The Provenance of Electronic Data

Luc Moreau, Paul Groth, Simon Miles, Javier Vázquez-Salceda,
John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana,
Andreas Schreiber, Victor Tan, Laszlo Varga

December 7, 2007
Word count: 3005 cacm06.tex

Abstract

In the study of fine art, provenance refers to the documented history of some art object. Given that documented history, the object attains an authority that allows scholars to appreciate its importance with respect to other works, whereas, in the absence of such history, the object may be treated with some skepticism. Our IT landscape is evolving as illustrated by applications that are open, composed dynamically, and that discover results and services on the fly. Against this challenging background, it is crucial for users to be able to have confidence in the results produced by such applications. *If the provenance of data produced by computer systems could be determined as it can for some works of art, then users, in their daily applications, would be able to interpret and judge the quality of data better.* We introduce a provenance lifecycle and advocate an open approach based on two key principles to support a notion of provenance in computer systems: documentation of execution and user-tailored provenance queries.

Introduction

Provenance is already well understood in the study of fine art where it refers to the documented history of some art object. Given that documented history, the object attains an authority that allows scholars to understand and appreciate its importance and context relative to other works. Art objects that do not have a proven history may be treated with some skepticism by those who study them.

Such an idea, transposed into computer systems, has immediate, practical use: if the provenance of data produced by computer systems could be determined as it can for some works of art, then users would be able to understand how documents were assembled, how simulation results were determined, or how financial analyses were carried out. Thus, to accomplish such a vision, computer applications need to be transformed into what we term *provenance-aware applications*, for which the provenance of data may be retrieved, analyzed and reasoned over.

The Oxford English Dictionary defines provenance as: "(i) the fact of coming from some particular source or quarter; origin, derivation; (ii) the history or pedigree of a work of art, manuscript, rare book, etc.; concretely, a record of the ultimate derivation and passage of an item through its various owners." Hence, we can regard provenance as the derivation from a particular source to a specific state of an item. The description of such a derivation may take different forms, or may emphasize different properties according to interest. For instance, for a work of art, provenance usually identifies its chain of ownership; alternatively, the actual state of a painting may be understood better by studying the different restorations it underwent.

The above dictionary definition also identifies two distinct understandings of provenance: first, *as a concept*, it denotes the source or derivation of an object; second, *more concretely*, it is used to refer to a record of such a derivation. Against this background, a computer-based *representation* of provenance is crucial for users to perform analysis and reasoning, and decide whether they have confidence in electronic data.

In this article, we introduce the provenance lifecycle, summarising key principles underpinning existing provenance systems. We then examine an open data model for describing how applications are executed; in this context, provenance is seen as a user query over such descriptions. The vision of provenance-aware applications is illustrated over a concrete example in healthcare management, before we contrast it with existing systems.

Lifecycle of Provenance in Computer Systems

Both the scientific and business communities [FKNT03, Bur00] have adopted the service-oriented architectural (SOA) style, which allows services to be discovered and composed dynamically. SOA-based applications become more dynamic and open, but equally have to satisfy new requirements, both in e-Science and business.

In an ideal world, e-Science end-users would be able to: reproduce their results by replaying previous computations, understand why two seemingly identical runs with the same inputs produce different results, and find out which data sets, algorithms, or services were involved in the derivation of their results.

In business and e-Science, some users, reviewers, auditors, or even regulators have to verify that the process that led to some result is compliant with specific regulations or methodologies; they have to prove that results are derived independently of services or databases with given license restrictions; and, they need to establish that data was captured at source by instruments that possess some precise technical characteristics.

