ws-prov-overview

Authors:
Steve Munroe, U. Southampton
Paul Groth, U. Southampton
Sheng Jiang, U. Southampton
Simon Miles, U. Southampton
Victor Tan, U. Southampton
John Ibbotson, IBM
Luc Moreau, U. Southampton

October 24, 2006

Overview of the Provenance Specification Effort

Abstract

This document provides an overview of a model of provenance along with a description of a family of specification documents that support the model. Important aspects of the model are specified within these documents in a detailed and clear manner that provides an unambiguous reference for developers.

Contents

1 Introduction

The importance of understanding the process by which a result was generated is fundamental to many real life applications (science, engineering, medicine, supply management, etc). Without such information, users cannot reproduce, analyse or validate processes or experiments. Provenance is therefore important to enable users, scientists and engineers to trace how a particular result has been arrived at.

Two common sense definitions consider provenance to be the derivation from a particular source to a specific state of an item. They are taken from the Oxford English Dictionary, and the Merriam-Webster Online Dictionary respectively and are presented below.

Definition 1 (OED Provenance Definition) (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.; concr., a record of the ultimate derivation and passage of an item through its various owners.* □

Definition 2 (MWO Provenance Definition) (i) *the origin, source;* (ii) *the history of ownership of a valued object or work of art or literature.* □

Both definitions are compatible since they regard provenance as the derivation from a particular source to a specific state of an item. The nature of the derivation, or history, may take different forms, or may emphasise different properties according to interest. For instance, for a piece 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.

From Definitions 1 and 2, we can also distinguish two different 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. We have identified a process in a service oriented architecture (SOA) as the execution of a workflow, which we broadly see as a specification of a given service composition. Hence, by having a description of the process that resulted in a data item, we can explain how such a data item has been obtained. Inspired by previous work [GLM04a, GLM04b, GLM04b, MGBM05, SM03], we propose the following definition of provenance, which makes explicit the notion of process.

Definition 3 (Provenance of a piece of data) *The provenance of a piece of data is the process that led to that piece of data.* □

In relation to the two common sense definitions of provenance, we note that Definition 3 is concerned with provenance as a concept. Ultimately, our aim is to specify a computer-based *representation* of provenance that allows us to perform useful analysis and reasoning to support a wide variety of applications.

Consequently, the provenance of a piece of data is to be represented in a computer system by some suitable documentation of the process that led to the data.

While specific applications determine the actual form that such documentation should take, we can identify several of its general properties. Documentation can be complete or partial (for instance, when the computation has not yet terminated); it can be accurate or inaccurate; it can present conflicting or consensual views of the actors involved; it can be descriptive or conceptual; and it can abstract more or less from reality.

In this document, we introduce a framework for computational provenance; a set of nine technical specifications that define the normative description of the provenance framework in terms of a SOA model and related XML definitions. These technical specifications, summarised in Figure 1, define the means by which:

- a computational representation of process documentation can be realised;
- process documentation can be recorded;
- process documentation can be queried;
- the recording and querying of process documentation can be made secure;
- process documentation can be recorded in distributed systems.

