operability viewpoint but also with respect to the community of its contributors, reviewers and users. To formalize its openness, the workshop participants agreed on a lightweight, structured process, inspired by the open source community, to manage changes to the OPM specification. The process, aimed at making decisions by consensus, consists of a public call for change proposals, public review and public voting [16].

Using this open process, the workshop was followed by a revision of the specification, in light of the experience gained from the third provenance challenge. Proposed changes, discussions and vote results can be seen at the OPM wiki [12]. We summarize the key changes that are reflected in the version of the specification available in this special issue [11].

A number of structural changes were introduced to the document. While a formal specification is useful to make details precise and explicit, it was decided that it should be moved out of the core OPM specification, since it is difficult to keep it synchronized with the specification. It was agreed that extensibility is paramount in OPM, and that it must captured by a notion of profile. Collections were deemed to be an optional feature, and it was decided that they should be expressed as an optional profile, outside the core specification. On the other hand, time is regarded as an essential feature of OPM, intrinsic to its semantics, and this was kept in the core specification.

Several other issues were brought into the specification: it was made explicit that some edges must be asserted and cannot be inferred (*Used*, *WasGeneratedBy* and *WasDerivedFrom*). A number of common annotations were introduced to capture labels for nodes, sub-types for nodes and edges, and serial representation for artifacts.

Active community participation is helping build a wide user base around the open provenance model specification and plans are already afoot to define the fourth provenance challenge in 2010 to validate all features of OPM by the broader community.

5. Articles in this special issue

The articles in this special issue discuss the role of OPM in fostering interoperability.The diverse selection of articles present experiences using OPM–both as a part of PC3 and for other scenarios, technical and performance evaluations, interoperability challenges, and mapping across and extending provenance models. The Open Provenance Model Core Specification (L. Moreau et al.) is itself included in this issue, defining entities, dependencies, and inference rules with examples. This serves as the definitive version of the specification for both modelers and practitioners of provenance.

A novel application of OPM to represent message communication between distributed components is provided by P. Groth. A distributed profile for OPM is proposed and used to compactly represent the provenance for the load workflow used in PC3. A. Freitas considers another distributed system, the world wide web, and enumerates provenance requirements and use cases for online content. They present mappings between their W3P model based on semantic/web standards and OPM, and identify extensions that meet the identified requirements.

The Swift team used the Swift workflow scripting language to capture provenance for the PC3 workflow and attempt the challenge queries. Their experience, along with a comparison of their internal provenance data model to OPM is presented by L. Gadelha et al. A similar entity–relationship model is used by C. Lim et al. in their *OPMProv* database store for OPM graphs, and its performance evaluation presented for importing and querying OPM graphs from seven other PC3 teams.

As OPM gains traction, existing provenance systems will start incorporating support for OPM. Y. Simmhan et al. explore technical approaches considered when migrating the native provenance model in the Trident workflow workbench to OPM for the challenge and provides insights on programming overhead, data consistency and interoperability issues for each approach considered.

Interoperability challenges exist when collecting provenance from heterogeneous application platforms. L. Ding et al. take a critical look at further interoperability issues that they identify during the course of PC3, and recommend their Linked Provenance Data ontology to encapsulate and extend OPM to address them.

OPM specifies the causal provenance chain, and it is meant to coexist with current and future metadata specifications. S. Miles explores mapping the Dublin Core vocabulary popular in the digital library and archival community to OPM graphs, to enrich the context and attributes available to users while also automating metadata interchange between software components. An interesting comparison between OPM used to represent provenance of workflow execution (among others) and the model used to represent the workflow itself is provided by P. Missier et al. They examine information loss when ''reverse engineering'' a plausible workflow from a given OPM graph, using the Taverna dataflow model as an example, and suggest annotations to OPM to achieve losslessness.

References

[1] Yogesh Simmhan, Beth Plale, Dennis Gannon: a survey of data provenance in e-science, SIGMOD Rec. 34 (3) (2005) 31–36.

[2] L. Moreau, The foundations for provenance on the web, Found. Trends Web Sci. 2 (2–3) (2010) 99–241.

[3] J. Cheney, S. Chong, N. Foster, M. Seltzer, S. Vansummeren, (2009). Provenance: a future history. OOPSLA 2009.

[4] Juliana Freire, David Koop, Emanuele Santos, Claudio T. Silva, Provenance for computational tasks: a survey, Comput. Sci. Eng. 10 (3) (2008) 11–21.

