The requirements of recording and using provenance in e-Science experiments

Simon Miles, Paul Groth, Miguel Branco and Luc Moreau
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
Tel: +44 23 8059 8309
Email: `sm@ecs.soton.ac.uk`

Abstract. In e-Science experiments, it is vital to record the experimental process for later use such as in interpreting results, verifying that the correct process took place or tracing where data came from. The process that led to some data is called the *provenance* of that data, and a *provenance architecture* is the software architecture for a system that will provide the necessary functionality to record, store and use *process documentation* to determine the provenance of data items. However, there has been little principled analysis of what is actually required of a provenance architecture, so it is impossible to determine the functionality they would ideally support. In this paper, we present use cases for a provenance architecture from current experiments in biology, chemistry, physics and computer science, and analyse the use cases to determine the technical requirements of a generic, application-independent architecture. We propose an architecture that meets these requirements and evaluate a preliminary implementation by attempting to realise two of the use cases.

1. Introduction

In business and e-Science, electronic services allow an increasing volume of analysis to take place. The large amount of processing brings its own problems, however. Questions that can be answered relatively easily about the *provenance* of (process that led to) a low number of experimental results, such as when the experiment took place or whether two experiments were performed on the same initial material, become near impossible to resolve with large numbers of results. We use the term *process documentation* to describe the records of experiments used to answer such questions. Rather than relying on scientists to remember experiment details or write paper notes, there is a need to automatically *record* process documentation into reliable and accessible *storage* so that it can later be *used*.

A *provenance architecture* is the software architecture for a system that provides necessary functionality to record, store and use process documentation in a wide variety of applications. In the PASOA (`www.pasoa.org`) project, we aim to develop a provenance architecture and, therefore, we must be aware of the range of uses to which the process documentation will be put. For this reason, we have surveyed a range of application areas and de-

© 2005 *Kluwer Academic Publishers. Printed in the Netherlands.*

termined the use cases that each has for process documentation. This paper focuses on e-Science applications and presents the results of our requirements capture and analysis process and discusses its implications for a provenance architecture.

In this paper, we present the use cases independently of their analysis, so that others can draw different implications from them. Our presentation is not intended to be a detailed use case specification; instead, the aim of our requirements capture is to draw out the *generic, re-usable aspects* of each application area so that a provenance architecture can be designed and built.

Our specific contributions in this paper are as follows.

– A range of use cases regarding the recording, querying and use of information regarding scientific, and particularly e-Science, experiments.

– An analysis of the technical requirements needed to be fulfilled to achieve these use cases.

– A proposed architectural design to address these technical requirements.

– A preliminary evaluation of the architecture through an implementation to achieve two of the use cases.

2. Background

2.1. Service-Oriented Architectures

Service oriented architectures (SOA) are the underpinning of the common distributed system technology in e-Business and e-Science. A service-oriented architecture (SOA) consists of loosely-coupled *services* communicating via a common transport. A service, in turn, is defined as a well-defined, self-contained, entity that performs tasks which provide coherent functionality. Typically, a service is only available through an interface, identifying all possible interactions with the service and represented in some standard format. A *client* is an entity that interacts with a service through its interface, requesting that the service perform an *operation* by sending a *message* containing all the required data. SOA technologies include Web Services [7], Grids [18], Common Object Request Broker Architecture (CORBA) [30] and Jini [36].

SOAs provide several benefits. First, they hide implementation behind an interface allowing implementation details to change without affecting the user of the service. Secondly, the loosely-coupled nature of services allows for their reuse in multiple applications. Because of these properties, SOAs are particularly good for building large scale distributed systems.

Typically, multiple services are used in conjunction to provide more extensive functionality than each provides individually. For re-usability, the way in which services are combined to perform a function can be encoded as *workflow* [1, 8]. In e-Science, workflows are used to define experimental processes in enactable form.

2.2. Provenance

The idea of *provenance* is fundamental to provenance architectures. Prior research has referred to this concept using several other terms including audit trail, lineage [25], dataset dependence [10], and execution trace [33]. We define the *provenance of a data item* as the process that produced that data and *process documentation* as the recorded documentation of such processes. In this section, we review a number of systems and domains that respectively provide and manage provenance-related functionality.

The Transparent Result Caching (TREC) prototype [35] uses the Solaris UNIX proc system to intercept various UNIX system calls in order to build a dependency map and, using this map, capture a trace of a program's execution. The sub-pushdown algorithm [27] is used to document the process of array operations in the Array Manipulation Language. A more comprehensive system is the audit facilities designed for the S language [12], used for statistical analysis, where the result of users command are automatically recorded in an audit file.

These systems work on a single local system with a single administrator, and so have limited application in capturing documentation of distributed e-Science processes.

Much of the research into recording process documentation has come in the context of domain specific applications. Some of the first research in provenance was in the area of geographic information systems (GIS)[25]. Lanter developed two systems for tracking the provenance of results in a GIS, a meta-database for tracking the process of workflows and a system for tracking Arc/Info GIS operations from a graphical user interface with a command line [24, 26]. Another GIS system that includes provenance tracking is Geo-Opera, an extension of GOOSE, which uses data attributes to point to the latest inputs/outputs of a data transformation, implemented as programs or scripts [9]. In chemistry, the CMCS project has developed a system for managing metadata in a multi-scale chemistry collaboration [29], based on the Scientific Application Middleware project [28]. Another domain where provenance tools are being developed is bioinformatics. The ^{my}Grid project has implemented a system for recording process documentation in the context of in-silico experiments represented as workflows aggregating Web Services [21]. In ^{my}Grid, provenance is gathered about workflow execution and stored in the user's personal repository along with any other metadata that might be of interest to the scientist [39]. The focus of ^{my}Grid is personalising the way the provenance of results is presented to the user.

By their nature, domain-specific provenance architectures must be re-developed for each new domain. Recording process documentation is a problem common to many, if not all, domains and a generic system would allow for greater re-use.

Provenance in database systems has focused on the data lineage problem [16]. This problem can be summarised as given a data item, determine the source data used to produce that item. In [37], the authors look at solving this problem through the use of the technique of weak inversion, and later use it to improve database visualization [38]. The data lineage problem has been formalised and algorithms for generating lineage data in relational databases are presented in [16]. AutoMed [17] tracks data lineage in a data warehouse by recording schema transformations. In [14], Buneman *et al.* redefine the data lineage problem as "why-provenance" and defines a new type of provenance for databases, namely, "where-provenance". "Why-provenance" is the collection of data sets (tuples) contributed to a data item, whereas, "where-provenance" is the location of a data element in the source data. Based on this terminology a formal model of provenance was developed applying to both relational and XML databases. In [13], the authors argue for a time-stamped based archiving mechanism for change tracking in contrast to diff-based mechanisms. These mechanisms may not capture the complete provenance of a database because there may be multiple changes between each archive of the database.

There have been several systems developed to provide middleware provenance support to applications. These systems aim to provide a general mechanism for recording process documentation and querying the provenance of results for use with multiple applications across domains and beyond the confines of a local machine.

According to [31], each user is required to have an individual e-notebook which can record data and transformations either through connections directly to instruments or via direct input from the user. Data stored in an e-notebook can be shared with other e-notebooks via a peer-to-peer mechanism.