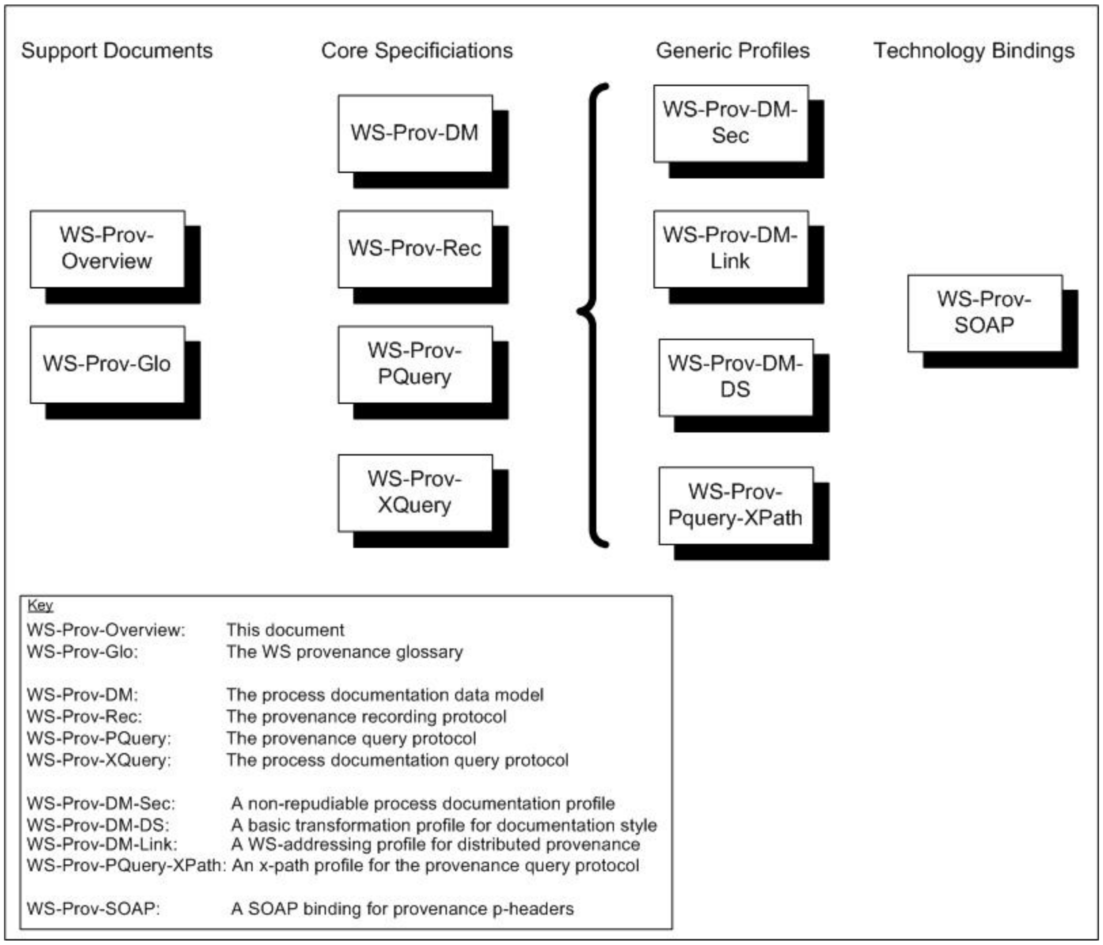
The family of documents comprise a set of 2 support documents, four documents that introduce and specify the core framework, four generic profiles that extend the basic framework and one example of a technology specific binding.

2 Overview of the Provenance Model

In this section we present a intuitive, non-technical view of provenance and the provenance model. In it we provide some context to the model before describing definitions of the different kinds of process documentation that go to form the basis of the model.

2.1 Context: Service Oriented Architectures

Given that our work predominantly focuses on Grid and Web Services, we summarise some relevant terminology in this section. We take the broad view that open, large-scale systems are typically designed using a *service*-oriented approach [MH05], usually referred to as service-oriented architectural style [Bur00]. As far as services are concerned, we do not intend to restrict ourselves to a specific technology; instead, we take services to be components that take inputs and produce outputs. Specifically, the following are all considered as "services" because they



WS-Prov-Overview:	This document
WS-Prov-Glo:	The WS provenance glossary
WS-Prov-DM:	The process documentation data model
WS-Prov-Rec:	The provenance recording protocol
WS-Prov-PQuery:	The provenance query protocol
WS-Prov-XQuery:	The process documentation query protocol
WS-Prov-DM-Sec:	A non-repudiable process documentation profile
WS-Prov-DM-DS:	A basic transformation profile for documentation style
WS-Prov-DM-Link:	A WS-addressing profile for distributed provenance
WS-Prov-PQuery-XPath:	An x-path profile for the provenance query protocol
WS-Prov-SOAP:	A SOAP binding for provenance p-headers

Figure 1: The family of specification documents

all take some inputs and produce some outputs: Web Service, Corba or RMI objects, command line program.

