Chapter 3
Unmanaged Workflows: Their Provenance and Use

Mehmet S. Aktas, Beth Plale, David Leake, and Nirmal K. Mukhi

Abstract. Provenance of scientific data will play an increasingly critical role as scientists are encouraged by funding agencies and grand challenge problems to share and preserve scientific data. But it is foolhardy to believe that all human processes, particularly as varied as the scientific discovery process, will be fully automated by a workflow system. Consequently, provenance capture has to be thought of as a problem applied to both human and automated processes. The *unmanaged workflow* is the full human-driven activity, encompassing tasks whose execution is automated by an orchestration tool, and tasks that are done outside an orchestration tool. In this chapter we discuss the implications of the unmanaged workflow as it affects provenance capture, representation, and use. Illustrations of capture include multiple experiences with unmanaged capture using the Karma tool. Illustrations of use include defining workflows by suggesting additions to workflow designs under construction, reconstructing process traces, and using analysis tools to assess provenance quality.

Keywords: Data provenance, e-Science workflows, provenance capture, data mining, case-based reasoning, intelligent user interfaces

Mehmet S. Aktas
Data to Insight Center, Indiana University, USA
e-mail: maktas@cs.indiana.edu

Beth Plale
School of Informatics and Computing, Indiana University, USA
Data to Insight Center, Indiana University, USA
e-mail: plale@cs.indiana.edu

David Leake
School of Informatics and Computing, Indiana University, USA
e-mail: leake@cs.indiana.edu

Nirmal K. Mukhi
IBM T. J. Watson Research Center, USA
e-mail: nmukhi@us.ibm.com

Q. Liu et al. (Eds.): Data Provenance and Data Management in eScience, SCI 426, pp. 59–81.
springerlink.com

3.1 Introduction

The data products produced during the course of workflow-driven scientific discovery have the potential to advance scholarly research and address pressing societal problems now and in the future. Nevertheless, however effective workflow systems have shown themselves to be at solving problems, there remain scientific discovery processes not amenable to representation within, and execution by, a workflow system. Workflows are inherently human processes, and it would be foolhardy to believe that human processes, particularly as varied as the scientific discovery process, can be fully automated. Computer scientists cannot hope to engineer the human out of the loop, nor can a workflow system promise to support within a single environment every tool scientists will ever use through the course of their research. Acknowledging this fact, we make the distinction between workflows that are executed end-to-end and fully under the control of a workflow orchestration system and those that are not, the latter we call the unmanaged workflow. The *unmanaged workflow* is the full human activity, encompassing tasks whose execution is automated by an orchestration tool, and tasks that are done outside an orchestration tool.

The issue of relevance to us with unmanaged workflows is provenance capture. An unmanaged workflow has a simple interpretation as two disjoint subworkflows with a gap between. We may know only that subworkflow-1, which began at time t_0 and completed at time t_i, occurred before subworkflow-2 which began at t_j and ended at t_n. Nothing more might be known about the relationship between the two. Given a distributed system with unsynchronized clocks, even this temporal relationship may not be known. The human activity occurring between the first workflow subworkflow-1 and second workflow subworkflow-2 could be the act of analyzing a result using a statistical package, could be the creation of a new layered product in at GIS tool, or could be simply a music break completely unrelated to either subworkflow-1 or subworkflow-2. Figure 3.1 illustrates this case.

For the human-in-the-loop workflow illustrated in Figure 3.1, the provenance of the humans actions may contribute to the provenance record of data product R. How can we know what human activity occurred between two workflow fragments? The two could be completely unrelated, and just mark a music and coffee break between two distinct and unrelated tasks. Or do we even need to know? The provenance of a piece of art has gaps in it; gaps that occur when the owner desired anonymity or when theft occurs. It may be sufficient to merely suggest that the two subgraphs are related and leave it at that. But suppose we can obtain provenance information from the human activity piece, how then can we stitch together the provenance from the human-action piece with the two subworkflows?

Mukhi [6] studies business workflow that cannot be fully automated, and addresses incompleteness through the notion of the unmanaged business process, which is a process that encompasses a large number of human driven workflows, the use of collaborative platforms to accomplish shared tasks, and handling of exceptional situations that arise in the context of automated workflows. BPEL4People [40] is an extension to WS-BPEL that allows people to participate in a business process. BPEL4People, though, requires a plan of activity (or a workflow) be known in

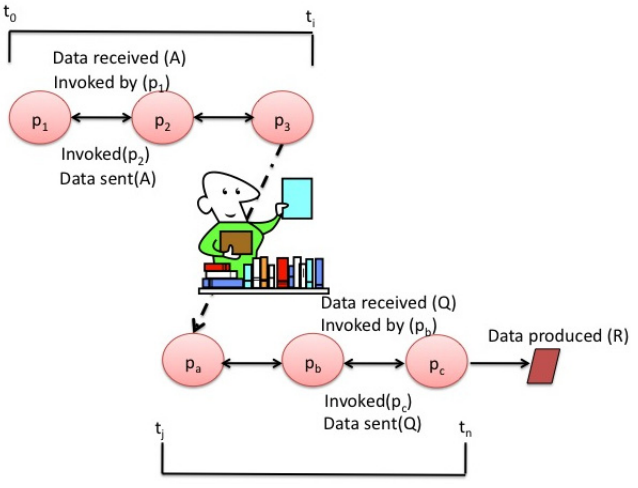

Fig. 3.1 Illustration of an unmanaged workflow

advance that describes the entire business process. BPEL4People, while useful in a limited sense, puts e-Science back at step one in that if scientists do not acknowledge that what they are doing can be described by a workflow, how can one possibly be specified in advance?

The unmanaged workflow defines a problem space of provenance capture wherein two things occur, first, there are non-automated steps in the activity, and second, the full activity cannot be specified in advance. This kind of workflow which is prevalent in e-Science though not in business where workflows are better understood, is a grand challenge for provenance capture.

With no single workflow specification to guide provenance capture, the effects are cascading. Provenance capture becomes more difficult because there is no guidebook of what is supposed to happen, nor is there a single workflow orchestrator that controls execution and determines failure and execution models of the workflow. Representation of the provenance in a provenance store has to deal with fragments of provenance because the captured provenance has a higher likelihood of being ad hoc, noisier, and less complete. Finally, the use of the provenance, though having much in common with use of the provenance of managed workflows, has unique challenges because of the ad hoc nature of the information.

This latter point of the ad hoc nature of provenance has significant implications for trust. A key benefit of having the provenance record of a scientific data object or set of objects is that someone with whom the data is shared can use the provenance to determine their level of trust in the data. If the provenance itself is of questionable quality, it undermines one of the key benefits of provenance of scientific data in the first place. On the other hand, thinking about the provenance record in terms of fragments of workflows that are either part of the lineage trace of a scientific data object or not related, models reality more closely. This is mainly because data are rarely

created from scratch and derived in the same workflow. Finally, the provenance of a scientific data object is a living record, just as the provenance of a piece of art is a living record. Flexibility in dealing with provenance over time will provide the greatest benefit for scientific data provenance.