Scientific Application Middleware (SAM) [28], built on the WebDav standard, provides facilities for storing and managing records, metadata and semantic relationships. Support for provenance is provided through adding metadata to files stored in a SAM repository.

The Chimera Virtual Data System contains a virtual data catalogue, which is defined by a virtual data schema and accessed via a query language [20]. The schema is divided into three parts: a transformation, a derivation and a data object. A transformation represents an executable, a derivation represents the execution of a particular executable, and a data object is the input or output of a derivation. The virtual data language provided by Chimera is used to both describe schema elements and query the data catalogue. Using the virtual data language, a user can query the catalogue to retrieve the transformations that led to a result. The benefit of using a common description language is that relationships between entities can be extracted without understanding the underlying data.

In [32], the authors argue for infrastructure support for recording process documentation in Grids and present a trial implementation of a system that offers several mechanisms for handling process documentation after it has been recorded. Their system is based around a workflow enactment engine submitting data to a *provenance service*. The data submitted is information about the invocation of various web services specified by the executing workflow script.

None of the existing technologies provide a *principled, application-independent* way of recording, storing and using process documentation. We attempt to achieve this with our provenance architecture.

3. Applications

In this section, we briefly introduce the *experiments*, i.e. scientific projects to check hypotheses or investigate material properties, to which our use cases apply. They have been classified by their scientific domain.

3.1. Biology

Intron Complexity Experiment
The bioinformatics domain already involves the analysis of a massive amount of complex data, and, as experiments become faster and automated to a larger degree, the experimental records are becoming unmanageable. The Intron Complexity Experiment (ICE) is a bioinformatics experiment to identify the relative *Kolmogrov complexity* of *introns* and *exons*, and the relation between the complexities of the two. Exons are subsequences of chromosomes that encode for proteins, introns are the sub-sequences that separate exons on a chromosome. This experiment uses a number of services, some externally provided, some written by the biologist, that analyse data drawn from publicly accessible databases such as GenBank [3]. When a potentially interesting result is found, the biologist re-runs parts of the workflow with different configuration parameters to try and determine why that result was produced.

Candidate Gene Experiment
The ^{my}Grid [5] project attempts to provide a working environment for bioinformaticians, particularly providing portals and middleware that can be used by many parties. Experimental processes are automated or partially automated by encoding them as workflows and executing them within a workflow enactment engine. ^{my}Grid has been concentrating on a few bioinformatics experiments that fit into a class called Candidate Gene Experiments (CGE). These experiments aim to discover as much information as possible about a gene (the *candidate gene*) from existing data sources, to determine whether it is involved in causing a genetic disorder.

Protein Identification Experiment
Proteomics is the study of proteomes, which are defined as all the proteins produced by a single organism. The Protein Identification Experiment (PIE) is performed to identify proteins from a given sample, e.g. to determine what proteins are present only in someone with a certain disease. To this end, the characteristics of protein fragments can provide evidence for the identification of the protein. This requires first breaking the protein at well-identified points, i.e. at given amino acids, resulting in a set of peptides. The peptides are examined using a mass spectrometer to determine their mass-to-charge ratio. To obtain more accurate results, the peptides are then further fragmented, at random points, by bombarding the peptides with a charged gas, and these fragments are again fed to the spectrometer. Databases of previously analysed results are used to match peptide characteristics to possible proteins, as well as to provide further information on the proteins such as the functional group to which they belong.

3.2. PHYSICS

Particle Detection Experiment
In High Energy Physics (HEP) experiments, vast amounts of data are collected from detectors and stored ready to be analysed in different ways by groups of specialised physicists, *Physics Working Groups* (PWG), in order to identify traces of particles produced by the collision of particles at high energies. Experimental processes in a Particle Detection Experiment (PDE) are complex, with the data provider, CERN, providing some processing of the raw data, followed by further analysis localised around the world. The group of PWGs that manage the data as a whole, along with everyone that provides the resources to do so, is called the *Collaboration* for this experiment.

3.3. Chemistry

Second Harmonic Generation Experiment
The Second Harmonic Generation Experiment (SHGE) analyses properties of liquids by bouncing lasers off them and measuring the changes that have occurred in the polarisation of the laser beam [15].

3.4. Computer Science

Service Reliability Experiment
The e-Demand [2] project attempts to make service-oriented Grids more reliable and better tailored to those using them by examining the relative reliability and quality of services. In the Service Reliability Experiment (SRE), several services implement the same function using different algorithms. The results returned by the services are compared in order to increase the assurance that the results are valid.

Security Testing Experiment
The Semantic Firewall project aims to deal with the security implications of supporting complex, dynamics relationships between service providers and clients that operate from within different domains, where different security policies may hold and different security capabilities exist [11]. In the Security Testing Experiment (STE), a client wishes to delegate their access to data to another service, and so a complex interaction between the services is necessary to ensure security requirements are met. A *semantic firewall* will reason about the multiple security policies and allow different operations to take place on the basis of that reasoning. The reasoning can be dependent on the entities interacting and other contextual information provided to and from the existing security infrastructures. The semantic firewall can be seen as guiding the interacting parties through a series of interaction protocol states on the basis of reasoning, ensuring that interactions follow the security policies of individual domains.

4. Use Case Analysis

The above experiments provided us with a selection of use cases involving the capture and use of process documentation. In this section, we present each of the issues raised by the use cases, introducing each use case where it is most illustrative. The issues identified are expressed as general *technical requirements* so that design decisions can be made regarding a suitable provenance architecture. In each case, we have given the technical requirement in the form of a statement “PASOA should provide for...” with reference to a particular behaviour of the system, where PASOA refers to the provenance architecture we wish to design. Each statement makes no implications about how the architecture achieves the requirement, so that others can use them to develop alternatives to PASOA.

4.1. Methodology

Given the project aims, we followed the methodology below for gathering use cases from each user.

– We provided a broad description of our goals, making it clear that we intended to design an architecture to aid recording what occurred during experiments. Since we aim to uncover tasks that the user cannot currently perform, we presented some of the use cases gathered from previous users to each subsequent user as inspiration.

– We catalogued the provenance-related use cases that the user has already considered and thoughts regarding possible other benefits that may be obtained from having process documentation available, i.e. functional requirements. Also, we asked the user about the non-functional requirements of any software we may provide.

– We extracted the concrete functional and non-functional use cases from the interviews, identifying the actors involved and the actions they perform, and wrote them in a consistent form.

– We presented the written use cases to the user for confirmation that they were correct, and for them to correct where not.

4.2. Functional Requirements

In this section, we present those use cases providing functional requirements on the provenance architecture. Each use case in this section is defined in terms of the relevant actors and the actions they perform. The final sentence of each use case is a *provenance question*: an action that can be realised by processing recorded process documentation. The provenance questions place explicit demands on the provenance architecture and so imply general technical requirements. For ease of identification, the provenance question in each use case is *italicised*. All experiments produce some data, so the record of an experiment is the provenance of one or more data items. Where a question is asked of the information recorded by the provenance architecture, we mean that it is asked of the provenance of one or more data items produced by the experiment.