Such services are brought together to solve a given problem typically via a *workflow* that specifies their composition. With such a broad definition, we see that WS-BPEL, WSFL, VDL, Dagman's DAGs or Gaudi are all workflow frameworks capable of expressing the composition of services. Likewise, a script calling several command line commands is also regarded as a workflow.

In this abstract view, invocations of services take place using *messages* that are constructed in accordance with service *interface* specifications. Such messages take the form of SOAP messages for Web services. In the case of command line executables, we do not have explicit messages; instead, they take some explicit arguments potentially representing both inputs and outputs. We also see a memory shared by two threads as a way of implementing such message-passing mechanism; the message itself is the information stored in the shared memory.

In a service-oriented architecture (SOA), clients typically invoke services, which may themselves act as clients for other services; hence, we use the term *actor* to denote either a client or a service in a SOA. An actor that sends a message is referred to as a *sender*, whereas an actor that receives a message is known as a *receiver*. One message exchanged between a sender and a receiver is termed an *interaction*. Hence, a given interaction comprises two views: the sending of the message and its receiving. The running of an application programmed in a SOA style requires the execution of the workflow, which characterises composition of the services that belong to the application. Hence, the execution of a workflow is referred to as a *process*. Our definition of process, like the Unix notion of process, refers to an instance of a running program (workflow here). It has a beginning, and, if it is finite, it has an end.

At this stage of the specification, we do not make the distinction between resource and service [SRB06] since they are defined in the context of the specific Web Services technology. Our broad view of message allows us to include in a message the necessary reference to resources, as required by WSRF.

2.2 Representation of Provenance

In this section, we introduce the key elements that form the representation of provenance in a SOA.

In the previous section, we stated that provenance of a data item is to be represented in a computer system by some suitable documentation of the process that led to it. To this end, we distinguish a specific piece of information documenting some step of a process from the whole documentation of the process. The former shall be referred to as a *p-assertion*, which we define as follows.

Definition 4 (p-assertion) *A p-assertion is an assertion that is made by an actor and pertains to a process.* □

From this definition, we derive the notion of process documentation.

Definition 5 (Process Documentation) *The documentation of a process consists of a set of p-assertions made by the actors involved in the process.* □

Should the actors involved in the process be the only one to document it? The answer is yes. Indeed, if actors are not involved in the process, then no message has been sent to them. Hence, they cannot be aware of the process, and therefore could not possibly provide any documentation relevant to this specific execution.

We note that a given p-assertion may belong to the provenance representation of multiple pieces of data. When a p-assertion is created (and later recorded), it documents a step of a process in progress, which ultimately will lead to a piece of data. At the time of the p-assertion creation, we may not know the piece of data that will be produced; however, the p-assertion being recorded constitutes an element of the provenance representation of the data. For instance, when some quality wood is being transported in the Amazon forest, one may not know that it will be used for creating the frame for a future famous painting, still to be painted and framed.

Among all the p-assertions, we now introduce two kinds of p-assertions that allow us to capture an explicit description of the flow of data in a process: *interaction p-assertions* and *relationship p-assertions*.

Computer science has a long tradition of focusing on communications and interactions as a central concept used in the study and modelling of complex systems, e.g., programming language semantics, process algebra and more recently in biological systems models. In the context of SOAs, interactions consist of the messages exchanged between actors. By capturing all the interactions that take place between actors involved in the computation of some data, one can replay an execution, analyse it, verify its validity or compare it with another execution. Describing such interactions is thus core to the documentation of process.

Therefore, the documentation of a process includes a set of *interaction p-assertions*, each describing an interaction between actors involved in the process.

Definition 6 (Interaction p-assertion) *An interaction p-assertion is an assertion of the contents of a message by an actor that has sent or received that message; the message must include information that allows it to be identified uniquely.* □

We do not prescribe the nature of the assertion of the message contents; instead, such decisions are left to the specific application. For instance, an interaction p-assertion could simply contain a copy of the message exchanged between two actors. Alternatively, if some data contained in the message is regarded as confidential by the actor or too large to be manipulated, the assertion may consist of the message in which the data concerned has been replaced by some other data or a pointer.