While some users need to perform such tasks today, they cannot do so, or they can do it only imperfectly, because the underpinning principles have not been investigated, and systems have not been designed to support such requirements. A key observation is that electronic data does not typically contain the necessary historical information that would help end-users, reviewers, or regulators make the necessary verifications. Hence, there is a need to capture extra information — which we name *process documentation* — that describes what actually occurred at execution time. *Process documentation is to electronic data what a record of ownership is to a work of art.* Provenance-aware applications create process documentation and store it in a *provenance store*, the role of which is to offer a long-term persistent, secure storage of process documentation (cf. Figure 1). This logical role accomodates various physical deployments: for instance, a provenance store can be a single, autonomous service or, to be more scalable, it can be a federation of distributed stores.

Once process documentation has been recorded, the provenance of data results can be re-

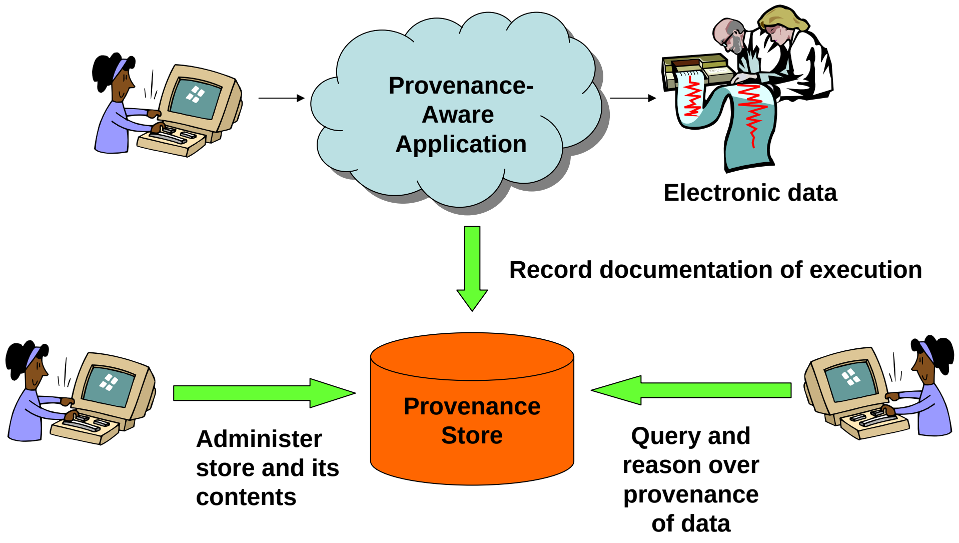
Provenance-Aware Application
Electronic data
Record documentation of execution
Administer store and its contents
Provenance Store
Query and reason over provenance of data

Figure 1: Provenance Lifecycle

trieved by querying the provenance store, and analyzed to suit the user's needs. Finally, over time, the provenance store and its contents may need to be managed, maintained, or curated. In summary, the provenance lifecyle consists of four different phases: *(i)* creating, *(ii)* recording, *(iii)* querying, and *(iv)* managing, all of which provenance systems should cater for.

An Open Model for Process Documentation

For many applications, process documentation cannot be produced in a single, atomic burst, but instead its generation must be interleaved continuously with execution. Given this, it is necessary to distinguish a specific item documenting part of a process from the whole process documentation. We see the former — referred to as a *p-assertion* — as an assertion made by an individual application service involved in the process. Thus, the documentation of a process consists of a set of p-assertions made by the services involved in the process.

In order to minimize its impact on application performance, documentation needs to be structured in such a way that it can be constructed and recorded autonomously by services, on a piecemeal basis. Otherwise, should synchronizations be required between these services to agree on how and where to document execution, application performance may suffer dramatically. To satisfy this design requirement, various kinds of p-assertions have been identified, which we expect applications to adopt in order to document their execution. Figure 2 illustrates a computational service sending and receiving messages, and creating p-assertions describing its involvment in such activity.