In this chapter we focus on the provenance of digital scientific data that is generated automatically from unmanaged workflows, and discuss three areas, capture, representation, and use. Dealing with unmanaged workflow has implications at every step of provenance management. In the provenance capture phase, attention must be focused on the instrumentation used to capture provenance and the communication protocols by which provenance information is ferried outside the application. The provenance representation phase is marked by the noted absence of a overall plan (the workflow specification), creating uncertainty as execution-level events arrive. Finally, the use phase must make the provenance valuable for use. This is complicated in the unmanaged workflow by information that is known to be incomplete and ad hoc. Good tool support for automatic provenance capture, representation, and use in the unmanaged workflow setting is critical for realizing the vision of broad scientific data sharing today and in decades to come.

3.2 Provenance Creation

3.2.1 Overview

Provenance creation for unmanaged workflow is the activity of identifying the important provenance activities, defining a data model by which to represent the provenance, mapping the activities to the model, identifying the right communication protocols and instrumentation techniques to employ, then finally, putting capture in place for running applications. A good graph-based model for provenance is the Open Provenance Model (OPM) [7]. OPM represents entities, artifacts, actors, and relationships in the form of a directed graph. The provenance of a data object, D_i, for instance, can be defined by the processes or transformations that were applied to create the data object. The processes can in turn be further described by their inputs and outputs. A relationship between process P_1 and process P_2 exists if process P_2 consumes a data product generated by process P_1. OPM defines the minimal provenance but supports name-value pair annotations that can be used to enhance the information known about the entities and relationships. Provenance can be further enhanced by extending the existing set of relationships and objects, such as was done by Missier et al. [22].

In the unmanaged workflow setting, extracting provenance from an executing application has similarities to real time performance monitoring of a complex, distributed and parallel application. The terminology used in performance monitoring literature when referring to the mechanism for extracting information from an executing application [25] is “instrumentation”, ”instrumentation points”, and ”sensors”, so we adopt the same terminology here. Provenance capture focuses on the

sensors that collect information; as with performance monitoring these sensors must be lightweight, and minimally perturb or pollute the application.

One of the first design decisions is the types of instrumentation that can be supported. Mechanisms for collecting provenance have tradeoffs that must be made between burdening the user, the developer, or application performance; and in the ultimate quality of the provenance information as well. Collection mechanisms fall into one of three categories: user annotation, scavenging, or full provenance instrumentation. Provenance capture through *User annotation* is a human data entry activity where users enter textual annotations, capturing for instance all the data sets used during an analysis of a particular forest use in the Amazon forest over a multiyear period, including video interviews of nearby residents, satellite imagery, and survey data. To make the entry more uniform, the scientist might be prompted to enter specific information. It is however widely understood that user-entered metadata is often incomplete and inconsistent [26]. The annotation approach imposes a low burden on the application, but a high burden on the humans responsible for annotation. The implication is that error rates of the provenance are high.

Full provenance instrumentation refers to instrumentation that is added directly to an application, such as when a programmer must insert calls into their code to call out to a provenance library. Full provenance instrumentation allows for good provenance completeness and consistency, but imposes a substantial burden on the programmer who must modify the application directly. A middle approach is something we refer to as *scavenging*. Here collection is done by means of piggybacking onto existing collection mechanisms, such as a logging tool or an auditing tool, or is carried out in the middleware layer so as to not burden the application programmer. VisTrails [27] implements a form of scavenging when it captures the "do" and "undo" actions of graphical modeling tools as a way to pick up provenance for free. Scavenging has a disadvantage of resulting in incomplete information. Incomplete provenance information can be an acceptable tradeoff for high levels of collection interoperability as long as we can provide a sense of the level of completeness and provide an estimate of the accuracy of the resulting provenance.

The capture layer ingests provenance events, alternately called "notifications", as they are generated at runtime, and queues them for storage to a provenance capture system. The layer is implemented as a protocol and framework to carry provenance events from application components to the database. In this layer, there may be different protocols ranging from a Web Service based system to a publish-subscribe system. This is illustrated on the left of Figure 3.2.

3.2.2 Application in the Karma Tool

The application of provenance capture in unmanaged workflows is best illustrated through example. Our team has had experience applying provenance capture in multiple and varied settings, in which we had to think through the data model, instrumentation mechanisms, protocols. We summarize this experience in Table 3.1. PC3 is Provenance Challenge 3 [37], a friendly competition undertaken June 2009 in the

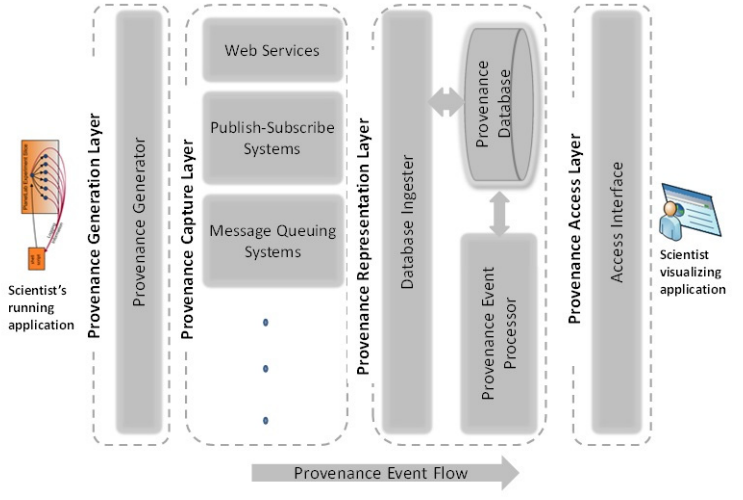

Fig. 3.2 Logical architecture of a provenance system

provenance community. Teams implemented an astronomy workflow that used their provenance system to answer a set of queries. Ratings depended on the number of queries the systems could answer and the ability to capture the most information in the query. AMSR-E is shorthand for a satellite instrument processing pipeline application. The processing pipeline is for the Advanced Microwave Scanning Radiometer - Earth Observing System sensor located on the NASA Aqua satellite. The pipeline is a script-driven application that continuously ingests images from the polar orbiting satellite, processes the images to identify sea ice over the poles, etc. The pipeline is made up of legacy processing scripts and scientific algorithms, the latter of which are of deep importance to the provenance record. The LEAD application is a workflow system for executing weather related analysis and modeling activities. The GENI application is a computer networking application. It applies provenance capture to the PlanetLab distributed network. Table 3.1 identifies three different kinds of instrumentation and two protocols that are used in the four applications, in some applications two instrumentation techniques are used. The implications of the capture mechanism, using the vernacular identified in Table 3.1, are given in the column titled "Provenance capture burden". As it can be seen from Table 3.1, there are various instrumentation mechanisms, one of which is a full provenance instrumentation approach, with the attendant high burden on the application programmer to correctly place the instrument points in the code, and to write provenance events to a format that Karma requires. We see this as our least viable solution because of this dual-headed burden. The first scavenging approach taps into the messages flowing between applications and transparently routes a copy of the event to Karma. It treats the application as a black box, so provenance is limited to what can be captured through message traffic. Within that approach, there are two schools of thought as

to whether or not one should peek into the message contents to extract further provenance. The second scavenging approach captures provenance information from log files.

