Collaborative provenance for workflow-driven science and engineering

Altıntaş, İ.

Publication date
2011

3

Provenance Tracking for Scientific Data and Process

"A process cannot be understood by stopping it. Understanding must move with the flow of the process, must join it and flow with it."

– Frank Herbert, Dune (First Law of Mentat)

Provenance can be described in different ways depending on its context (Buneman *et al.* 2001, Simmhan *et al.* 2005, Zhao 2007). In the context of scientific workflows, *provenance* usually means the lineage and processing history of a data product, and the record of the process that led to it. Conducting computational experiments as scientific workflows provides a unique opportunity to capture and maintain such provenance information via the workflow environment. In this chapter, we focus on provenance for scientific workflows. Note, however, that the database community has also studied provenance extensively, i.e., in the context of database queries (e.g. see (Buneman *et al.* 2001, Cheney *et al.* 2009, Cheney 2010)). Below we will briefly compare with this related but different notion of provenance in databases.

Within the area of scientific workflows, we can distinguish two kinds of provenance information: By *data provenance* (often just 'provenance' in the following), we usually mean lineage and dependency information about a data product, that is, which other data items contributed to it, and through which "process invocations" (workflow steps). In contrast, by *workflow evolution provenance*, we mean the history of changes of a workflow during workflow design and development. For example, the Vistrails system (Callahan *et al.* 2006) has a particularly well developed system to capture changes during the incremental (exploratory) design of a workflow, making it easy for the user to jump between different versions of the workflow, undo design steps, parameter settings, etc.

[0]This chapter is based on (Altintas *et al.* 2006*a*), (Altintas 2008), (Crawl and Altintas 2008), (Ludäscher *et al.* 2008), (Moreau *et al.* 2008), (Mouallem *et al.* 2010) and (Anand *et al.* 2010*c*) co-authored by Altintas.

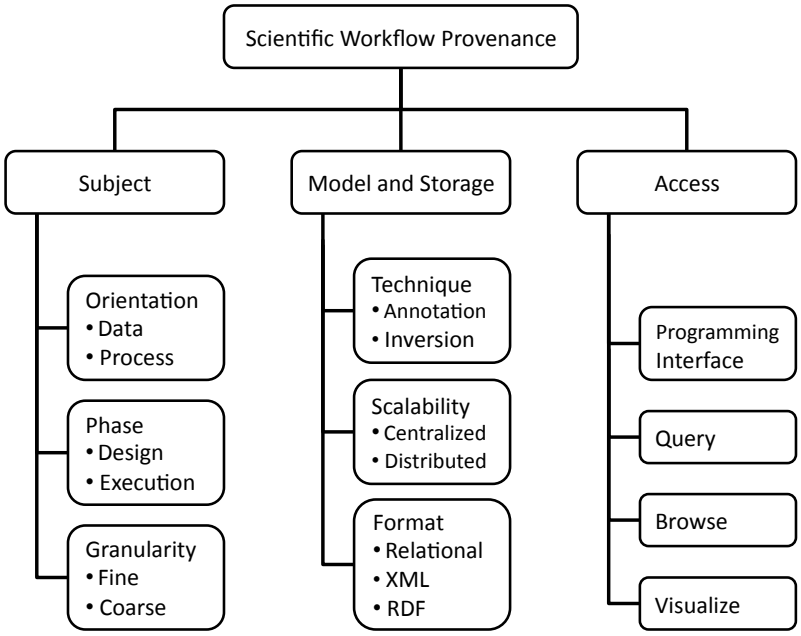

Figure 3.1: A simplified taxonomy of provenance system characteristics in scientific workflows.

Scientific workflows are repeatable patterns of computational activities, typically designed iteratively by a user and ran multiple times by one or more users. Thus, information on data collection, data usage, and especially the computational outcome of a scientific workflow provides a rich source for conducting similar scientific studies in the future (Miles *et al.* 2007). Capturing this provenance information for computational experiments and simulations is therefore a significant advantage of using scientific workflows. Many scientific workflow systems today provide provenance recording functionality for this purpose (Davidson and Freire 2008). A taxonomy of provenance in scientific workflow systems was proposed in (Serra da Cruz *et al.* 2009).