In a grid application based on command line executables, an interaction p-assertion can include the executable fully qualified name, its inputs and its outputs, whereas in a Web Services based approach, interactions documentation can include input and output SOAP messages, and a reference to the service, port and operation being invoked. In the latter case, we note that an interaction p-assertion potentially includes not only the SOAP message body, but also its envelope, containing valuable information such as security, addressing, resource or coordination contexts.

A crucial element of an interaction p-assertion is information to identify a message uniquely. Such information allows us to establish a flow of data between actors. Indeed, let us consider two interaction p-assertions: actor A making an assertion α_A that it sent actor B a message with identity i, and actor B making an assertion α_B that it received from A a message with the same identity i. Such a pair of interaction p-assertions α_A, α_B is said to be *matching*; it identifies a flow of data from actor A to B.

Actors may directly return outputs for the inputs they receive; alternatively, they may invoke other actors in order to obtain intermediate results that help them return their outputs. In both circumstances, the relationship between the outputs and the inputs of the actor is not explicit in the messages themselves, and can only be understood by an analysis of the actor's business logic, which is private to the actor.

We do not expect the source code of the actor to be made available, because it may not be feasible, or the code may not be at a suitable level of abstraction. Instead, in order to permit some understanding of the flow of data, an actor may decide to "volunteer" some information that is only available to it. An actor may provide *relationship p-assertions* that identify the relationship between its outputs (whether as returned result or invocation message to other actors) and its inputs (or intermediary results received from invoked actors).

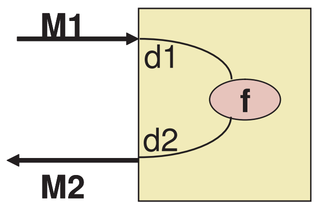
Definition 7 (Relationship p-assertion) *A relationship p-assertion is an assertion by an actor that the sending of a message would not be occurring or a data item it is sending would not be as it is (the* effect*), if it had not received other messages or data items had not been as they are (the* causes*), and that this relationship is due to its own action, expressible as the function applied to the causes to produce the effect.* □

While matching interaction p-assertions denote a flow of data between actors, relationships explain how data flows inside actors. Relationship p-assertions are directional since they explain how some data was computed from other data.

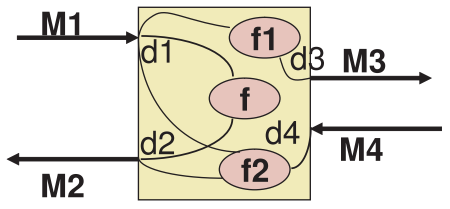
Figure 2 illustrates two actors. The first is a primitive actor, i.e., one that receives a message and produces a result, but does not invoke subsequent actors, or alternatively, an actor that does not make assertions of the invocations it makes of subsequent actors (say, for privacy reasons). In order to contribute some information about its internal flow of information, it can indicate that its output data (in the output message) is a function of the input data (contained in the input message). The second actor of Figure 2 is not primitive, and makes assertions of the contents of the messages it sends to and receives from another actor that it invokes. Like the first actor, it may indicate that its output is a function of its input; alternatively, it may explain how the data contained in the secondary invocation message and its result relate to the input and output.

Figure 2 displays the ideal case of purely functional actors, which do not maintain a persistent state across invocations. The same approach generalises to stateful actors: the data in an output message can be a function of the data received during a previous interaction and kept in a persistent store.