The Karma system currently supports two forms of communication in which the provenance event come to the system, a Web service based model and a publish-subscribe messaging system [38, 39]. In addition to synchronous submission of notifications, support for asynchronous publishing of provenance is supported through a publish subscribe system. Sometimes called an Enterprise Service Bus, a publish subscribe system decouples publishers of events and consumers of events, allowing new publishers and subscribers to join simply by having the topic name and location of the broker that brokers subscriptions.

3.3 Provenance Representation

A provenance system can be viewed as a repository that (1) actively collects and ingests events in real time, (2) stores the events in a data model that supports time-series data storage, aggregation and synthesis of the events to form new knowledge, and (3) provides an access layer that supports access to the data. Provenance systems are often designed to serve a single use, such as provenance capture for a single workflow system. As attention is increasingly being paid to the long term sharing and preservation of digital scientific data, provenance systems can be valuable repositories of information about the circumstances under which data objects were created, information that is essential to reuse of the data object in a new setting.

The provenance representation layer, illustrated in the middle of Figure 3.2, stores provenance data using a data model that represents the execution instance notifications, and higher layers that abstract from execution instances. The representation layer is where post-processing is carried out such as to organize the events and derive higher levels of behavior or knowledge from the events. The representation layer is where the impact of unmanaged workflows is most strongly felt because for unmanaged workflows there is no obvious reference point to which arriving execution events can be tied such as would be provided by a workflow known in advance.

Provenance systems use different data models, Karma uses a two level model; Trident and VisTrails use a three level data model. Karma includes both execution details for utilizing the data and high level information for long term preservation [41]. This layer should contain information about services and data products at a sufficient level of detail to support discovery and automated decisions about whether to bind a particular data product or service. This layer should contain information for locating and retrieving data artifacts for use in a workflow execution and capture instance invocation and execution details of a particular run. The representation layer should store common information consistently and without redundancy. A provenance capture system captures provenance by accumulating discrete run time activities during the lifecycle of unmanaged workflows, that is, workflows whose structure is not known to the system in advance of execution.

Table 3.1 Instrumentation techniques supported by Karma and the two communication protocols have different tradeoffs [44].

Instrument mechanism	Communication protocol	Provenance collection burden	Application
Application responsible for invoking library that constructs provenance notifies	and invokes Axis2 send call. Specialized Axis2 handler routes message to Karma.	Programmer must be provenance savvy. High application burden.	PC3
Application responsible for invoking library that constructs XML provenance notifies	and publishes to messaging system (i.e. RabbitMQ). Karma listens for events.	Application programmer must be provenance savvy. High application burden.	AMSR-E
Application publishes SOAP notifications as part of normal activity	and publishes to Axis2 call. Axis2 handler transparently grabs copy of event and sends to Karma without application being aware.	Karma parses notifications on server side to extract useful provenance information. Assumes basic provenance behavior is present in message. Scavenging approach.	LEAD
Application publishes notifications as part of normal activity	and publishes to RabbitMQ. Karma is sitting on topic/-subject so captures event without application being aware.	Same as above. Scavenging approach.	GENI
Application writes log messages to log file as part of normal activity	and Karma Adaptor parses log file (client side parsing) and generates notifications that are sent via Axis2 or RabbitMQ.	Adaptors need to be written to parse log file; assumes core provenance behavior has been written to log. Scavenging approach.	GENI, AMSR-E

3.3.1 Representation in Karma

Karma stores provenance data using a two-layer information model which includes both execution details for utilizing the data and registry information for long term preservation [41]. The two-layer information model contains a registry level, which contains metadata about the instance, and an execution level. The registry level has similarities to registries used in web service architectures in that it contains information about services and data products at a sufficient level of detail to support discovery and automated decisions about whether or not to bind a particular data product or service. The registry level is not used for locating and retrieving data artifacts for use in a workflow execution, but nevertheless contains sufficient information for building a data object that can be preserved indefinitely. The execution level captures instance invocation and execution details of a particular run. The two-layer model recognizes commonalities in workflows and stores that common information consistently and without redundancy. Some of the concepts of this two-layer information model map directly to OPM. For instance, data products such as data granule and data collection can be considered artifacts; entities (services including composite service and opaque service, and methods) can be considered processes; and clients (a kind of entity, which may be a user or a workflow engine that initiates the workflow) can be considered agents.

A significant implication of not knowing the structure of workflows in advance is that in addition to not having a picture of execution before it occurs, Karma can make no assumptions as to the existence of global state in the application either. A provenance notification message will be issued by a task, and the information contained in that notification will be based on what can be gathered from local state only. For instance, tasks within a workflow may not know the session or workflow to which they "belong" so it can be difficult for the Karma service to tie a service invocation back to the workflow that invoked it, particularly for recursive or chained services. For unmanaged workflows, OPM is inadequate to the task of defining the formats of provenance events if for no other reason than provenance capture is messier and more incomplete than the graph-based OPM can handle.

3.4 Provenance Use

Captured provenance provides a rich source of information about workflow execution. Automatic provenance collection over time generates a substantial body of knowledge which may be used in many ways. Referring back to the logical architecture shown in Figure 3.2, the access layer supports the Query API, which is used to pose queries to the provenance capture system to retrieve provenance. This layer that allows users to explore and examine large quantities of data requires browse capabilities. The browse pattern characteristically involves starting with some broad information, performing a search, finding general result sets and then selecting more

specific information for drill-down. The access layer includes macro-level queries, tracing provenance relationships back through time in order to construct a graph, as well as object-level queries that locate information about specific entities matching the conditions specified in passed arguments.

Our research investigates three novel additional uses for provenance, described in the following sections: Using captured provenance to aid workflow construction, to repair problems in provenance capture, and to analyze workflow trace quality.

3.4.1 Using Provenance to Aid Workflow Construction

Provenance acquired by provenance capture systems is rich source of information about the workflows which gave rise to the observed processes. We are investigating how such information can be used to aid workflow construction. When a subpart of a partially constructed workflow involves a sequence of services observed in a past provenance instance, that remainder of the stored provenance can suggest extensions of the partial workflow, to present to the workflow author. We are exploring both the mining of stored provenance for statistical correlations on which to base predictions, and the use of case-based reasoning (CBR [48, 49]) to predicting solutions to new problems based on relevant instances of similar prior problems. CBR is a "lazy learning" method in that cases are stored with minimal pre-processing, simplifying knowledge acquisition. Because CBR reasons from relevant prior episodes—cases—rather than rules, it is a natural approach for reasoning from libraries of examples such as provenance databases. The performance of CBR systems depends on how well their stored cases cover the space of problems to solve. Large-scale provenance databases provide an extensive starting point, and each new workflow execution provides a new case to extend coverage. In addition, workflow problem types tend to recur—for example, scientists in a particular domain will tend to generate certain types of workflows [50], increasing the chance that stored traces will be relevant to new situations.