4.2.1. *Types of Provenance*

The types of process documentation that users considered to be relevant to the provenance of a result varied, and it is helpful to distinguish and describe these types with reference to a few particular use cases.

USE CASE 1. (ICE) A bioinformatician, B, downloads sequence data of a human chromosome from GenBank and performs an experiment. B later performs the same experiment on data of the same chromosome, again downloaded from GenBank. B compares the two experiment results and notices a difference. B *determines whether the difference was caused by the experimental process or configuration having been changed, or by the chromosome data being different (or both).* □

First, this use case requires a record of the *execution* of the experiment, i.e. the interaction between services that took place including the data that was passed between them.

The same use case provides an example of another type of process documentation, i.e. extra information from either service participating in the experiment at the time that the experiment was run. Each service typically relies on an algorithm, which may be modified over time, and it is likely that only the service running the algorithm will have access to it. If B can determine whether the algorithm has changed between experiment runs, B can also determine whether the results are due to that change.

USE CASE 2. (CGE) A bioinformatician, B, enacts an experimental workflow using a workflow enactment engine, W. W processes source data to produce intermediate data, and then processes the intermediate data to produce result data. B retrieves the result data. B *then examines the source and intermediate data used to produce the result data.* □

Use Case 2 demonstrates the desire for a third type of process documentation: the *relationship* between data items in a process, e.g. relating a result to the intermediate data in the process that produced it. We can summarise the types of process documentation as follows.

– *Interaction:* A record of the interaction between services that took place, including the data that was passed between them.

– *Actor State:* Extra information about a service participating in the experiment at the time that the experiment was run.

– *Relationship:* Information on how one data item in a process relates to another.

TECHNICAL REQUIREMENT 1. *PASOA should provide for the recording and querying of interactions, actor states and relationships.*

4.2.2. *Structure and Identity of Data*

Services exchange data in the form of *messages*. Messages specify the operation that the client wishes to perform as well as a set of structured data to be analysed and/or to be used to configure the analysis.

USE CASE 3. (ICE) A bioinformatician, B, performs an experiment on a set of chromosome data, from which the exon and intron sequences have been extracted. As a result of that experiment, B identifies a highly compressable intron sequence. B *identifies which chromosome the intron originally came from.* □

In Use Case 3, data elements within the messages exhanged between services need to be consistently identified. We cannot guarantee that the content of the data itself provides unique identification, so an identitifier may have to be associated with the data. To satisfy the questions regarding a data element, its identifier should be usable in queries about the process documentation. Finally, to associate an identifier with an element of a message recorded in the process documentation, there must be a way to *reference* that element.

USE CASE 4. (PDE) A physicist, P, extracts a subset of data from a large data set, owned by the Collaboration, and performs experiments on that subset over time. The Collaboration later updates the data set with new data. P *determines whether the experiments should be re-run based on the new data set.* □

TECHNICAL REQUIREMENT 2. *PASOA should provide for association of identifiers with data, so that it can be referred to in queries and by data sources linking experiments together.*

TECHNICAL REQUIREMENT 3. *PASOA should provide for referencing of individual data elements contained in message bodies recorded in the process documentation.*

4.2.3. *Metadata and Context*

The questions that users wish to ask often draw together process documentation regarding particular experiments with other information. For example, in the Candidate Gene Experiment, information such as the semantic type of each data item in an ontology, such as the Gene Ontology [4], may be used by the bioinformatician to provide further reason to believe the candidate gene is involved in the genetic disease. Similarly, the lab and project on which the producer of a given data item worked may be used to help determine its likelihood of being accurate.

USE CASE 5. (SHGE) In order to conform to health and safety requirements, a chemist, C, plans an experiment prior to performing it. The plan is at a high-level, e.g. including the steps of mixing and analysing materials but excluding implied steps like measuring out materials. C performs the experiment. *Later, another chemist,* R, *determines whether the experiment carried out conformed to the plan.* □

In Use Case 5, the pre-defined plan of the experiment does not necessarily exactly match the actual steps performed. As shown in Figure 1, a single planned activity may map to one or more actual activities. As described in the use case, the plan is produced before any process documentation is recorded, but is used in comparison with the process documentation. It is an example of *process metadata*: data independent from but used in conjunction with process documentation. Given that process metadata is of an arbitrary wide scope, any framework for supporting the use of provenance must take into account stores of metadata that will be queried along with the process documentation.

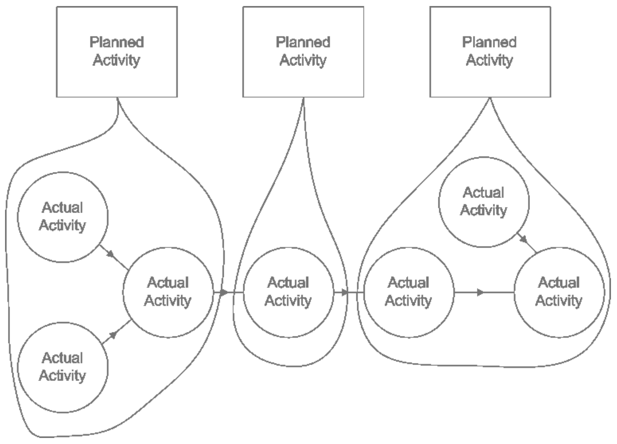

Figure 1. Plans in CombeChem: planned activities do not map exactly to performed activities

The *context* of an experiment is anything that was true when the experiment was performed. Some contextual information is relevant to the provenance questions. In Use Case 6, the experiment *configuration*, the spectrometer voltage, is relevant to the question asked later.

USE CASE 6. (PIE) A biologist, B, sets the voltage of a mass spectrometer before performing an experiment to determine the mass-to-charge ratio of peptides. Later another biologist, R, judges the experiment results and considers them to be particularly accurate. R *determines the voltage used in the experiment so that it can be set the same for measuring peptides of the same protein in future experiments.* □

A particular type of metadata is *semantic information* about the entities involved in an experiment. For instance, the following use case requires semantic metadata about the data exchanged between services in the experiments.

USE CASE 7. (ICE) A bioinformatician, B, performs an experiment on a FASTA sequence encoding a nucleotide sequence. *A reviewer,* R, *later determines whether or not the sequence was in fact processed by a service that actually only meaningfully processes protein sequences.* □

Use Case 7 requires not only that an ontology of biological data types is provided, but also that process documentation can be annotated with semantic types. This does not require, however, that the semantic annotation be stored in the same place as the data.

TECHNICAL REQUIREMENT 4. *PASOA should provide for process documentation and associated metadata in different stores to being integrated in providing the answer to a query.*

4.2.4. *Sessions*

We have found that many use cases compare the run of one experiment to that of another, requiring that records regarding those experiments include a delimitation of one experiment from another. In service-oriented architecture terms, this means that we need to delimit one set of service interactions from another. We define a *session* as a group of service interactions (experiment activities).