On the right-hand side of Figure 2, we see a symbolic representation of the p-assertions generated by the actors. Each p-assertion has a type and a content, and is asserted in the context of an interaction identified by a key.



interaction key	p-assertion type	p-assertion content
1	interaction	M1
2	interaction	M2
2	relationship	d2=f(d1)



interaction key	p-assertion type	p-assertion content
1	interaction	M1
2	interaction	M2
3	interaction	M3
4	interaction	M4
2	relationship	d2=f(d1)
3	relationship	d3=f1(d1)
2	relationship	d2=f2(d4,d1)

Figure 2: Data flow assertions by opaque and transparent actors

Hence, interaction p-assertions denote data flows *between* actors, whereas relationship p-assertions denote data flows *within* actors. Such data flows are core elements to reconstitute functional data dependencies in execution. In the most general case, such data flows constitute a directed acyclic graph (DAG). From a specific data item, the data flow DAG indicates where and how the data item is used; vice versa, following relationships in reverse helps us identify how a data item was produced. The data flow DAG is thus a core element of provenance representation, but it is not the only one; other p-assertions can provide further information about internal states of actors during execution, as we now explain.

Interaction and relationship p-assertions capture the flow of data in a process. In some circumstances, however, actors' internal states may also be necessary to understand the functionality, performance or accuracy of actors, and therefore the nature of the result they compute. Hence, we introduce the notion of an *actor state p-assertion* as the documentation provided by an actor about its internal state in the context of a specific interaction.

Definition 8 (Actor State p-assertion) *An actor state p-assertion is an assertion, by an actor, of data received from an (unspecified) internal component of the actor just before, during or just after a message is sent or received. It can, therefore, be viewed as documenting part of the state of the actor at an instant, and may be the cause, but not effect, of other events in a process.* □

Actor state p-assertions can be extremely varied: they may include the function the actor performs, the workflow that is being executed, the amount of disk and CPU a service used in a computation, the floating point precision of the results it produced, or application-specific state descriptions.

In summary, p-assertions can be of three disjoint kinds: interaction p-assertions, relationship p-assertions and actor state p-assertions. We note that p-assertions are independent of the actual service technology used to implement applications.

2.3 Provenance Lifecycle

In the previous section, we characterised the syntactic nature of p-assertions, in the form of a broad classification in three different categories, according to whether they document interactions, relationships or actor states. We now focus on a dynamic characterisation of p-assertions and, in particular, when they are created, recorded, queried and managed, with respect to process execution. These different phases identify a *provenance lifecycle*, which we now describe. (We note that such a lifecycle is to be understood in the context of application execution and should be distinguished from a methodology that identifies design steps in order to conceive an application that is provenance aware.)

Before discussing the provenance lifecyle, it is necessary to introduce an architectural element. Since we aim to provide a long-term facility for storing the provenance representation of data items, we delegate to a specific element, which we refer to as a *provenance store*, the role of making persistent, managing and providing controlled access to such provenance representation. The choice of an explicit architectural element to embody this role in no way implies any form of physical deployment; instead, it helps us identify the kind of functionality that is necessary in order to offer support for provenance.

The provenance lifecycle is composed of four different phases. As execution proceeds, actors create p-assertions that are aimed at representing their involvement in a computation. After their creation, p-assertions are stored in a provenance store, with the intent they can be used to reconstitute the provenance of some data. The provenance store therefore acts as storage of p-assertions. After a data item has been computed, users (or applications) may need to obtain the provenance of this data item: they can do so by querying the provenance store. At the most basic level, the result of the query is the set of p-assertions pertaining to the process that produced the data. More advanced query facilities may return a representation derived from p-assertions that is of interest to the user. Finally, as time progresses, the provenance store and its contents may need to be managed to handle distribution, change management, curation etc. In summary, the provenance lifecyle is composed of four different phases: *(i)* creating, *(ii)* recording, *(iii)* querying and *(iv)* managing. A provenance system should provide support for all these phases.

3 The Family of Specification Documents

In this section we describe each of the specification documents and supporting documents listed in Figure 1. The content is split into four groupings. First, we discuss two supporting documents: the overview and the glossary. The core specifications come next, which define the key aspects of the provenance model. After this we have a set of generic profiles that describe non-core aspects of the model. Finally, we provide an example of one specific technology binding – a tieing in of an aspect of the model to an implementation technology.

3.1 Support Documents

WS-Prov-Overview This document.

WS-Prov-Glo: The Provenance Glossary The WS-Prov-Glo document [TGJ+06] provides a glossary that defines a set of terms used in the draft provenance specification documents. The terms described are intended to be implementation and technology independent, with the intent that they can be analysed and applied to as many contexts as possible.