The Phala project[1] [50, 51] develops and tests a case-based approach to aiding workflow construction. Phala is a plug-in to the XBaya graphical workflow composer [45]. Phala's processing cycle is illustrated in Figure 3.3.

As a user develops a workflow, Phala monitors the partially constructed workflow and generates background retrieval queries to a provenance database. The provenance database need not have been generated for a single particular task; similarity assessment process selects those cases which are relevant. When cases are not available, the system can provide recommendations based on statistical methods, which use statistics mined from the provenance database. Because there is no guarantee that the suggestions generated by statistical methods and cases will agree (or even that all suggestions from relevant cases will agree), we are developing approaches

[1] The name Phala was inspired by the naming of the Karma provenance capture system. In Sanskrit, Karma means causality and reflects captured provenance, Karma means the ripened fruit, so KarmaPhala means the fruit of provenance capture, reflecting the cases generated by Phala.

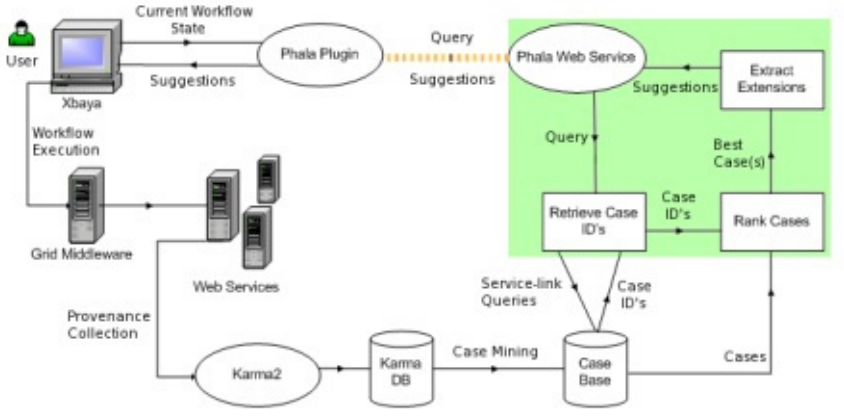

Fig. 3.3 Phala's processing cycle

for combining multiple (and possibly conflicting) recommendations, in order to extend the range of situations for which Phala can make recommendations and increase recommendation accuracy. Initial tests of these methods are promising [51]. In addition, Phala allows users to control the level of confidence required for the system to propose suggestions.

Large provenance case bases provide both benefits and challenges for CBR. Given a large and diverse set of cases, a CBR system can solve a wide range of problems. However, retrieval and similarity assessment for large case bases can be computationally expensive, especially when the cases which need to be compared involve structured information. For graph-structured case information such as workflow traces—for which matching could be seen as an instance of the subgraph isomorphism problem—matching cost is a particularly acute issue. In addition, the anticipated size of provenance case bases far exceeds that of case bases previously studied by the case-based reasoning community (for example, as described in Section 3.4.3, a 10 GB database was recently developed as a provenance testbed). Consequently, a central goal of the Phala project has been to develop procedures enabling efficient retrieval of structured cases.

For generality, Phala's retrieval methods are primarily domain-independent. Phala's retrieval is performed by the Structure Access Interface (SAI), a toolkit for structure-based retrieval. To increase retrieval efficiency, SAI implements a two-phase retrieval approach in which the initial phase can be seen as coarse-grained filtering, to retrieve a small set of potentially relevant cases for more expensive structural matching. More detailed descriptions of the algorithms and evaluations are omitted here for reasons of space, but are available elsewhere [50, 51, 53].

3.4.2 Using Data Provenance Traces to Reconstruct Process Traces

Data provenance traces from multiple systems need to be connected into a coherent graph that represents the relationships between various data-related events. For example, one system might generate an event corresponding to the sending of a message, while a second system might generate an event corresponding to the message being received: if these are in fact the same message, these events need to be connected to recover the end-to-end trace of what actually occurred. The process of doing so is termed trace reconstruction; we first discuss how this is done in general, and then focus more closely on unmanaged processes. The process of reconstructing process traces from provenance data involves three distinct phases: Collection, Correlation, and Enrichment.

Collection: This phase involves gathering provenance data from various source systems. Adapters are built to extract events or log information from the source system, perform appropriate transformations to produce provenance items and then record these provenance items into a centralized provenance store. For reconstructing process traces, a provenance solution would involve deploying adapters to all systems where any process activity occurs, such as document repositories, web servers, email servers and so on.

Correlation: This phase involves correlation of provenance items within the provenance system. Correlation for the purpose of reconstructing a process trace will involve using an opaque process identifier if available, or a set of application data that collectively serves as the identifier for a process, and then using the identifier to stitch together the tasks, data and actors involved in the correct temporal sequence. The correlation will also locate identifiers that help to bridge systems (such as a message identifier that helps us connect a message sent from one system to that received in a different system).

Enrichment: When reconstructing process traces, provenance items recorded as multiple tasks by adapters may together correspond to a single process activity from the user's perspective. Creation of such a higher level abstraction would be done at this time. For example the entire sub-graph showing the sending of a message from one system to another could be abstracted into a single node labeled 'message exchange', with the underlying provenance of this activity preserved elsewhere in the graph. This simplification serves two purposes: it allows for easier visualization and also reduces the complexity of consuming new information. For example, further correlation or enrichment of the graph could be triggered on new 'message exchange' node, rather than on the more complex pattern it corresponds to.

It is important to note that these phases need to operate concurrently; data that is being recorded has to be correlated and enriched at the same time other data is arriving; i.e. development of the provenance information and its use to reconstruct the process trace is a continuous process. The outcome of this continuous process is a process trace represented as a provenance graph.

A result of the observed process being unmanaged is that sometimes correlation and enrichment are non-trivial. Consider the process of running a long-running scientific experiment across a computing infrastructure that requires data products from one system to be manually copied to a different system. In many real-world instances, the method for achieving this is for scientists running the experiment to request system administrator to perform the data copy. The request forms an important part of the overall process and allows the subsequent data set used to be verified as being the correct one. However, correlating an unstructured request such as an email with other events such as a file copy is non-trivial. In most cases, the request would specify the source file and target location, but this may not always be the case. In such instances, the time the request was made could be used for correlation. In general, it is certainly possible to miss the request that was used when creating the provenance graph, or to correlate an incorrect request. We classify uncertainties in provenance graphs under three categories: Node versus Edge uncertainty, Simple versus Complex Uncertainty, and Static versus Dynamic uncertainty.