USE CASE 8. (SRE) A computer scientist, C, calls service X which calculates the mean average of two numbers as (a/2)+(b/2). C then calls service Y with the same two numbers, where Y calculates the average as (a+b)/2. C does not know if X or Y are reliable, so by getting results from both, C can compare them and, if they are the same, be more sure having the correct result (because the same value is produced by two different services). However, X and Y may use a common third service, Z, behind the scenes, e.g. to perform division operations. If Z is faulty then the results from X and Y may be consistent but wrong. *For extra assurance,* C *determines whether* X *and* Y *did in fact use a common third service.* □

In Use Case 8, two sessions must be distinguished in order to answer the provenance question. The first session is the execution of X and all its dependencies, the second is the execution of Y and all its dependencies. The scenario is depicted in Figure 2. The provenance question can then be expressed as: was the same service used in both sessions? Similarly, Bioinformatics Use Case 1 requires that we compare two experiments, recorded as two sessions, and show the differences.

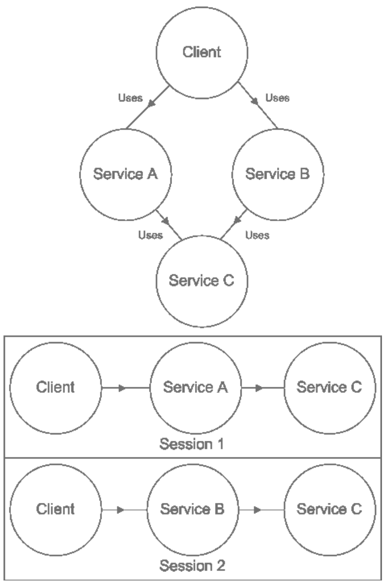

Figure 2. Sessions using the same common service in e-Demand: the client is unaware that two services, A and B performing the same function using different algorithms, rely on a common service C

TECHNICAL REQUIREMENT 5. *PASOA should provide a mechanism by which to group recorded process documentation into a session, and should allow comparison between sessions.*

4.2.5. *Query*

The actor asking a provenance question does not always know in advance which specific experiments or data their question addresses. For example, in Use Case 9, we do not know which experiments we are looking for in advance, only which source material was used as input to them, and perhaps contextual information such as the experimenter.

USE CASE 9. (SHGE) A chemist, C, performs an experiment but then examines the results and finds them doubtful. C determines the source material used in the experiment and then which other recent experiments used material from the same batch. C *examines the results of those experiments to determine whether the batch may have been contaminated and so should be discarded.* □

Given that we expect a large volume of process documentation to be recorded over the course of many experiments, a search mechanism is required to answer the provenance question of Use Case 9. Data from one experiment may be used to improve the quality of future results by filtering intermediary data, as follows.

USE CASE 10. (PIE) A biologist, B, performs many experiments over time to discover the characteristics of peptide fragments. The fragments are used as *evidence* that a peptide is in the analysed material. Usually the discovery of several fragments is required to confidently identify a peptide, but some fragments are unique enough to be adequate alone. B *determines that a fragment with particular characteristics is produced most times a particular peptide was analysed and rarely or never when that peptide was not present*. □

To understand the range of queries required, we can present those required to help achieve some of the use cases described above. To achieve Use Case 1, the user asks for the full contents of the records of two experiments, so that a comparison can then be made. To achieve Use Case 2, the user asks for the interaction that has a given data item as its output. To achieve Use Case 8, the user asks for all services used in two given experiments. To achieve Use Case 5, the user asks for all experiments using a given data item as input. To achieve Use Case 10, the user asks for all peptides output as intermediary data in previous protein identification experiments.

TECHNICAL REQUIREMENT 6. *PASOA should provide for the process documentation to be returned in the groups specified at the time of recording or searched through on the basis of contextual criteria.*

4.2.6. *Processing and Visualisation*

In most use cases, the full process documentation of an experiment is not presented to the user in order to answer the provenance question. It must first be analysed and then presented in a form that makes the answer to the provenance question clear.

USE CASE 11. (SHGE) A chemist, C, performs an experiment to determine the characteristics of a liquid by bouncing laser light off of it and examining the changes to the polarisation of the light. As this method is fairly new, it is not established how to then process the results. C analyses the results through a plan, i.e. a succession of processes, that seem appropriate at the time and ends with potentially interesting results. *At a later date,* C *determines the high-level plan that they followed and re-performs the experiment with different liquid and configuration.* □

USE CASE 12. (STE) A service, X, is accessed by by an intruder, I, that should not have rights to do so. *Later, an administrator becomes aware of the intrusion and determines the time and the credentials used by the intruder to gain access.* □

In Use Case 11, the process documentation provides the full information of what has occurred, but to answer the question, C requires a high-level plan. The process documentation therefore needs to be *processed* to answer the question. Again in Use Case 12, the process documentation must be processed in order to provide an answer to the provenance question. All answers to provenance questions have to be made presentable to the user. For example, in Use Case 13, the process documentation is presented in a report.

USE CASE 13. (ICE) A bioinformatician, B, performs an experiment. B *publishes the results and makes a record of the experiment details available for the interest of B's peers.* □

TECHNICAL REQUIREMENT 7. *PASOA should provide a framework for introducing processing of process documentation of all three types discussed in Section 4.2.1 (interactions, actor states and relationships), using various methods, then visualising the results of that processing.*

4.2.7. *Non-repudiation*

In some cases, such as where the experimental results justify the efficacy of a new drug for example, the provenance does not just need to verify that the experiment was performed as stated but *prove* it. To aid this, all parties in an experiment could record the process documentation from their own perspective, and these perspectives can then be compared. Along with other measures to prevent collusion or tampering with the process documentation, the joint process documentation provides evidence of the experiment that cannot be denied, or *repudiated*.

One use case that requires multiple parties to record process documentation independently is where the intellectual property rights of the experimenter may conflict with those of the services they use in experiments, as now described.

USE CASE 14. (ICE) A bioinformatician, B, performs an experiment from which they develop a new drug. B attempts to patent the drug. *The patent reviewer,* R, *checks that the experiment did not use a database that is free only for non-commercial use, such as the Ecoli database.* □

As well as being able to prove particular services were used in an experiment, we may also need to be able to prove the time at which it was done, so that researchers can (or cannot) claim they performed an experiment earlier than a published one.

USE CASE 15. (SHGE) A chemist, C, performs an experiment finishing at a particular time. D later performs the same experiment and submits a patent for the result and the process that led to it to patent officer R. C claims to R that they performed the experiment before D. R *determines whether* C *is correct.* □

TECHNICAL REQUIREMENT 8. *PASOA should provide a mechanism for recording adequate process documentation, in an unmodifiable way, to make results non-repudiable.*

4.2.8. *Re-using Experimental Process*

Process documentation can be used in deciding what should happen in the future. An experiment is performed to achieve some goal, such as verifying a hypothesis. The process documentation can be used to identify the process and to repeat it.

USE CASE 16. (CGE) A bioinformatician, B, performs an experiment using as input data a specific human chromosome from the most recent version of a database. Later, another bioinformatician, D, updates the chromosome data. B *re-enacts the same experiment with the most recent version of the chromosome data.* □

USE CASE 17. (PIE) A biologist performs an experiment to identify peptides in a sample. Identifications are made by comparing characteristics of the peptides and their fragments with already known matches in a database. In the experiment, some peptides are identified, others cannot be. Later, after other experiments have been conducted, the database contains more information. *The system automatically re-enacts the analysis of those peptides that were not identified.* □

In Use Case 16, the scientists can use process documentation to re-enact the experiment. The re-enactment can even be automatic, since changes in the databases can be matched to experiments that use those databases. In order to re-enact the experiment the following information is needed: the service called in at each stage of an experiment and the inputs given to each service. The process documentation regarding previous experiments may be used in a less automated fashion to determine how future experiments are to be run.