In SOAs, interactions consist of messages exchanged between services. By capturing all interactions, one can analyze an execution, verify its validity, or compare it with other executions. Therefore, process documentation includes *interaction p-assertions*, where an interaction p-assertion is a description of the contents of a message by a service that has sent or

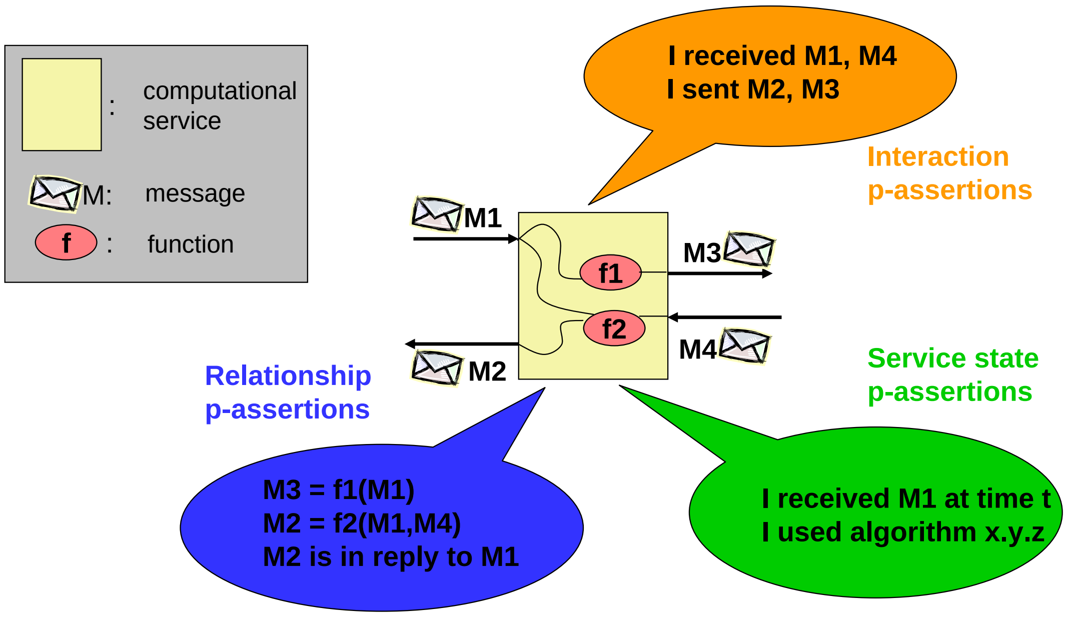

Figure 2: Kinds of p-assertions made by a computational service

received that message.

Generally, whether a service returns a result directly or calls other services, the relationship between its outputs and inputs is not explicitly represented in messages themselves, but can be understood only by an analysis of the service's business logic. To promote openness and generality, we do not make any assumption about the technology used by services to implement their business logic (such as source code, workflow language, etc.). Instead, we place a requirement on services to provide some information, in the form of *relationship p-assertions*: a relationship p-assertion is a description, asserted by a service, of how it obtained output data sent in an interaction by applying some function, or algorithm, to input data from other interactions. (In Fig. 2, output message M3 was obtained by applying function f1 to input M1.)

With these two kinds of p-assertions, process documentation as a whole is greater than the sum of its individual parts. Indeed, while p-assertions are simple pieces of documentation that can be produced by services autonomously, interaction and relationship p-assertions taken together capture an explicit description of the flow of data in a process: interaction p-assertions denote data flows *between* services, whereas relationship p-assertions denote data flows *within* services. Such data flows capture the causal and functional data dependencies that occur in execution and, in the most general case, constitute a directed acyclic graph (DAG). For a specific data item, the data flow DAG indicates how it is produced and used. The data flow DAG is thus a core element of provenance representation, but it is not the only one, as we now explain.

Beyond the flow of data in a process, internal service states may also be necessary to understand non-functional characteristics of execution, such as performance or accuracy of services, and therefore the nature of the result they compute. Hence, we define a *service state p-assertion* as documentation provided by a service about its internal state in the context of a specific interaction. Service state p-assertions can be extremely varied: they may include the amount of disk and CPU time a service used in a computation, its local time when an action occurred, the floating point precision of the results it produced, or application-specific state descriptions.