Node versus Edge uncertainty: All the nodes in the provenance graph can be grouped into two categories, nodes recorded by adapters (Type A) and nodes created through derivation from those (Type B nodes). Nodes recorded by adapters are accurate since they exactly represent something that occurred in a source system. Derived nodes, added through feature extraction or enrichment may however be inaccurate, and may therefore have uncertainty associated with them. Edges are recorded based on correlations or shared features between nodes. When edges are recorded between Type A nodes that were recorded from the same source system, they are guaranteed to be accurate (since they are based on correlations of consistent and accurate data). All other edges may be imprecise. For example, an edge showing time ordering between Type A nodes from different source systems may be inaccurate if the clocks are not synchronized. Additionally, edges between Type B nodes may be imprecise since the node uncertainty is propagated to the edge, i.e. the edge may be created based on imprecise data. Going back to our earlier example of correlating a request to copy data with the actual system activity corresponding to the data movement, the resulting provenance graph would have accurate information on each of these basic events, the uncertainty would arise when trying to determine which of the available email requests correspond to a particular data movement activity.

Simple versus Complex Uncertainty: Uncertainty associated with a provenance item may be entirely a function of features of that provenance item itself. We call such uncertainty simple. Sometimes the uncertainty is a function of a set of provenance data; in such cases it is said to be complex.

Static versus Dynamic uncertainty: When the uncertainty associated with a given provenance item is fixed, it is said to be static uncertainty. Sometimes the uncertainty associated with a piece of information varies over time. This is dynamic uncertainty.

Doganata et al. [43] have explored an approach in which information is correlated only if it appears to meet a certain confidence threshold. We [6] have found that it is better to represent uncertainty in a first class manner within the provenance

graph itself, and allow for provenance applications or further enrichment to decide what information is accurate, given a more global view of the end-to-end process.

3.4.3 Using Provenance for Analysis of Workflow Traces

Real world provenance data traces are often noisy, resulting in disjoint or incomplete provenance traces. Provenance messages may be dropped, messages can be incomplete (which could occur when the application scope at a point of notification generation is more restricted than anticipated), or execution of the application (or workflow) can simply fail. To properly asses captured workflow traces, it is crucial to establish algorithms to analyze those traces, identifying failures and assessing the quality of data provenance traces.

Identifying failure in provenance traces: We have developed a model for analyzing provenance traces and identifying failures [52]. The model identifies two types of failures: a) task failures where a node in a workflow does not complete successfully, b) communication failures in which a task completes but the notification is not successfully transmitted.

We have performed experiments studying four failure modes as follows: a) No failures and dropped notifications (success case), b) 1% failure rate, c) 1% dropped notification rate, d) 1% failure rate and 1% dropped notification rate. These failure rates are modeled using uniform distributions in a workflow emulator, WORKEM [54], to determine if a particular invocation must fail or drop a notification. Using the WORKEM to generate provenance, the following six major workflows were used as the basis for generating a large scale (10 GB) provenance database: LEAD North American Mesoscale (NAM) initialized forecast workflow, SCOOP ADCIRC Workflow, NCFS Workflow, Gene2Life Workflow, Animation Workflow, MotifNetwork Workflow. These workflows are pseudo-realistic, in the sense that they are modeled after real life workflows. The LEAD NAM, SCOOP and NCFS are weather and ocean modeling workflows, Gene2Life and MOTIF are bioinformatics and biomedical workflows, and the Animation workflow carries out computer animation rendering. Some of the workflows are small, having few nodes and edges, while others like Motif have a few hundred nodes and edges.

Figure 3.4 shows the results where the distribution is dissimilar. Even though the generation settings for WORKEM were identical across workflows, WORKEMs failure model does not result in the same uniform distribution across different workflows since the configuration for failure rates is per task in the workflow. For both Animation and Motif workflows, the number of runs that do not have failures or dropped messages is approximately half of what the smaller workflows exhibit, which supports that the larger a workflow, the higher the failure rate and dropped messages rate. The smaller workflows appear to have the same distribution compared to each other.

Quality Assessment of provenance traces: We have developed a methodology for assessment of the provenance goodness [52]. This methodology applies statistical approaches that operate over large volumes of data to zero in on suspicious

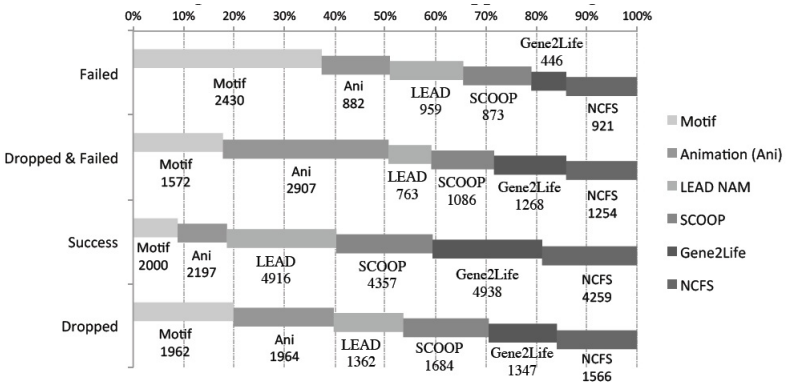

Fig. 3.4 Distribution of workflows by population cases

provenance records. Based on this approach, provenance goodness is determined by constructing the best possible provenance graph for an execution based on the captured provenance record, then assessing the goodness of the resulting graph by looking at the partitions in a provenance graph. In this approach, a provenance graph is be modeled as PG = V, E, where V is a collection of vertices that are linked by one or more directed edges, E. This approach is used to construct a provenance graph from nothing (no guiding workflow template) based only on the captured provenance. It relies on an assumption that all provenance notifications contain the correct ID for the workflow execution instance to which they belong. WORKEM workflow emulator supports this assumption. Even with this simplification, this approach still may yield disconnected components. The query of a graph using a workflow ID searches over the database tables for entities (processes) that have matching IDs. If there are dropped messages, the queried graph may have missing edges or missing vertices. The only guarantee for the retrieved graphs is that the components of the graph are linked through that workflow ID.

Figure 3.5 shows the results of this approach, when the algorithm is applied to the aforementioned large scale 10GB provenance database. Observing the number of edge counts of each workflow instance, we conclude that the results for the LEAD NAM workflow are preliminary. The plot points are classified based on the statuses of each workflow. Based on the results, the workflows with dropped messages cluster towards the upper end of Figure 5. This implies that dropped messages for successful workflows are few. In comparison, workflows that involve failures typically result in more missing notifications, resulting in lesser number of edges in their provenance graphs.