In fact, there are several different ways in which experimental process can be re-used. *Re-enactment* is performing the same experiment, but using contemporary data and services, while *repetition* means performing the same experiment with the same data and services as before, e.g. to test that the results can be reproduced. Also, rather than performing the whole experiment again, a scientist may wish to perform it only up until the stage that intermediate results differ, to detect at what point the difference lies.

TECHNICAL REQUIREMENT 9. *PASOA should provide for the use of process documentation to* re-enact *an experiment using the same process but new inputs, and to* reproduce *an experiment with the same process and inputs.*

4.2.9. *Aggregated Service Information*

The process documentation provides information on services used in experiments as well as experiments themselves. Combining the information of several traces allows the scientist to aggregate data about individual services used in multiple experiments, as illustrated in the next use case.

USE CASE 18. (CGE) Several bioinformaticians perform experiments using service X. Another bioinformatician, B, constructs a workflow that uses X. B *can estimate the duration that the experiment might take on the basis of the average time* X *has taken to complete its tasks before.* □

TECHNICAL REQUIREMENT 10. *PASOA should provide for querying, over process documentation of multiple experiments, about the aggregate behaviour and properties of services.*

4.3. NON-FUNCTIONAL REQUIREMENTS

Other use cases provide us with non-functional requirements, regarding *how* the architecture should operate. Since the use cases presented highlight demands on the way in which process documentation should be recorded, stored and used, there is not a provenance question in every case, i.e. there is not always a new function realised by the provenance architecture.

4.3.1. *Storage*

All provenance use cases require some reliable storage mechanism for the process documentation; however, some require long-term storage of provenance to satisfy their needs, while others require the data to be preserved and accessible only in the short-term. An example of the former type of use case is the following.

USE CASE 19. (SHGE) A chemist, C, performs an experiment. C then publishes their results on-line. Another chemist, R, discovers the published results years later. R *determines whether the results are valid by checking the experimental process that was performed.* □

In order for process documentation to be accessible as a part of a publication, it should persist as long as the publication, preferably forever. On the other hand, for many use cases the process documentation may only retain its relevance for a matter of hours, months or years.

TECHNICAL REQUIREMENT 11. *PASOA should provide for the management of the period of storage of process documentation to be managed, including preservation of data for indefinite periods or deletion after given periods.*

4.3.2. *Distribution*

Given that e-Science experiments can involve many services owned by many parties, it is impractical to expect a single data store to be used to retain all of the process documentation. An example of this is given in Use Case 20.

USE CASE 20. (PDE) A physicist, P, performs a set of experiments. A selective subset of the results, including the process documentation of the experiments that produced them, are made available to the physicist's Physics Working Group, G. The administrators of G then make a subset of those results, including their provenance, available to the Collaboration. The Collaboration stores the results and process documentation with security, fidelity and accessibility for a longer period of time that P or G are able to. □

As services are distributed, process documentation may be stored in a distributed manner and must be linked up in order to answer queries. It is clear that provenance storage should be distributed but that queries should draw process documentation from all relevant stores.

TECHNICAL REQUIREMENT 12. *PASOA should provide for distribution in the storage of process documentation and allow queries to draw data from multiple stores.*

4.3.3. *Very Large Data Sets*

Where data is relatively small it can be stored easily for long periods. However, in some cases, it can be very large, such as in the Use Case 21.

USE CASE 21. (PDE) A physicist, P, performs an experiment using detector data as input. The size of the detector data is in the order of petabytes. The process documentation of the experiment is recorded for later use without copying the data set. □

It is impractical to store or process data multiple times for very large data sets, and provenance architectures must address this.

TECHNICAL REQUIREMENT 13. *PASOA should provide for recording process documentation and querying the provenance of very large data sets.*

4.3.4. *Integration with Existing Software*

In some domains, de-facto standards exist for recording some of the process information electronically, and in some cases there is also software support. For example, the provenance question in Use Case 22 can be answered using data from legacy software.

USE CASE 22. (PDE) An existing service, X, regularly records the versions of libraries installed on computer node N. X records the version of library L at time T. A physicist, P, performs an experiment using data produced by N. P examines the experiment results and judges that they may be incorrect. P *queries the process documentation to discover the library versions used by* N *when producing the data.* □

Developers of a new provenance architecture have to be aware of existing standards for recording and accessing process documentation and ensure that their software interoperates with that which already exists. Also, forthcoming standards that have the support of the community should be acknowledged, and provenance architectures should be able to interoperate with them.

USE CASE 23. (PIE) A biologist, B, performs an experiment. B then queries the process documentation regarding that experiment by using software that follows the widely supported Proteomics Standards Initiative [6]. □

TECHNICAL REQUIREMENT 14. *PASOA should provide for the integration of the architecture with existing standards and software.*

4.4. SUMMARY

The types of use use case listed above can be summarised as the following general tasks.

– Checking whether results were due to interesting features of the material being experimented on or nuances of the experiment performed.

– Determining the probable effectiveness of similar future experiments.

– Accessing a historical record, or aide memoire, of work conducted.

– Proving that the experiment claimed to have been done was actually done.

– Proving that the experiment done conformed to a required standard.

– Checking that the experiment was performed correctly, and the services involved used correctly.

- Tracing where data came from and the processes it had been through to reach its current form.
- Tracing which source data was used to produce given result data and vice-versa.
- Linking together data and experiments by their process documentation, to provide extra context to understanding those experiments.
- Deriving the higher-level processes that have been gone through to perform an experiment, so that they can be checked and re-used.
- Providing the process information required for publishing an experiment's results.
- Verifying that services used are working as they should be.
- Allowing experiments to be re-enacted to check that services and/or data has not changed in a way which affects the results.

5. Proposed Architecture

In the PASOA project, we aim to provide a framework architecture capable of tackling the presented use cases. Our analysis has led to a number of architecural design decisions, which we outline in this section. We then describe our provenance architecture.

5.1. Design Decisions

The technical requirements of Section 4 have informed a number of design decisions regarding the PASOA architecture. We describe the most significant ones below.

5.1.1. *Separation of concerns*

The breadth of use cases shows the potentially unlimited scope of functionality that a provenance architecture could provide. We need to separate concerns so as to provide a framework which can be built upon to satisfy not only use cases above, but also new ones as they appear. It should be noted that very few of the concerns expressed in the technical requirements apply universally and uniformly to all applications; there is just a general need for *recording*, *querying* and *processing* process documentation. As querying requires that data be recorded in a queryable form and processing requires that data can be queried using a pre-defined mechanism, recording can be seen as the most fundamental part of a provenance architecture. We also note that recording needs to be consistent across applications to meet our goal for re-usable open system querying and processing of the process documentation.