Figure 3.1 provides a simplified taxonomy of characteristics of provenance systems used by scientific workflows based on the taxonomies that were discussed in (Simmhan *et al.* 2005, Zhao 2007, Serra da Cruz *et al.* 2009). The *subject* branch represents the type of information that is collected, e.g., granularity and orientation of provenance data during design or execution of a workflow. Provenance information can be *data-oriented* where the system gathers the lineage metadata explicitly about the data product or *process-oriented* where the system gathers metadata about data lineage through deriving processes, e.g., a workflow. The subject data can then be analyzed and mapped into data models for different systems as a precondition to provenance information collection, e.g., (Altintas *et al.* 2006*a*, Zhao *et al.* 2006*a*, Gil *et al.* 2006, Simmhan *et al.* 2006*a*) and stored in different formats in distributed or centralized archives. This is captured by the *model and storage* branch in Figure 3.1. The

model and storage can be done using the *annotation technique* marking up existing source data, or *inversion technique* (Buneman *et al.* 2000, Gadang *et al.* 2008) recreating source data from collected information. *Access* branch in Figure 3.1 refers to the usage of the modeled and stored provenance information to answer different users' queries on different aspects of the workflows. Note that provenance information may be accessed via application programming and query interfaces, sometimes in combination with browsing a visualization of the provenance graph (Anand *et al.* 2010*a*). The provenance modeling, storage and access in scientific workflow management systems will be discussed in more detail later in this chapter.

Some provenance architectures focused on efficient and easy querying of the collected provenance as a part of the provenance support architecture achieving high scalability in recording and querying provenance through a loosely-coupled publish-subscribe architecture (Simmhan *et al.* 2006*a*) and reproduction of data products, or can be queried with application-specific semantics (Zhao *et al.* 2006*b*). The PrIME methodology (Miles *et al.* 2009) specifies a software engineering technique to design provenance-aware software applications that interact with a provenance middleware layer. PASOA/PReServ implements the the Provenance Recording Protocol (PReP) (Groth *et al.* 2005) for recording (and querying) provenance.

In the database community, provenance has been studied extensively and notions such as why-, where-, and how-provenance have been introduced (Buneman *et al.* 2001). Similar to scientific workflow provenance, the question in database provenance is also to "trace back" a result data item to those input data items which have (or could have) "contributed" to the result. One can view a database query Q (defined e.g. as a composite of relational operations) as a special "workflow" that takes a database instance D as input, returning answer tuples $Q(D)$ as a result. However, there are also significant differences: In scientific workflows, individual components are usually treated as "black boxes", i.e., little or nothing may be known about these operations. A provenance recorder in a scientific workflow system thus typically records the data items consumed and produced by these black boxes. Unlike workflows and workflow components, queries and query operators in databases can be viewed as "white boxes", i.e., whose semantics is completely known. For this kind of "white-box provenance", static analysis and inference techniques can often be applied to infer some provenance information from a given query Q, database instance D, and the output $Q(D)$, i.e., without recording runtime provenance information ((Cheney *et al.* 2009) calls this "lazy provenance"). In contrast, since little or no semantics is available for black-box components in scientific workflows, runtime recording of provenance (a.k.a. "eager provenance" in the terminology of (Cheney *et al.* 2009)) is inevitable in scientific workflows.