In order for provenance-aware applications to be interoperable, it is crucial that the process documentation they respectively produce be structured according to a shared data model. Therefore, the novelty of our approach is the openness of the proposed model of documentation [GJM$^+$06], which is conceived to be independent of application technologies [MGBM07]. Taken together, these characteristics allow process documentation to be produced autonomously by application services, and be expressed in an open format, over which provenance queries can be expressed.

Querying the Provenance of Electronic Data

Provenance queries are user-tailored queries over process documentation aimed at obtaining the provenance of electronic data. In this context, a first challenge is to characterize the data item that is of interest to the user. Indeed, since data can be mutable, its provenance, i.e. history, can vary according to the point in execution from which a user wishes to find its provenance. A provenance query, therefore, needs to identify a data item with respect to a given documented event (i.e., sending or receiving a message).

The full details of everything that ultimately caused a data item to be as it is could potentially be very large. For example, the full provenance of an experiment's results would include a description of the process that produced the materials used in the experiment, the provenance of any materials used in producing those materials, the devices and software used in the experiment and their settings, etc. Ultimately, should documentation be available, we would include details of processes leading back to the beginning of time, or at least the epoch of provenance awareness.

Thus, users need to be able to express the scope of interest in a process, by means of a provenance query. Such a query then essentially performs a reverse graph traversal over the data flow DAG and terminates according to the query-specified scope; the query output is a DAG subset. Scoping can be based on types of relationships, intermediary results, services, or subprocesses [GJM$^+$06].

Example: Provenance in Healthcare Management

In order to illustrate the proposed approach, we consider a healthcare management application. The Organ Transplant Management (OTM) system manages all the activities pertaining to organ transplants across multiple Catalan hospitals and their regulatory authority [AVSK$^+$06]. OTM consists of a complex process, involving the surgery itself, but also a wide range of other activities, such as data collection and patient organ analysis, which all have to comply with a set of regulatory rules. Currently, OTM is supported by an IT infrastructure that maintains records allowing medical personnel to view (and edit) a given patient's local file within a given institution or laboratory. However, the system does not connect records, nor capture dependencies between them. It does not allow external auditors or patients' families to analyze or understand how decisions are reached.

By making OTM provenance-aware, powerful queries that were not possible before can be supported, such as: find all doctors involved in a decision, find the blood test results that were involved in a donation decision, find all data that led to a decision to be taken. Such functionality can be made available not only to the medical profession but also to regulators or families.

We limit ourselves to a small, simplified subset of the OTM workflow, namely the process leading to the decision of donating an organ. As a hospitalized patient's health declines and in anticipation of a potential organ donation, one of the attending doctors requests the full health record for the patient and sends a blood sample for analysis. Through a user interface (UI), these requests are made by the attending doctor and passed to a software component (Donor Data Collector) responsible for collecting all the expected results. After brain death is observed and logged into the system, if all requested data and analysis results have been obtained, a doctor is asked to make a decision about the donation of an organ. The decision, i.e., the outcome of the doctor's medical judgment based on the collected data, is explained in a report that is submitted as the decision's justification.