Hence, we define a *layered architecture* with three layers, each building on the previous one: (i) Fundamentals of recording and access, (ii) Querying, and (iii) Processing. Application specificity should be pushed up these three layers where possible, in order to separate out general from application-specific concerns.

5.1.2. *Documentation as assertions*

In Section 4.2.7, we noted that multiple actors can submit different documentation on the same process. The provenance architecture relies on actors to record accurate process documentation about what has occurred in order for provenance questions to be answered. Process documentation may be inaccurate when an actor is faulty or is maliciously recording incorrect information. The provenance architecture should not, therefore, require those asking provenance questions to believe all the process documentation provided to them: they can judge the likely accuracy based on their opinion of the actor submitting each piece of process documentation. We therefore consider process documentation to be made up of a set of *assertions* about a process that has occurred, made by the actors that took part in that process. A *p-assertion* is an assertion that is made by an actor and pertains to a process, and process documentation is a set of p-assertions.

5.1.3. *Documentation structure based on interactions*

As described in Section 4.2.1, we have determined there to be at least three types of process documentation: interactions, actor states and relationships. Our architecture, therefore, has to support the recording and use of p-assertions regarding all these types of data. We argue that any p-assertion can be viewed as being with regard to an interaction, as follows. Interaction p-assertions state what information is exchanged between actors in an interaction. Actor state p-assertions are metadata to documentation of interactions, as they describe the state of actors at the time when interactions took place. Relationships between data can be documented as relationships between the interactions in which the data is exchanged. Therefore, our architecture should be based on the recording of the interactions between actors and allow metadata regarding each interaction to be additionally recorded in association.

5.1.4. *Interaction-specific or non-provenance metadata*

Given the basis of interactions, we can further separate concerns. Metadata specific to an interaction, including the state of an actor or the data exchanged, must clearly be associated directly with the interaction and so should be recognised in our recording process documentation procedures. Other metadata can be stored elsewhere and references made to the process documentation to make the association explicit. The metadata will then be used together when performing queries or processing.

5.1.5. *Reference of elements in the store*

In order to associate metadata with actors and data in interactions, there must be a way to refer to those entities. First, we can provide a way to reference recorded interactions and the messages passed in those interactions. Then, while the structure of data used in experiments will vary widely, we can provide some uniformity in referring to elements of the data at the query level.

5.1.6. *Tracers to delimit sessions*

In Section 4.2.4, we identified the need to delimit independent processes, or sessions, within an application's execution. One means by which we can do this is to introduce identifiers, called *tracers*, into all messages sent within one session. Any actor receiving a message containing a tracer is expected to perpetuate it into all subsequent messages the actor sends in the same session: in this way a tracer acts like dynamic context in transactional systems. By querying a provenance architecture for all interaction p-assertions containing a tracer, we can then retrieve all (documented) interactions in a session.

5.1.7. *Extensible architecture for querying*

As the data comes in many forms and structures, because we should attempt to fit in with existing standards and software in some cases, and because the questions asked about past experiments vary considerably between applications, we cannot and should not provide a single query interface for them all. However, we can take a layered approach, whereby we provide a few general search mechanisms over the process documentation with the aim that it will ease the development of application-specific query engines. There should be no compulsion for these query mechanisms to be used if it is easier to search for results without them.

5.2. Proposed architecture

We have developed a protocol for recording process documentation according to the design decisions of Section 5.1, which is detailed in [22] and not expanded on further here. We can now design an architecture to address the use cases as a whole. Our proposed architecture is shown in Figure 3, which embodies the design decisions of Section 5.1, and each entity depicted is explained below.

Central to the architecture is the notion of a *provenance store*, which is a service designed to store and maintain process documentation beyond the

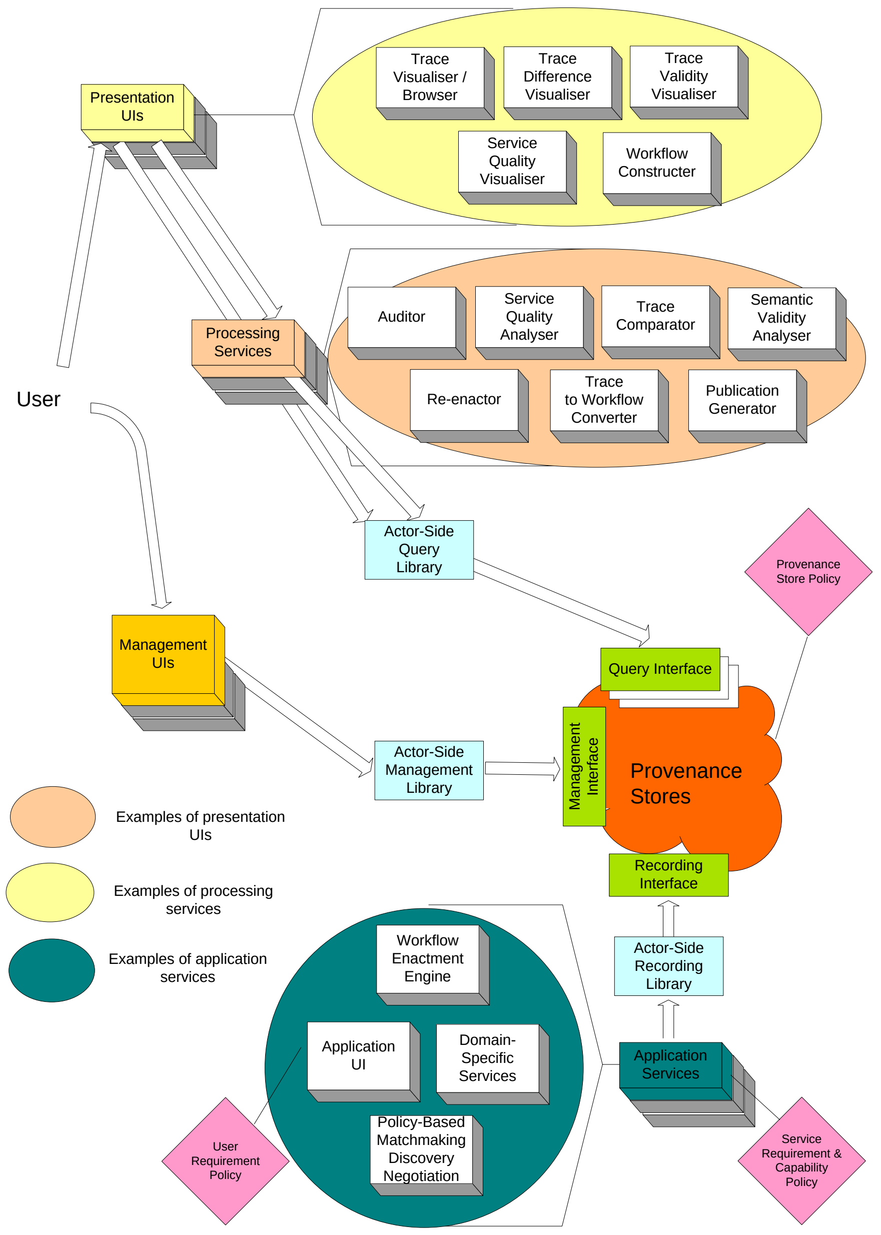
Presentation UIs
Trace Visualiser / Browser
Trace Difference Visualiser
Trace Validity Visualiser
Service Quality Visualiser
Workflow Constructer
Processing Services
Auditor
Service Quality Analyser
Trace Comparator
Semantic Validity Analyser
Re-enactor
Trace to Workflow Converter
Publication Generator
User
Actor-Side Query Library
Provenance Store Policy
Management UIs
Query Interface
Actor-Side Management Library
Management Interface
Provenance Stores
Examples of presentation UIs
Examples of processing services
Examples of application services
Recording Interface
Actor-Side Recording Library
Workflow Enactment Engine
Application UI
Domain-Specific Services
Policy-Based Matchmaking Discovery Negotiation
User Requirement Policy
Application Services
Service Requirement & Capability Policy