3.1 Life-cycle of Scientific Workflow Provenance

Provenance information related to scientific workflows is acquired and used in a variety of ways. The life-cycle of provenance information from the collection of provenance information to its different usages is shown in Figure 3.2. This life-cycle naturally starts with scientific workflow design and execution. Different users need different parts of this infor-

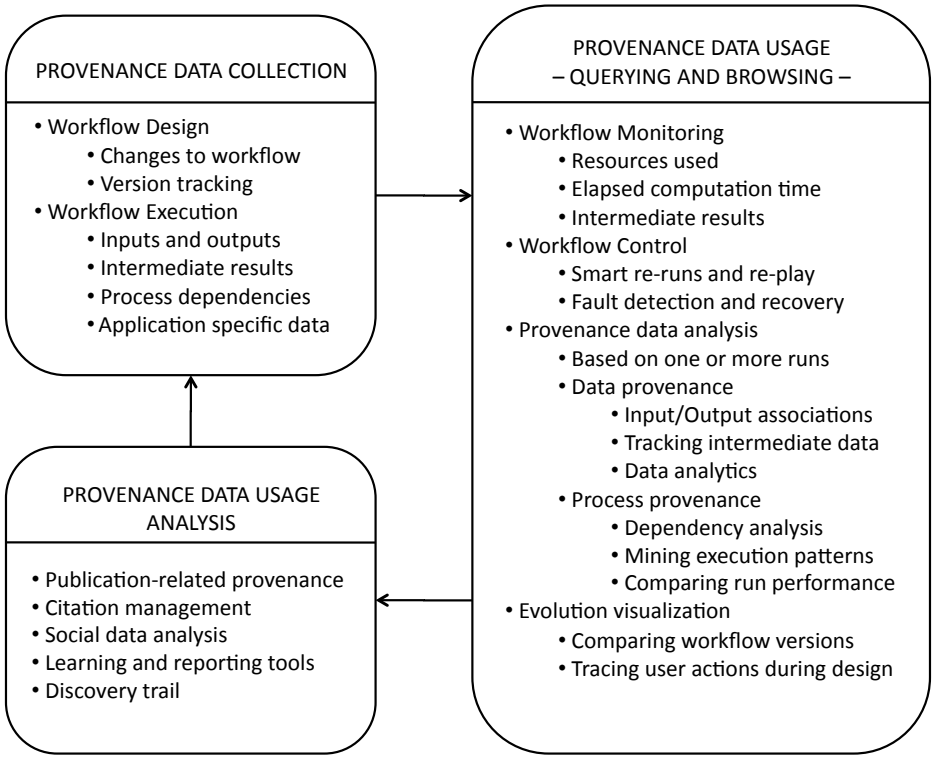

Figure 3.2: Life-cycle of scientific workflow related provenance information from its collection to the analysis of different usages of the collected information.

mation. Users of provenance data include the workflow developers who might want to record their exploratory work, workflow users who keep a trace of their runs, scientific dashboards that help execute and monitor a workflow, and other systems that draw on this data. Different workflow users need information about different phases of the workflow. In addition, different users have different data models and storage access requirements. To best serve these different users, provenance recorders need to allow for customized data collection through parametric interfaces. The collected information could be used for evaluation of the results as well as for mining different patterns during workflow design. Three main stages of this life-cycle can be identified as follows:

- **Provenance Collection.** Since workflow developers often change a workflow while experimenting with different computational tools and multiple scientific datasets, provenance recording can start during the design phase of the workflow. Capturing these user actions is important since it records what did and did not work, as well as how the workflow developer came up with the final workflow. The collection of provenance information continues during the experiment preparation (parameter binding) and exe-

cution of the workflow. The information related to design and execution of a workflow can be categorized (Davidson and Freire 2008) as *prospective* (workflow definition), *retrospective* (workflow products), and *user-defined* (user annotations on the executed workflow and its products). The third aspect of collecting scientific workflow provenance information that often gets ignored is collection after the workflow results and the provenance information are published. It is important for the publisher to know the citations for the results and statistics on how the collected information was used. Only after knowing this information, can one verify the scientific impact the workflow has made. This part of the provenance collection information is depicted by the arrow from the data usage analysis box to the data collection box in Figure 3.2.