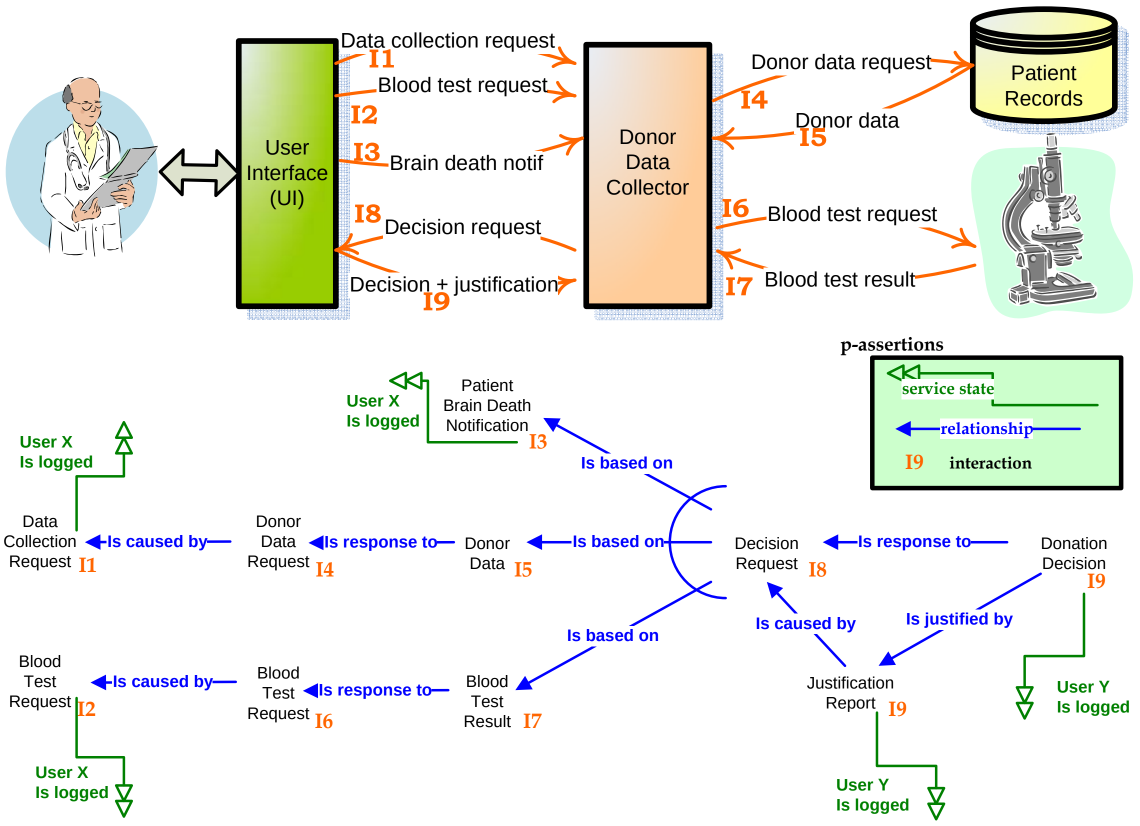

Figure 3: Provenance DAG of Donation Decision

Figure 3 (top) displays the components involved in this scenario and their interactions. The UI sends requests (I1, I2, I3) to the Donor Data Collector service, which gets data from the patient records database (I4, I5) and analysis results from the laboratory (I6, I7), and finally requests a decision (I8, I9).

To make OTM provenance-aware, we augment it with a capability to produce an explicit representation of the process actually taking place. This includes p-assertions for all interactions (I1 to I9), relationship p-assertions capturing dependencies between data, and service state p-assertions. In Figure 3 (bottom), we find the DAG that represents the provenance of a donation decision, made of relationships p-assertions produced by provenance-aware OTM. DAG nodes denote data items, whereas DAG edges (in blue) represent relationships such as data dependencies (is based on, is justified by) or causal relationships (in response to, is caused by). Each data item is annotated by the interaction in which it occurs. Furthermore, the UI asserts a service state p-assertion, for each of its interactions, about the user who is logged into the system.

Over such documentation, we can issue provenance queries that navigate the provenance graph and prune it according to the querier's needs. For instance, from the graph, we can derive that users X and Y are both causing a donation decision to be reached. Figure 3 is small, but in real life examples with vast amount of documentation, users benefit from a powerful and accurate provenance query facility.

Existing Systems

The approach presented in this article has been derived from an extensive requirement analysis [MGBM07], which has resulted in a complete architectural specification [GJM$^+$06]. It is being used as the basis for writing a standardization proposal for data models and interfaces. Its open approach allows complex distributed applications, possibly involving multiple technologies (such as Web Services, command line executables, monolithic executables), to be documented. It also allows complex provenance queries to be expressed, identifying data and scoping processes independently of technologies used.