Figure 3. Proposed PASOA Provenance Architecture

lifetime of a Grid application. Such a service may encapsulate at its core the functionality of a physical database, but also provides additional functionality pertinent to the requirements of the provenance architecture. In particular, the provenance store's responsibility is to offer long-term persistence of p-assertions in a consistent structure, i.e. according to a pre-defined schema.

The structure of process documentation in a provenance store is based on interactions between actors in an experiment, the states of actors during interactions and the relationships between data sent in interactions (satisfying TR 1). The structure allows recorded process documentation to be referred to (satisfying TR 3) and identifiers associated with the referenced process documentation (satisfying TR 2). The structure makes explicit which actor made each p-assertion (satisfying TR 8) and has explicit space for recording the tracers exchanged in interactions, thereby allowing sessions to be delimited (satisfying TR 5)

In a given application, one or more provenance stores may be used in order to act as storage for p-assertions (satisfying TR 12): multiple provenance stores may be required for scalability reasons or for dealing with the physical deployment of a given application, possibly involving firewalls.

In order to accumulate p-assertions, a provenance store provides a *recording interface* that allows recording actors to submit p-assertions related to their interactions and internal states, for recording purposes. A provenance store is not just a sink for p-assertions: it must also support some query facility that allows, in its simplest form, browsing of its contents and, in its more complex form, search, analysis and reasoning over process documentation so as to support use cases. To this end, we introduce *query interfaces* that offer multiple levels of query capability (satisfying TR 6). Because the process documentation can be referred to and the query languages are flexible, aggregated information regarding services can be derived (satisfying TR 10). Finally, since provenance stores need to be configured and managed, an appropriate *management interface* is introduced.

Some *actor-side libraries* facilitate the tasks of recording p-assertions in a secure, scalable and coherent manner and of querying and managing provenance stores. They are also designed to ease integration with legacy applications. We also expect actor-side libraries to provide some support to create common forms of p-assertions. The creation of p-assertions here will need to take into account the expression of causal relationships in an appropriate manner, as previously discussed.

During an application's execution, all *application services* are expected to submit p-assertions to a provenance store; this not only applies to *domain-specific services*, but also to generic middleware, such as *workflow enactment engines*, *registries* and *application user interfaces*.

Once p-assertions have been recorded in a provenance store, process documentation can be used by *processing services* and *presentation user inter-*

faces. The former provide added-value to the query interfaces by further searching, analysing and reasoning over recorded p-assertions, whereas the latter essentially visualise query results and processing services' outputs (satisfying TR 7). Figure 3 provides examples of such processing services and presentation UIs. For instance, processing services can offer auditing facilities, can analyse quality of service based on previous execution, can compare the processes used to produce several data items, can verify that a given execution was semantically valid, can identify points in the execution where results are no longer up-to-date in order to resume execution from these points, can re-construct a workflow from an execution trace (satisfying TR 9), or can generate a textual description of an execution. Presentation user interfaces can, for instance, offer browsing facilities over provenance stores, visualise differences in different execution, illustrate execution from a more semantic viewpoint, visualise the performance of execution, and be used to construct provenance-based workflows. We note that such a list of processing services and presentation UIs is illustrative and not exhaustive.

Another kind of user interface to the provenance store is also identified in the architecture. This is the *management user interface*, which allows users to manage the contents of the provenance store.

To be generic, a provenance architecture must be deployable in many different contexts and must support user preferences. To adapt the behaviour of the architecture to the prevailing circumstances and preferences, several *policies* are introduced to help configure the system in its different aspects. Specifically: policies state user requirements about recording, e.g., to identify the provenance stores to use, the level of documentation required by the user, desired security aspects; policies specify capabilities of documenting process that services may wish to advertise (such as their ability to provide some type of actor states p-assertions), and any requirements they have on other services they rely upon in order to perform this documenting (such as their need for high throughput or highly persistent provenance stores); policies define configurations of provenance stores, from a deployment and security viewpoint (e.g., resources they use, their access control list, or registry where they should be advertised).

By making explicit all these policies, it becomes possible to *discover* services that *match* user or other service needs. When requested policies conflict with discovered policies, *negotiation* can be initiated to find a compromise between the offer and demand.

Non-provenance data stores are stores of data that do not relate to the provenance of a particular experimental result, and can be used in supplement to the services shown in the provenance architecture. The data may exist before any auditable experiment is run, examples include ontologies, which are used to provide semantic terms for testing the semantic validity of experiments, and user stored metadata that can be referred to by process metadata. Because, in our architecture, it can be processed along with the process documentation, this satisfies TR 4.

We believe this architecture addresses the functional requirements of the presented use cases. In future work, discussed in Section 7, we need to make the architecture robust enough to work as a production provenance system, in particular addressing non-functional TRs 11, 12, 13 and 14.

6. Concrete Implementations

We have created a first, basic implementation of the architecture, including a provenance store Web Service available to download from `www.pasoa.org`, and are beginning to evaluate its effectiveness in satisfying the use cases. Two distinct use case implementations are described here, one for the Intron Complexity Experiment, the other for the Service Reliability Experiment. For each, we describe the implementation and map the physical components involved to the elements of the logical architecture shown in Figure 3.

6.1. Semantic Validity

We have translated the Intron Complexity Experiment into a distributed workflow using the Globus toolkit and the Chimera Virtual Data System (VDS) [19]. We recorded provenance data from each of the services in the workflow and analysed the provenance data to determine whether the workflow run was *semantically valid*, as specified in Use Case 7. The services were a mixture of Tcl and UNIX shell scripts, and a wrapper script performed the recording of process documentation by extracting each script command line and sending it to the Web Service provenance store. The workflow was a Condor directed acyclic graph (DAG), generated from a definition of the dependencies between data encoded in the Virtual Data Language. Full details of the experiment can be found in [23].

For each experiment run, a processing service, written in Java, verified that the experimental process was *semantically valid*, which we explain as follows. For each service interaction recorded in the provenance store, we looked up, in a *registry*, the semantic types of the outputs of one service in the process and the inputs of the subsequent service. If the output of the former service has a compatible semantic type to the input of the latter, the interaction is semantically valid. If all interactions in a workflow, collected into a *session* in the provenance store, are semantically valid, then the workflow run as a whole was semantically valid.

We can use this set-up to help clarify the logical architecture shown in Figure 3. In the first column of Table 6.1, we refer to the name of one box in the logical architecture, while the second column describes how it was instantiated in the particular application.