- **Provenance Usage.** Once the provenance data is collected, it proves to be useful in many contexts. The collected data could be analyzed to query input and output associations, to verify results and recover from execution failures. Different types of provenance information analysis based on the design and one or more runs of the same workflow are listed in Figure 3.2. The collected information on the workflow design could be used both to analyze it and to visualize the evolution of the workflow as demonstrated in the Vistrails system (Freire *et al.* 2006). In addition, provenance information on data-dependencies could be used to re-play the previous executions and perform smart re-runs.

- **Provenance Usage Analysis.** This step in the life-cycle of scientific workflow provenance relates to how the collected provenance information was used, providing a high-level view across provenance archives and workflow systems. A requirement for having such an analysis is achieving interoperability between provenance collection systems through shared models. Another requirement is development of publication systems that can provide links to the provenance of published results, even reproducible documents (Mesirov 2010).

3.2 Modeling and Storing Scientific Workflow Provenance

Several provenance models have been proposed by the scientific workflow community and were summarized in surveys that have been conducted on the modeling aspect of provenance data (Bose and Frew 2005, Simmhan *et al.* 2005, Davidson and Freire 2008). All of these models support various levels of information related to workflow design and execution. Various provenance recording and storage systems have been implemented in the recent years based on these different models.

The Kepler scientific workflow system provides a Provenance Recorder (KPR) (Altintas *et al.* 2006*a*) that records information about workflow definition, inputs and customization parameter of workflow executions along with the products associated to these executions. KPR also has a plug-in interface for new data models, metadata formats and storage destinations, and extension points for supporting workflow evolution provenance. Recorded information

on executions of a workflow have also been used for a prototype smart re-run system (Altintas *et al.* 2006*a*) and fault-tolerance (Crawl and Altintas 2008) in Kepler, which helps to re-run a workflow. The Taverna scientific workflow system also allows for recording provenance information on data, process, knowledge and organization using Semantic Web technologies (Zhao *et al.* 2006*a*, Stevens *et al.* 2007). In Vistrails, the provenance feature focuses on experiment design as a unique feature (Freire *et al.* 2006). The Karma provenance model (Cao *et al.* 2009) to record execution information and higher level process details was implemented in a framework (Simmhan *et al.* 2006*b*) that uses the publish/subscribe architecture for propagating provenance activities. The Wings-Pegasus system (Kim *et al.* 2008) uses semantic representations to describe and map workflows to available computing resources, and tracks application-level provenance information generated during this process. The Swift system (Gadelha Jr. *et al.* 2010) uses an extension of the Virtual Data Language (VDL) (Clifford *et al.* 2008) to track the provenance of the computational procedure(s) that must be executed to materialize the dataset, the runtime logs produced by the execution of the computations, and optional metadata annotations that associate application semantics with data and procedures. In (Golbeck and Hendler 2007), the authors present a scientific workflow provenance model where the processing steps, e.i., services in this case, output RDF files that represent metadata about their execution as well as the provenance of the output files. The REDUX system (Barga and Digiampietri 2008) provides a layered provenance model on top of the Windows workflow engine where the data stored in the relational database allows navigation from an abstract model of the experiment to instance data collected during a specific experiment run.

Most of the existing provenance approaches store provenance for a single runs, and do not capture or maintain associations across runs. These models also don't capture any collaborative relationships that happens during design and execution of workflows. The framework described in (Bowers *et al.* 2007) records associations between multiple related workflow runs. Further work is need based on capturing associations not only across workflow runs, but also across users, where users play an active role of publishing data, or publishing workflows, or executing workflow runs. The collaborative provenance ideas discussed in this thesis focus on the provenance across scientific workflow executions and systems.

As demonstrated by the diversity of the systems summarized here, many e-Science systems have provenance tracking capability. This is demonstrated by the provenance challenges[1] and lead to a discussion on how to exchange the provenance information between multiple workflow systems, resulting in the Open Provenance Model (OPM) (Moreau *et al.* 2010).