[5] Peter Buneman, Sanjeev Khanna, Wang Chiew Tan, Why and where: a characterization of data provenance, in: ICDT, 2001, pp. 316–330.

[6] Workshop on Data Derivation and Provenance, Chicago, 2002.

[7] International Provenance and Annotation Workshop, Chicago, 2006.

[8] L. Moreau, et al., The first provenance challenge, Concurr. Comput.: Pract. Exp. 20 (5) (2008) 409–418. (special issue).

[9] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, P. Paulson, The Open Provenance Model (v1.0). Technical Report, ECS, University of Southampton. December 18, 2007.

[10] L. Moreau (Ed.), B. Plale, S. Miles, C. Goble, P. Missier, R. Barga, Y. Simmhan, J. Futrelle, R. McGrath, J. Myers, P. Paulson, S. Bowers, B. Ludaescher, N. Kwasnikowska, J. Van den Bussche, T. Ellkvist, J. Freire, P. Groth. (2008) The Open Provenance Model (v1.01). Technical Report, Electronics and Computer Science, University of Southampton. July 17, 2008.

[11] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, Jan Van den Bussche, The open provenance model—core specification (v1.1), Future Generation Computer Systems 27 (6) (2011) 743–756.

[12] The Third Provenance Challenge Wiki website. http://twiki.ipaw.info/bin/view/Challenge/ThirdProvenanceChallenge.

[13] The Pan-STARRS Data Processing and Science Analysis Software Systems, Heasley, J. N. Classification and Discovery in Large Astronomical Surveys. AIP Conference Proceedings, vol. 1082, pp. 352–358 2008.

[14] Yogesh Simmhan, Roger Barga, Catharine van Ingen, Maria Nieto-Santisteban, Lazslo Dobos, Nolan Li, Michael Shipway, S. Szalay Alexander, Sue Werner, Jim Heasley, GrayWulf: scalable software architecture for data intensive computing, hicss, in: 42nd Hawaii International Conference on System Sciences, 2009, pp. 1–10.

[15] Third Provenance Challenge participating teams and members: NCSA (Joe Futrelle, Jim Myers, Robert Clark), Swift (Ben Clifford, Luiz M.R. Gadelha Jr.), Trident (Yogesh Simmhan, Roger Barga, and Dean Guo), UCDGC (Manish Kumar Anand, Shawn Bowers, Bertram Ludaescher, Sven Kholer, Timothy McPhillips, Sean Riddle), SOTON-ISI (Luc Moreau, Paul Groth), UManchester (Paolo Missier), RPI/TWC (James Michaelis, Li Ding, Zhenning Shangguan, Rui Huang), UvA/VL-e (Victor Guevara, Ammar Benabdelkader, Adam Belloum), SDSC (Ilkay Altintas, Daniel Crawl), VisTrails (Juliana Freire, David Koop, Tommy Ellkvist), KCL (Simon Miles), PASS (Uri Braun, David Holland, Peter Macko, Diana MacLean, Daniel Margo, Kiran-Kumar Muniswamy-Reddy, Margo Seltzer, Robin Smogor), Karma (Girish Subramanian, Bin Cao, Beth Plale), UTEP (Paulo Pinheiro da Silva, Nicholas Del Rio, Leonardo Salayandia).

[16] Governance of the Open Provenance Model, L. Moreau, J. Freire, J. Futrelle, J. Myers and P. Paulson, (2009).

Yogesh Simmhan is a Postdoctoral Research Associate in the Computer Engineering Division and Center for Energy Informatics at the University of Southern California. His research focuses on workflows, provenance, data management and information integration on distributed resources like Clouds and clusters, with the goal of building scalable systems for data-driven applications. Dr. Simmhan has a Ph.D. in Computer Science from Indiana University and was previously with Microsoft Research. He is an IEEE and an ACM member.

Paul Groth is Postdoctoral Researcher in the Knowledge Representation and Reasoning Group at the VU University of Amsterdam. His research covers data provenance, scientific workflow and knowledge sharing. He holds a Ph.D. in Computer Science from the University of Southampton (2007) and has done research at the University of Southern California and the Florida Institute for Human and Machine Cognition. He is a member of the ACM and IEEE.

Luc Moreau is a Professor of Computer Science in the School of Electronics and Computer Science at the University of Southampton. His research is concerned with large-scale open distributed systems not subject to centralised control. He has a Ph.D. in Computer Science from the University of Liege, Belgium. He is a member of the ACM and is a BCS fellow.