The Virtual Data System (VDS) [CFV$^+$07] and myGrid [ZGST07] are execution environments for scientific workflows, which also provide support for provenance. Their focus is on producing documentation from a workflow enactor's viewpoint, using data models that are compatible with p-assertions. They assume their respective workflow language, which allows them to obtain compact process documentation. By adopting an open data model for process documentation similar to the one advocated here, such systems could be integrated into heterogeneous applications for which provenance queries could be executed seamlessly.

The database community has also investigated provenance [BKT01, CWW00] but considered different assumptions; in particular, the existence of a query language is assumed, for which queries can be reversed to identify the origin of results. As in our approach, different kinds of provenance are perceived to be of value (cf. why and where provenance [BKT01]), which can be seen as specific specific instances of what we term provenance queries.

The Provenance Aware Storage System [SHBMR07] seeks to automatically produce documentation of execution by capturing file system events in an operating system. Like all other approaches, capturing small granularity documentation presents scalability and performance challenges, and deriving information at a suitable level of abstraction for the user can be difficult.

Conclusion

The IT landscape is evolving: where we used to have closed monolithic applications, we now face applications that are open, that are composed dynamically, and that discover results and services on the fly. Against this challenging background, users must be able to decide whether they have confidence in electronic data. Our vision is that such data must be accompanied by their provenance, which describes the process that led to their production.

To realise this vision, we have proposed an open approach, by which applications, irrespective of their technology, document their execution in an open data model, which can then be used to run provenance queries, tailored to user's needs. As scholars can appreciate works of art by studying their documented history, users will derive their confidence in electronic data from provenance queries.

References

[AVSK$^+$06] Sergio Alvarez, Javier Vázquez-Salceda, Tamás Kifor, László Varga, and Steven Willmott. Applying Provenance in Distributed Organ Transplant Management. In *International Provenance and Annotation Workshop (IPAW'06)*, volume 4145 of *Lecture Notes in Computer Science*, pages 28–36, 2006.

[BKT01] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and Where: A Characterization of Data Provenance. In *Proceedings of 8th International Conference on Database Theory (ICDT'01)*, volume 1973 of *Lecture Notes in Computer Science*, pages 316–330, London, UK, 2001. Springer.

[Bur00] Steve Burbeck. The tao of e-business services. Technical report, Emerging Technologies, IBM Software Group, October 2000.

[CFV$^+$07] Ben Clifford, Ian Foster, Jens-S. Voeckler, Michael Wilde, and Yong Zhao. Tracking provenance in a virtual data grid. *Concurrency and Computation: Practice and Experience*, 9999(9999), 2007. Published Online: Aug 21 2007 6:12AM.

[CWW00] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25(2):179–227, 2000.

[FKNT03] Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. *Grid Computing — Making the Global Infrastructure a Reality*, chapter The Physiology of the Grid — An Open Grid Services Architecture for Distributed Systems Integration, pages 217–249. Wiley Series in Communications Networking and Distributed Systems. John Wiley and Sons, Chichester, England, 2003.

[GJM$^+$06] Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, Victor Tan, Sofia Tsasakou, and Luc Moreau. D3.1.1: An Architecture for Provenance Systems. Technical report, University of Southampton, November 2006.

[MGBM07] Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of recording and using provenance in e-science experiments. *Journal of Grid Computing*, 5(1):1–25, 2007.

[SHBMR07] Margo Seltzer, David A. Holland, Uri Braun, and Kiran-Kumar Muniswamy-Reddy. Pass-ing the provenance challenge. *Concurrency and Computation: Practice and Experience*, 9999(9999), 2007. Published Online: Sep 11 2007 11:00AM.

[ZGST07] Jun Zhao, Carole Goble, Robert Stevens, and Daniele Turi. Mining taverna's semantic web of provenance. *Concurrency and Computation: Practice and Experience*, 9999(9999), 2007. Published Online: Aug 22 2007 11:15AM.