3.2.1 Open Provenance Model

OPM is a model of provenance that is designed to allow provenance information to be exchanged between systems through a shared provenance model. The model encourages the

[1] Provenance Challenges website: http://twiki.ipaw.info/bin/view/Challenge/

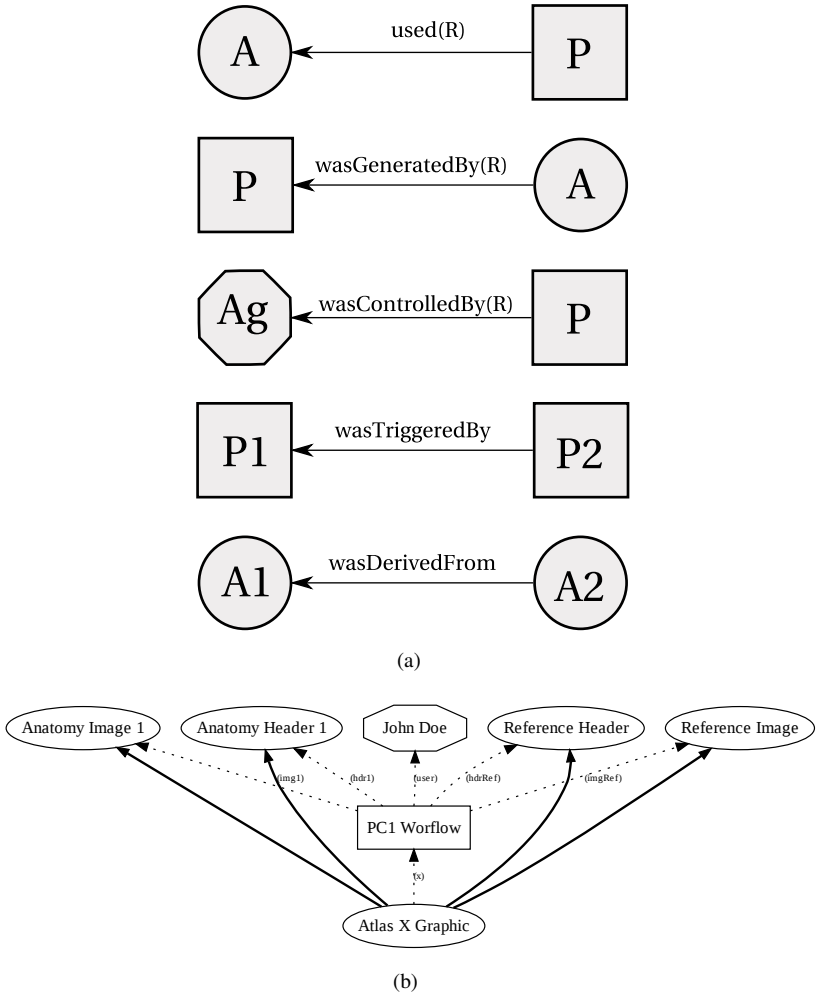

Figure 3.3: (a) Edges in the Open Provenance Model (OPM) from effects to causes; and (b) An OPM graph for the "Atlas X Graphic Workflow" implemented for the First Provenance Challenge.

provenance community to build and share tools that operate on this common provenance model in a technology-agnostic manner (Moreau *et al.* 2010).

From a scientific workflow based perspective, the Open Provenance Model allows for capturing causal dependencies between different entities in a workflow run as a directed graph consisting of nodes to express artifacts (e.g., digital products), processes (e.g., a series of actions) and agents that catalyze and control the processes.