3.2 Core Specifications

WS-Prov-DM: The Process Documentation Data Model The WS-Prov-DM document [MGJ+06] presents a specification of the data model for process documentation. The approach is top down in nature, and starts by describing the p-structure — the logical organisation of process documentation, before drilling down into the models of the different forms of p-assertions. The identification of p-assertions and data items is then described, followed by a description of a model of context that allows information about related interactions to be passed between actors.

WS-Prov-Rec: The Provenance Recording Protocol Every provenance store supplies a Web Service interface for recording process documentation. It has a single operation, *record*, that takes Record document as input and returns an acknowledgement as result. The WS-Prov-Rec document [GTM+06] defines the schema for the record request and acknowledgement messages.

WS-Prov-Query: Provenance Queries In the WS-Prov-Query document [MMG+06b], a protocol is specified by which a querying actor and provenance store can communicate in performing a provenance query. This protocol primarily takes the form of an abstract WSDL interface defining messages to be accepted and produced by a provenance store. This document defines the schema for a provenance query request, the behaviour expected in processing that request and the resulting response.

WS-Prov-XQuery The process documentation data model defines schemas to be used for documentation about the execution of a process, *process documentation*, and introduces a *provenance store* — a type of Web Service with the capability for storing and giving access to process documentation. In particular, process documentation has a defined schema, the *p-structure*, which clients of a provenance store can navigate in queries to extract particular pieces of process documentation. In this document, a protocol is specified by which a querying actor and provenance store can communicate in performing a process documentation query. This primarily takes the form of an abstract WSDL interface defining messages to be accepted and produced by a provenance store.

3.3 Generic Profiles

WS-Prov-DM-Sec: Secure Provenance The data model for process documentation [MGJ+06] describes p-assertions as individual units for documenting process. These p-assertions can be signed by asserting actors in order to establish accountability for their creation. The WS-Prov-DM-Sec document [TMG+06a] extends on the data model for the basic p-assertions (relationship, actor-state and interaction) to include support for signatures.

WS-Prov-DM-Link: Distributed Provenance Process documentation can often be distributed across different provenance stores. To enable the discovery of related process documentation, a mechanism is required to link disparate but related process documentation to enable the effective collection of such documentation to answer provenance queries. The WS-Prov-DM-Link document [MTG+06b] represents a WS-Addressing profile on distributed process documentation that provides mechanisms to solve this problem.

WS-Prov-DM-DS: Transforming Process Documentation The activity of constructing an interaction p-assertion from a message can be considered as a single atomic transformation, which needs to be qualified by the actor creating that p-assertion in order for actors that retrieve that p-assertion from the provenance store to understand the exact nature of the transformation applied. This is equally true for an actor state p-assertion. Documentation styles are essentially descriptions of the types of transformations that can be applied to a message or to the internal state of an actor. The WS-Prov-DM-DS document [TMG+06b] presents a profile of several basic documentation style transformations that are likely to be useful in application domains that use process documentation. It is not intended to be exhaustive; other profiles may be provided of alternative documentation style transformations which may be generic or more specific in nature.

WS-Prov-PQuery-XPath The provenance query protocol [MMG+06a] has been defined, and includes the request for, algorithms to execute and result from a provenance query, as executed by a provenance query engine. Many parts of the request document are unspecified, being dependent on the provenance query engine implementation. This document defines an XPath-based profile by which provenance queries can be fully specified against process documentation that is in, or can be mapped to, XML format.

3.4 Technology Bindings

WS-Prov-SOAP: Provenance and SOAP Technology In order for p-assertions to be created, asserting actors need to identify which process they are making an assertion about, which requires some *shared context* between asserting actors. As it is application actors that make assertions, a further obligation is placed on them to pass context information between each other regarding the process being executed. This would often be achieved by putting the context information in the header of an application message (such as a SOAP message). The WS-Prov-SOAP document [MTG+06a] describes a specification of the p-header in the context of SOAP [Mit03] messages.