Automatic provenance repair: As described previously, automatic provenance capture is imperfect. Correlation and enrichment methods aid in reconciling and merging information, but in some cases, the messages relating to a process may simply be lost, resulting in gaps in the provenance trace. Consequently, we are studying

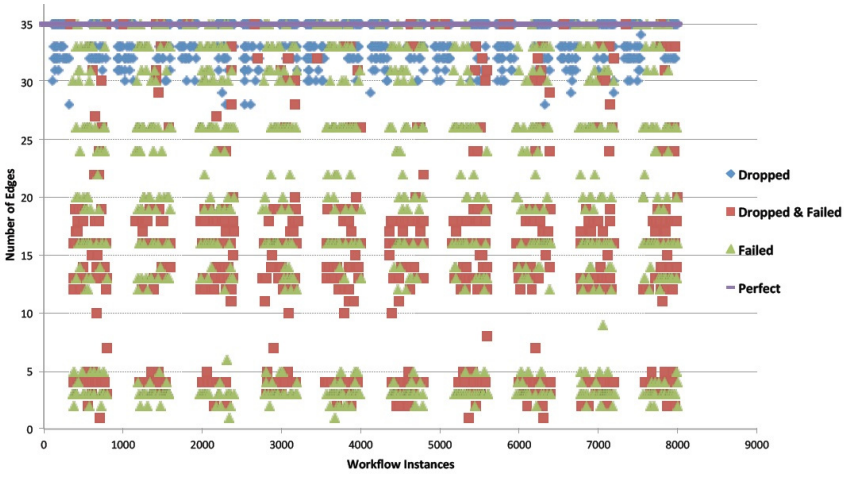

Fig. 3.5 Plot of edge counts for LEAD NAM workflow instances with different statuses

methods for assessing overall provenance goodness. For example, we have begun to explore using simple graph analysis methods to identify connected components: If a provenance graph is disconnected, it reveals gaps in provenance capture. In such instances, methods such as Phala's prediction methods, applied to the disconnected components, may be able to infer missing steps. Confidence assessment methods already developed for Phala's recommendations may be used to determine confidence in proposed repairs. This will enable a flexible repair approach in which repairs are only pursued if their confidence exceeds a use-designated threshold, and will also enable annotating repaired provenance with confidence values for the quality of the provenance.

3.5 Related Work

Provenance has been studied from different perspectives and several surveys have been published [4, 21, 28, 29]. Simmhan et al. [4] introduced a taxonomy of provenance in e-science, specifically for scientific workflow systems, based on why they record provenance, what they describe, how they represent and store provenance, and ways to disseminate it. Moreau [29] did a comprehensive survey analyzing 425 papers in the provenance literature and reviewing its potential benefits in e-science, curated databases, and semantic web. In this section, we give a brief review of provenance capture in major provenance systems particularly designed for scientific workflows. Generally, provenance systems for scientific workflows can be categorized into two classes: those in scientific workflow systems and standalone provenance systems.

Provenance Use: Provenance systems track artifacts from various systems and correlate these to create a provenance graph. Using a provenance graph, it is possible to recover process traces or data lineage, or to maintain a record of user activity for various purposes. It is also possible to build systems that aid workflow construction. Workflows are often generated by scientists who are not experts on scientific computation, who may have difficulty choosing appropriate services. Even for experts, workflow generation may be time-consuming. Consequently, software to facilitate workflow generation is highly desirable and a number of efforts have aimed to assist scientist in workflow generation, using both generative planning and interactive approaches. Systems such as FlowRecommender [11] and Viscomplete [12] mine process traces for the development of automated workflow recommendations. Wen et al. introduced an approach for process mining problem in dealing with invisible tasks, i.e., such tasks that exist in process model but not in its event logs [10], for deploying new business processes as well as auditing, analyzing and improving already interacted ones. Interfaces such as XBaya [45] aid users by abstracting away from the details of workflow languages. Knowledge-rich artificial intelligence methods have been developed to generate workflows automatically [46] and to provide interactive support for carefully codified domains [47]. Such approaches can provide excellent performance, but at the cost of expensive knowledge capture, which becomes a major impediment to fielding such systems in new domains. We are investigating data-driven methods to support human workflow generation with minimal knowledge capture.

Provenance tracking has been used to assist in reproducibility of scientific experiments [16], monitoring complex processes that span multiple systems [15] and measure compliance of unmanaged processes [17]. However, none of the existing literature has dealt with the the issue of uncertainty in provenance data. We are investigating the uncertainties involved in creating provenance traces for unmanaged processes and how to represent these using a provenance data model.

Systems such as Karma address provenance capture that is tightly coupled to a workflow system and provenance capture in non-structured e-Science environments. These systems provide a controlled provenance generation environment and do not necessarily contain provenance with failures.

A number of synthetic workflows have been generated and used in distributed systems [18, 19, 20] and computing networking [13, 14] research areas for performance evaluations and benchmarking purposes. However, none of these workloads attempt to model failures and have been specifically developed for the purpose of provenance research. As discussed previously, we use a noisy 10 GB provenance database that models failures of provenance notifications to explore methods to provenance repair and provenance quality assessment.

Provenance Systems: Provenance collection is widely supported in scientific workflow management systems because provenance data can be easily captured and recorded during execution of a workflow. Kepler [30] is used by scientists in multiple disciplines for the design and execution of workflows. The provenance component, Provenance Recorder (PR), is optional depending on the user's requirement

to track provenance. To enable provenance capture in a workflow instance, the user drags the provenance recorder from the toolbox, places it on the workspace, and fills in the configuration menu. The provenance associated with the workflow definition is automatically generated by the existing MOML (Kepler's internal XML workflow representation) generation capabilities during a workflow run. To receive the provenance data, the PR implements several event listener interfaces. When the workflow is loaded, the PR will register with the appropriate concerns in the workflow. When the workflow is executed, PR will process information received as events, and save it in provenance store.

The Taverna workbench [31] is developed for the composition and execution of workflows for the life sciences community. Provenance data is recorded for workflows in the Simplified conceptual workflow language (Scufl) language with four levels [32]: a) process provenance, b) data provenance, c) organization provenance, and d) knowledge provenance. The process provenance records the order of service invocations, inputs/outputs to these services, and the time information of service invocations and workflow executions. The data provenance builds a derivation path of data objects in a workflow run. The organization provenance stores the metadata for the experiment such as who, when, and where the information was created and how it evolved during experiments. The first two levels of provenance are automatically logged during workflow execution. The organization and knowledge provenance can be obtained from three different sources: users' annotations of the Scufl workflows through a knowledge template plug-in; service descriptions from the myGrid semantic service discovery component Feta; and provenance published by the third-party data providers [33].

VisTrails [27] is a workflow and provenance management system that provides support for scientific data exploration and visualization. It is designed to handle rapidly-evolving workflows by using a change-based provenance model. The VisTrails provenance information is organized into three layers: workflow evolution, which captures the relationships among the series of workflows created in an exploratory task; workflow, which consists of individual workflows; and execution, which stores run-time information about the execution of workflow modules. The information for the first two layers is naturally captured by the change-based provenance mechanism. When a user modifies a workflow, his/her actions are captured by the History manager and saved in the VisTrails Repository. Run-time information is captured by the Workflow Execution Engine and stored in the VisTrails Log. Annotations are allowed at all levels of the layered provenance model.