The dependencies between the nodes in an OPM graph are captured by relationships to express, as shown in Figure 3.3(a) (Figure 1 in (Moreau *et al.* 2010)):

- which artifacts were used by which processes (*used*)
- which processes generated which artifacts (*was generated by*)
- which processes were caused by which agents (*wasControlledBy*)
- which processes depended on which processes (*wasTriggeredBy*)
- which artifacts depended on which artifacts (*wasDerivedFrom*)

Example. Figure 3.3(b) (Figure 2 in (Moreau *et al.* 2010)) shows an OPM-based provenance graph for an execution of the "Atlas X Graphic" workflow of the First Provenance Challenge[2]. This graph demonstrates the OPM nodes at the workflow-execution level along with the "used", "wasGeneratedBy" and "wasDerivedFrom" edges between the nodes. In this example, the process *PC1 Workflow wasControlledBy* the agent *John Doe*, and *used* the artifacts *Anatomy Image 1*, *Anatomy Header 1*, *Reference Header* and *Reference Image* as input. The artifact *Atlas X Graphic wasGeneratedBy* the process *PC1 Workflow*. The "wasDerivedFrom" edges between the artifacts are represented explicitly by plain lines in the figure, since the data dependency relationship cannot be implicitly inferred from an OPM graph.

All the OPM dependency relationships refer to execution that happened in the past and are causal between the source and destination. These relationships can be extended using *Annotations* (part of the OPM specification) that allow edges to be further subtyped. The latest description of OPM also allows for *Profiles* to develop further annotations and practices on top of the top-level representation framework. OPM Profiles are intended to allow for specializations of OPM consisting of a mandatory unique global identifier for the profile and an optional controlled vocabulary for annotations.

Current Status of OPM. Although it is designed as a workflow-independent provenance standard, OPM is a community-effort that has the potential to achieve interoperability of provenance information between workflows. OPM represents the data products and past computations that derived these products as a directed acyclic graph. Through the Provenance Challenges, many of the above-mentioned workflow systems either built extensions to store provenance information in OPM or were able to port their models into OPM. Given the increased interest in OPM, the W3C established the Provenance Incubator Group[3] as part of the W3C Incubator Activity with a charter to "provide a state-of-the art understanding and develop a roadmap in the area of provenance and possible recommendations for standardization efforts". This group recently published their final report[4] and the activity was recommended as a W3C provenance standard working group.

Another provenance model that influenced the discussions of the W3C incubator group was the Proof Markup Language (PML) (McGuinness *et al.* 2007, Graves 2010). Through

[2]First Provenance Challenge website: http://twiki.ipaw.info/bin/view/Challenge/FirstProvenanceChallenge

[3]W3C Provenance Incubator Group wiki: http://www.w3.org/2005/Incubator/prov/wiki/Main_Page

[4]W3C Provenance Incubator Group Final Report: http://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/

three ontology modules for provenance (for annotation of processes, data products, etc.), justification (for annotation of data derivations) and trust (for annotation of trust relationships between provenance and justification), PML is used to capture workflow provenance in multiple application areas (Zhenning *et al.* 2009).

3.3 Querying and Browsing Provenance

In addition to the ability to model and store provenance data efficiently, the existing systems use different querying techniques based on system specific APIs, logical programming and standard querying languages. e.g., SQL, XQuery, and SPARQL. However, the existing provenance querying methods are mostly specific to the data model and storage system used by the provenance systems (Cohen *et al.* 2006).

Kepler provenance framework (KPR) provides a querying API that allows for querying any data model that KPR allows for including a SQL-based querying interface for relational data, a Java API for the data collected in the file system and an XQuery-based interface for provenance data that conforms to OPM. The REDUX system implements SQL-based querying since the provenance data is stored in a relational system (Barga and Digiampietri 2008). The Karma provenance framework offers a querying component (Simmhan *et al.* 2008) based on based on XQuery (and XPath) to query the XML-based provenance data that is collected by the system. Taverna provenance is queried through a SPARQL interface specific to this system (Missier *et al.* 2010*b*). Similarly, Wings-Pegasus (Kim *et al.* 2008) system offers a SPARQL interface for the RDF-based workflow instantiation interface and, additionally, provides a SQL-based interface for the relational workflow execution provenance data.