4 Conclusion

In this document we have presented an overview of the provenance model, the provenance lifecycle and the set of supporting specification documents that describe the model in detail. During the provenance lifecycle, the actors perform several roles: application actors execute processes; asserting actors create p-assertions about these processes; and recording actors record p-assertions in provenance stores, which allow querying actors to retrieve p-assertions. For these functions we have provided detailed models in the family of specification documents, which also specify models for transforming documentation, distributed provenance, security and a technology specific binding for SOAP messages. A glossary of all the terms found in this document and the other specification documents is also available.

References

[Bur00] S. Burbeck. The tao of e-business services. Technical report, IBM Software Group, 2000.

[GLM04a] P. Groth, M. Luck, and L. Moreau. Formalising a protocol for recording provenance in grids. In *Proc. of the UK OST e-Science second All Hands Meeting 2004 (AHM'04)*, Nottingham, UK, September 2004.

[GLM04b] Paul Groth, Michael Luck, and Luc Moreau. A protocol for recording provenance in service-oriented grids. In Teruo Higashino, editor, *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS'04)*, volume Lecture Notes in Computer Science, pages 124–139, Grenoble, France, December 2004. Springer-Verlag.

[GTM$^+$06] Paul Groth, Victor Tan, Steve Munroe, Sheng Jiang, Simon Miles, and Luc Moreau. Process Documentation Recording Protocol. Technical report, University of Southampton, June 2006.

[MGBM05] Simon Miles, Paul Groth, Miguel Branco, and Luc Moreau. The requirements of recording and using provenance in e-science experiments. Technical report, University of Southampton, 2005.

[MGJ$^+$06] Steve Munroe, Paul Groth, Sheng Jiang, Simon Miles, Victor Tan, and Luc Moreau. Data model for Process Documentation. Technical report, University of Southampton, June 2006.

[MH05] M.P.Singh. and M.N. Huhns. *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.

[Mit03] N. Mitra. Soap version 1.2 part 0: Primer. http://www.w3.org/TR/soap12-part0/, 2003.

[MMG$^+$06a] Simon Miles, Luc Moreau, Paul Groth, Victor Tan, Steve Munroe, and Sheng Jiang. Provenance Query Protocol. Technical report, University of Southampton, June 2006.

[MMG$^+$06b] Simon Miles, Steve Munroe, Paul Groth, Sheng Jiang, Victor Tan, John Ibbotson, and Luc Moreau. Process Documentation Query Protocol. Technical report, University of Southampton, June 2006.

[MTG$^+$06a] Steve Munroe, Victor Tan, Paul Groth, Sheng Jiang, Simon Miles, and Luc Moreau. A SOAP Binding For Process Documentation. Technical report, University of Southampton, June 2006.

[MTG$^+$06b] Steve Munroe, Victor Tan, Paul Groth, Sheng Jiang, Simon Miles, and Luc Moreau. WSRF Data Model Profile for Distributed Provenance. Technical report, University of Southampton, June 2006.

[SM03] M. Szomszor and L. Moreau. Recording and reasoning over data provenance in web and grid services. In *Int. Conf. on Ontologies, Databases and Applications of Semantics*, volume 2888 of *LNCS*, 2003.

[SRB06] David Snelling, Ian Robinson, and Tim Banks. Web Services Resource Framework v1.2 OASIS Standard, 1st April 2006. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf, 2006.

[TGJ$^+$06] Victor Tan, Paul Groth, Sheng Jiang, Simon Miles, Steve Munroe, and Luc Moreau. WS Provenance Glossary. Technical report, Electronics and Computer Science, University of Southampton, 2006.

[TMG$^+$06a] Victor Tan, Steve Munroe, Paul Groth, Sheng Jiang, Simon Miles, and Luc Moreau. A Profile for Non-Repudiable Process Documentation. Technical report, University of Southampton, June 2006.

[TMG$^+$06b] Victor Tan, Steve Munroe, Paul Groth, Sheng Jiang, Simon Miles, and Luc Moreau. Basic Transformation Profile for Documentation Style. Technical report, University of Southampton, June 2006.