The Trident [34] workbench is a scientific workflow system which is built on top of Windows Workflow Foundation (WF), a workflow enactment engine included in the Windows operating system. It provides an integrated way to collect, store, query, and view provenance for scientific workflows. Provenance information in Trident is a combination of the workflow schema—static, composition information about the workflow—and the provenance schema—dynamic, runtime information about the actual execution of a workflow instance. The Workflow Composer is the primary source of the workflow schema, and the Execution Service and Windows WF engine are the two main sources of the provenance schema. The Execution Service tracks

the submission of each workflow instance, and the Windows WF engine natively generates tracking events of the workflow execution. Trident uses the BlackBoard [35] publish-subscribe, asynchronous messaging framework, to distribute the events from the source to the provenance storage. The Provenance Service listens to the events and records them in the provenance store. Trident event handlers listen for the built-in events to trace the workflow's control flow. The data flow knowledge obtained from the input and output parameter values passed to/from the activities are captured by the instrumentation in the Trident base activity class to generate customized user events.

The PASAO project provides an interoperable way to collect provenance in a grid environment using an open provenance protocol. Miles et al. [36] analyzed 23 use cases in biology, chemistry, physics, and computer science and determined 14 technical requirements for a generic, application-independent provenance architecture. PASOA is designed in three layers: fundamentals of recording and access, querying, and processing. PASAO supports the recording and use of three types of provenance: interaction provenance, which records interactions between components and data passed between them; actor provenance, which records processes information and the time of the execution; and input provenance, which records the set of input data to infer a data product. Groth and Moreau described the recording protocols in [9] in detail. Additionally, Frew et al. [23] captures application calls to the operating system (i.e., kernel calls) and Holland et al. [24] captures file system access.

3.6 Current and Future Challenges

The notion of unmanaged workflows reflects acceptance that human processes cannot be fully automated. For whatever reason, there are pieces of the workflow that remain with the user or are executed outside of and away from the "eyes" of the workflow system. If the task of provenance capture is to record a complete provenance or lineage record then the task is doomed to failure. This gives rise to the question: What can be done? The assumption in the phrase "complete provenance" must be revisited. Just as the provenance of a work of art may have gaps, so incompleteness in provenance of scientific data may be more common than we may think. Methods such as trace reconstruction may help to fill in the trace. However, is complete provenance necessary, or even desirable? Provenance capture can result in large volumes of very low level information. Provenance capture in the AMSR-E satellite imagery processing stream reveals significant amounts of "housekeeping" information, such as that the processing script ran on a certain day. Scientists who question a resulting image are interested in the version of the science algorithm that was applied, but not in the specific day. How can we separate the wheat from the chaff to identify the provenance that contributes meaningfully to the final outcome? Finally, too few examples of compelling uses of provenance captured from real applications exist to convince communities of users that provenance systems are worth the investment of time. These questions and challenges make provenance and provenance use a rich research area for the future.

Acknowledgements. We thank Bin Cao of Teradata Corp. and Yiming Sun, You-Wei Cheah, Yuan Luo and Peng Cheng of Indiana University for their contributions and discussions. This research funded in part by the National Science Foundation under grant OCI-6721674, by BBN Technologies, and by NASA under NNX10AM03G.

References

1. Droegemeier, K., Gannon, D., Reed, D., Plale, B., et al.: Service-oriented environments for dynamically interacting with mesoscale weather. Computing in Science and Engineering. IEEE Computer Society Press and American Institute of Physics 7(6), 12–29 (2005)
2. Folino, G., Forestiero, A., Papuzzo, G., Spezzano, G.: A grid portal for solving geoscience problems using distributed knowledge discovery services. Future Generation Computer Systems 26(1), 87–96 (2010)
3. 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(5), 528–540 (2009)
4. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-Science. ACM SIGMOD Record 34(3), 31–36 (2005)
5. Simmhan, Y., Plale, B., Gannon, D.: Towards a Quality Model for Effective Data Selection in Collaboratories. In: IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow 2006), Held in Conjunction with ICDE, Atlanta, GA (2006)
6. Mukhi, N.K.: Monitoring Unmanaged Business Processes. In: 18th Int'l Conf. on Cooperative Information Systems, CoopIS (2010)
7. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., Bussche, J.V.D.: The Open Provenance Model core specification (v1.1). Future Generation Computer Systems 27(6), 743–756 (2010) ISSN 0167-739X, 10.1016/j.future.2010.07.005
8. Groth, P., Miles, S., Moreau, L.: PReServ: Provenance Recording for Services. In: UK e-Science All Hands Meeting 2005, Nottingham, UK (September 2005)
9. Groth, P., Moreau, L.: Recording Process Documentation for Provenance. IEEE Trans. Parallel Distrib. Syst. 20(9), 1246–1259 (2009)
10. Wen, L., Wang, J., van der Aalst, W.M.P., Huang, B., Sun, J.: Mining Process Models with Prime Invisible Tasks. Data and Knowledge Engineering 69(10), 999–1021 (2010)
11. Zhang, J., Liu, Q., Xu, K.: Flow Recommender: a workflow recommendation technique for process provenance. In: Proceedings of the Eighth Australasian Data Mining Conference (AusDM 2009), Brisbane, Australia (December 2009)
12. Koop, D., Scheidegger, C.E., Callahan, S.P., Freire, J., Silva, C.T.: Viscomplete: Automating Suggesstions for Visualization Pipelines. IEEE Transactions on Visualisation and Computer Graphics 14(6), 1691–1698 (2008)
13. Antonatos, S., Anagnostakis, K., Markatos, E.: Generating realistic workloads for network intrusion detection systems. In: ACM Workshop on Software and Performance, Redwood Shores, CA, USA (2004)
14. Noble, B.D., Satyanarayanan, M., Nguyen, G.T., Katz, R.H.: Trace-Based Mobile Network Emulation. In: Proceedings of SIG-COMM 1997, Cannes, France, pp. 51–61 (September 1997)
15. Curbera, F., Doganata, Y.N., Martens, A., Mukhi, N., Slominski, A.: Business provenance - a technology to increase tracibility of end-to-end operations. In: OTM Conferences, vol. (1), pp. 100–119 (2008)