A Prolog-based querying interface for CoMaD (McPhillips *et al.* 2006) is presented in (Bowers *et al.* 2008*a*). (Zhao and Lu 2008) presents a Frame Logic (Kifer *et al.* 1995) based approach to querying based on a set of characteristics across provenance models. Extending the Lorel query language, PQL (Holland *et al.* 2008) picks up on where the common solutions (relational, XML, RDF) fall short and defines a semistructured data and query model for handling provenance.

In addition, browsing and visualizing provenance have gained interest in the recent couple of years. (Margo and Seltzer 2009) talks about the case for browsing provenance data in our increasingly networked world. In this context, provenance is the metadata that captures the causality and lineage of data obtained via the browser. An example of provenance browsing in the scientific workflow context has been implemented in Kepler (Anand *et al.* 2010*a*) where the users can visualize and navigate through data dependencies and provenance information created during a workflow run. Vistrails (Callahan *et al.* 2006) system also allows for browsing of provenance data through a graphical user interface including an interactive display for scientific workflow evolution.

3.3.1 Query Language for Provenance

Provenance queries often require computing transitive closures over dependency relations, and expressing such queries using standard approaches is typically done using recursion or stored procedures (Heinis and Alonso 2008, Anand *et al.* 2010*b*). Expressing such queries is both cumbersome and error-prone, and requires considerable user expertise. High-level languages such as Query Language for Provenance (QLP) (Anand *et al.* 2009*c*) provide a separation between the logical provenance model and its underlying physical representation, which allows for the use of different representation schemes and additional optimization techniques.

QLP allows users to formulate provenance queries by providing specialized operators for expressing both structural, i.e., parent-child, and lineage, i.e., data dependency, queries, and their combination ("hybrid" queries). QLP queries return sets of nodes that are closed under lineage relationships where lineage dependencies form provenance subgraphs. These subgraphs are "provenance preserving", i.e., result of a provenance lineage graph is a subgraph that can be queried further.

Different constructs are used to query distinct dimensions of the flow graph representing: (i) lineage relations among nodes and invocations; (ii) flow relations among input and output structures of invocations; and (iii) structural relations among nodes within and across data structures.

3.4 Comparing Different Scientific Workflow Provenance Approaches

Based on the summary of different provenance modeling, storage and querying approaches discussed, we present a table of the approaches in different systems in Table 3.1. Although this table might evolve over time, it show some commonalities and trends between different systems. This table confirms that current scientific provenance models are fine-grained, process-oriented and use the annotation technique. Most systems focus on the execution phase of a scientific workflow, but a few systems recently started collecting provenance information about the design phase of a workflow. While there are different querying approaches, all of the systems use a query language that fits the format and the model of provenance.

A similar comparison was presented in (Serra da Cruz *et al.* 2009) along with a detailed discussion on an extended provenance taxonomy for scientific workflows.

Summary

Developing a framework that allows the gathering of provenance information targeted at keeping data and the associations of the data products is an area currently being researched and developed by many in the field. A taxonomy of provenance in scientific workflows, significant current efforts in the field, and the techniques and models for capturing and querying

Table 3.1: Summary of the characteristics of different scientific workflow provenance approaches.

	Kepler	**Taverna**	**Vistrails**	**Karma**	**Wings-Pegasus**	**Swift**	**REDUX**
Orientation	Process	Process	Process	Process	Process	Process and Data (VDL)	Process
Phase	Execution	Design and Execution	Design and Execution	Execution	Execution	Design and Execution	Execution
Granularity	Fine	Fine	Fine	Fine	Fine	Fine	Fine
Technique	Annotation	Annotation	Annotation	Annotation	Annotation	Annotation	Annotation
Scalability	Central	Central	Central	Distributed	Central	Central	Central
Format	Relational and XML	RDF	Relational	XML	Relational	Relational	Relational
Access	QLP and Programming Interface	SPARQL and Programming Interface	VTQL	XQuery	SPARQL and SQL	SQL	SQL

provenance were summarized in this chapter. Next, we review e-Science infrastructures that act as gateways for provenance information to be captured throughout the scientific process.