16. Davidson, S.B., Freire, J.: Provenance and scientific workflows: challenges and opportunities. In: Proceedings of ACM SIGMOD, pp. 1345–1350 (2008)
17. Doganata, Y., Curbera, F.: Effect of Using Automated Auditing Tools on Detecting Compliance Failures in Unmanaged Processes. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 310–326. Springer, Heidelberg (2009)
18. Bodnarchuk, R.R., Bunt, R.B.: A synthetic workload model for a distributed systems file server. In: Proceedings of the SIGMETRICS Interational Conference on Measurement and Modeling of Computer Systems, pp. 50–59 (1991)
19. Mehra, P., Wah, B.: Synthetic Workload Generation for Load-balancing Experiments. IEEE Parallel and Distributed Technology 3(3), 4–19 (1995)
20. Sreenivasan, K., Kleinman, A.J.: On the construction of a representative synthetic workload. Communications of the ACM, 127–133 (1974)
21. Freire, J., Koop, D., Santos, E., Silva, C.T.: Provenance for Computational Tasks: A Survey. Computing in Science and Engineering 10(3), 11–21 (2008)
22. Missier, P., Sahoo, S.S., Zhao, J., Goble, C., Sheth, A.: *Janus*: From Workflows to Semantic Provenance and Linked Open Data. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 129–141. Springer, Heidelberg (2010), doi:10.1007/978-3-642-17819-1-16.
23. Frew, J., Janée, G., Slaughter, P.: Automatic Provenance Collection and Publishing in a Science Data Production Environment—Early Results. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 27–33. Springer, Heidelberg (2010)
24. Holland, D., Seltzer, M., Braun, U., Muniswamy-Reddy, K.: PASSing the provenance challenge. Concurrency and Computation: Practice and Experience 20(5), 531–540 (2008)
25. Gu, W., Eisenhauer, G., Schwan, K.: Falcon: On-line Monitoring and Steering of Parallel Programs. Concurrency Practice and Experience 10(9), 699–736 (1998)
26. Newhouse, S., Schopf, J., Richards, A., Atkinson, M.: Study of user priorities for e-infrastructure for e- research (SUPER). In: Proc. of the UK e-Science All Hands Conference (2007)
27. Scheidegger, C., Koop, D., Santos, E., Vo, H., Callahan, S., Freire, J., Silva, C.: Tackling the Provenance Challenge one layer at a time. Concurrency and Computation: Practice and Experience 20(5), 473–483 (2008)
28. Bose, R., Frew, J.: Lineage retrieval for scientific data processing: a survey. ACM Comput. Survey 37(1), 1–28 (2005)
29. Moreau, L.: The foundations for provenance on the web. Foundations and Trends in Web Science 2(2-3), 99–241 (2010)
30. 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)
31. Oinn, T., Greenwood, M., Addis, M., Alpdemir, N., Ferris, J., Glover, K., Goble, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Pocock, M., Senger, M., Stevens, R., Wipat, A., Wroe, C.: Taverna: lessons in creating a workflow environment for the life sciences. Concurrency and Computation: Practice and Experience 18(10), 1067–1100 (2006)
32. Zhao, J., Wroe, C., Goble, C., Stevens, R., Quan, D., Greenwood, M.: Using Semantic Web Technologies for Representing E-science Provenance. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 92–106. Springer, Heidelberg (2004)

33. Zhao, J., Goble, C., Stevens, R., Turi, D.: Mining Taverna's semantic web of provenance. Concurrency and Computation: Practice and Experience 20(5), 463–472 (2008)
34. Barga, R., Simmhan, Y., Withana, E.C., Sahoo, S., Jackson, J.: Provenance for Scientific Workflows Towards Reproducible Research. Bulletin of Technical Committee on Data Engineering, Special Issue on Data Provenance 33(3), 50–58 (2010)
35. Valerio, M., Sahoo, S., Barga, R., Jackson, J.: Capturing Workflow Event Data for Monitoring, Performance Analysis, and Management of Scientific Workflows. In: IEEE Fourth Int'l Conf. on e-Science 2008 (e-Science 2008), pp. 626–633 (2008)
36. Miles, S., Groth, P., Branco, M., Moreau, L.: The requirements of recording and using provenance in e-science experiments. Journal of Grid Computing 5(1), 1–25 (2007)
37. PC3, `http://twiki.ipaw.info/bin/view/Challenge/ThirdProvenanceChallenge` (accessed December 20, 2009)
38. Data to Insight Center, `http://pti.iu.edu/d2i/provenance-karma` (accessed January 2011)
39. RabbitMQ Messaging System, `http://www.rabbitmq.com` (accessed July 2011)
40. The WS-BPEL Extension for People (BPEL4People), Version 1.0 Specification, `http://www.ibm.com/developerworks/webservices/library/specification/ws-bpel4people` (accessed December 2011)
41. Cao, B., Plale, B., Subramanian, G., Robertson, E., Simmhan, Y.: Provenance Information Model of Karma. In: IEEE Third Int'l Workshop on Scientific Workflows (SWF 2009), Los Angeles, CA (July 2009)
42. Mukhi, N.K.: Monitoring Unmanaged Business Processes. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 44–59. Springer, Heidelberg (2010)
43. Doganata, Y., Curbera, F.: Effect of Using Automated Auditing Tools on Detecting Compliance Failures in Unmanaged Processes. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 310–326. Springer, Heidelberg (2009)
44. Plale, B., Cao, B., Aktas, M.: Provenance Collection of Unmanaged Workflows with Karma. Journal Manuscript Accepted with Revisions (July 2011)
45. Shirasuna, S.: A Dynamic Scientific Workflow System for the Web Services Architecture. PhD thesis, Indiana University (September 2007)
46. Gil, Y., Ratnakar, V., Deelman, E., Mehta, G., Kim, J.: Wings for Pegasus: Creating Large-Scale Scientific Applications Using Semantic Representations of Computational Workflows, pp. 1767–1774. AAAI (2007)
47. Kim, J., Gil, Y., Spraragen, M.: Principles for interactive acquisition and validation of workflows. J. Exp. Theor. Artif. Intell. 22(2), 103–134 (2010)
48. Leake, D.B.: Case-Based Reasoning in Context: The Present and Future. In: Leake, D.B. (ed.) Case-Based Reasoning: Experiences, Lessons, and Future Directions, pp. 1–35. AAAI Press/MIT Press (1996)
49. de Mántaras, R.L., McSherry, D., Bridge, D.G., Leake, D.B., Smyth, B., Craw, S., Faltings, B., Maher, M.L., Cox, M.T., Forbus, K.D., Keane, M.T., Aamodt, A., Watson, I.D.: Retrieval, reuse, revision and retention in case-based reasoning. Knowledge Eng. Review 20(3), 215–240 (2005)
50. Leake, D.B., Kendall-Morwick, J.: Towards Case-Based Support for e-Science Workflow Generation by Mining Provenance. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 269–283. Springer, Heidelberg (2008)
51. Leake, D., Kendall-Morwick, J.: Four Heads Are Better than One: Combining Suggestions for Case Adaptation. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS, vol. 5650, pp. 165–179. Springer, Heidelberg (2009)

52. Cheah, Y.-W., Plale, B., Kendall-Morwick, J., Leake, D., Ramakrishnan, L.: A Noisy 10GB Provenance Database. In: Second International Workshop on Traceability and Compliance of Semi-Structured Processes, Clermont-Ferrand, France (2011) (in press)
53. Kendall-Morwick, J., Leake, D.: A Toolkit for Representation and Retrieval of Structured Cases. In: Proceedings of the ICCBR 2011 Workshop on Process-Oriented Case-Based Reasoning, Greenwich, U.K. (2011) (in press)
54. Ramakrishnan, L., Plale, B., Gannon, D.: WORKEM: Representing and Emulating Distributed Scientific Workflow Execution State. In: Proceedings of the 10th IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010), Melbourne Australia (2010)