Table I. Mapping of Logical Architecture to Semantic Validity Use Case Implementation

Logical Architecture Component	**Instantiation**
Provenance Store	Implemented as a single Web Service
Recording API	A port of the Provenance Store with record methods
Query API	A port of the Provenance Store with browsing and retrieval methods
Workflow Enactment Engine	Using VDS, workflows are enacted using Condor DAGMan
Domain-Specific Services	The protein analysis services implemented by the bioinformatician were a mixture of Tcl scripts and UNIX shell scripts
Actor-Side Recording Library	Implemented as a script wrapping each domain-specific service and recording the interactions via SOAP messages
Processing Services	A Java program that extracted the interaction provenance and checked semantic validity
Presentation Services	The results of processing were presented on the console as a yes or no answer
Non-Provenance Data Stores	The registry, implemented as a Web Service, containing semantic types of each domain-specific service's inputs and outputs

6.2. SERVICE RELIABILITY

In our second test, we chose to attempt to achieve Use Case 8, which asks a simple question of potentially complex process documentation. A far more detailed version of this evaluation was conducted by the scientists themselves and is discussed in [34].

We implemented three Web Services and a client as stated in the use case. We wrote all code in Java 1.4, used Axis 1.1 for all sending and parsing all Web Service calls and deployed the services on Tomcat 5.0. We used a single provenance store for all process documentation. Axis allows *handlers* to easily be introduced into the parsing of incoming and outgoing handlers, by modifying the deployment descriptor and including a JAR archive on the class path. Our architecture implementation includes an Axis handler that automatically sends to a provenance store every SOAP message that is received or sent by the service.

The message passed between each client/service in invocation or result is recorded in the provenance service by both parties in each interaction (via the Axis handler). To distinguish the calling of X and the calling of Y, we use two *tracers*, as illustrated in Figure 2. The first tracer is generated and recorded along with the interaction of C and X and with the interaction of X and Z. The second tracer identifier is generated and recorded along with the interaction of C and Y and with the interaction of Y and Z. Each tracer is perpetuated between services in the SOAP message header.

After X and Y have finished, C attempts to determine whether they used a common service. C queries the provenance store find the list of interactions that were recorded with the first session identifier, and from this data discovers which services were used. The same is then done for the other session identifier. Finally, C takes the intersection of the set of services used in the first session and those used in the second session, to produce the set of services used in both, and outputs this set. The set consists of a single element, the identity of Z, so C knows this was used by both X and Y.

The same process will work regardless of the complexity of the operation of X and Y. For example, X may call a long succession of other services in order to achieve its results, one or more of which occur in Y's operation also. The common set of services can still be discovered.

As with the previous use case, we can use this set-up to help clarify the logical architecture. In the first column of Table 6.2, we refer to the name of one box in the logical architecture. The second column describes how it was instantiated in the particular application.

Table II. Mapping of Logical Architecture to Service Reliability Use Case Implementation

Logical Architecture Component	**Instantiation**
Provenance Store	Implemented as a single Web Service
Recording API	A port of the Provenance Store with record methods
Query API	A port of the Provenance Store with browsing and retrieval methods
Domain-Specific Services	Web Services X, Yand Z
Actor-Side Recording Library	Implemented as a generic Axis handler, intercepting SOAP messages and recording them as interaction provenance
Processing Services	A Java program performing the task of client Cin determining whether common services were in use
Presentation Services	The results of processing were presented on the console as the set of services used by both X and X

7. Future Work

While the architecture described is a framework for satisfying use cases, there are many details to be resolved.

First, several non-functional requirements relating to storage of process documentation must be met, particularly the management of storage duration (TR 11) and storage of large quantities of data (TR 13).

There are a number of compelling reasons for distributing the storage of process documentation, as suggested in TR 12. First, our architecture should ensure there is not a single point of failure in providing access to process documentation. Further, we should allow service owners to keep data related to their service within their own security domain. However, as pointed out in Use Case 20, the architecture should provide a way to view data from multiple provenance stores in a unified way.

The PASOA architecture should ensure that the performance of the system does not significantly deteriorate as the number of provenance stores, process documentation, process documentation recorders or distribution of data increases. As indicated in TR 14, adapters for storing and querying process documentation may have to be provided to integrate our provenance architecture with other existing standards, software and protocols.

Finally, the current architecture does not address the needs of controlling access to the process documentation, which is essential for any real world deployment.

8. Conclusions

We have presented a broad range of use cases regarding the recording and use of the process documentation of scientific experiments. We have observed that there is little that spans all use cases, but many issues appear in a range of areas. Our proposed protocol and architecture attempts to separate the general from the application specific concerns and provide a framework for building solid recording process documentation, querying and processing software.

It is clear that we can provide generic middleware that allows the provenance-related use cases to be more easily achieved. We have separated the tasks supported by the architecture into recording, querying and processing, with each depending on the former. As far as possible, we intend to push application-specific solutions into the processing. While there are many issues still to be addressed, we believe our architecture provides the foundations of a full solution.

9. Acknowledgement

We wish to thank all that have contributed the requirements used in this paper. In particular, this includes Klaus-Peter Zauner for the Intron Complexity Experiment, David O'Connor and Paul Skipp for the Protein Identification Experiment, Mark Greenwood, Chris Wroe and Nedim Alpdemir for the Candidate Gene Experiment, Paul Townend for the Service Reliability Experiment, Hugo Mills for the Simple Harmonic Generation Experiment, and Ronald Ashri and Terry Payne for the Security Testing Experiment. This research is funded by the PASOA project (EPSRC GR/S67623/01).

References

1. : 2004, ‘Business Process Execution Language for Web Services Version 1.1’. http://www-128.ibm.com/developerworks/library/ws-bpel/.
2. : 2004, ‘e-Demand’. http://www.comp.leeds.ac.uk/edemand.
3. : 2004a, ‘GenBank’. http://www.ncbi.nlm.nih.gov/Genbank/.
4. : 2004b, ‘Gene Ontology Consortium’. http://www.geneontology.org/.
5. : 2004, ‘myGrid’. http://www.mygrid.org.uk.
6. : 2004, ‘PSI’. http://psidev.sourceforge.net.
7. : 2004, ‘Web Services Architecture’. http://www.w3.org/TR/ws-arch/.
8. Addis, M., J. Ferris, M. Greenwood, D. Marvin, P. Li, T. Oinn, and A. Wipat: 2003, ‘Experiences with eScience workflow specification and enactment in bioinformatics’. In: *Proc. of the UK OST e-Science second All Hands Meeting 2003 (AHM’03)*. Nottingham, UK, pp. 459–467.
9. Alonso, G. and A. E. Abbadi: 1993, ‘GOOSE: Geographic Object Oriented Support Environment’. In: *Proc. of the ACM workshop on Advances in Geographic Information Systems*. Arlington, Virginia, pp. 38–49.
10. Alonso, G. and C. Hagen: 1997, ‘Geo-Opera: Workflow Concepts for Spatial Processes’. In: *Proc. 5th Intl. Symposium on Spatial Databases (SSD ’97)*. Berlin, Germany.
11. Ashri, R., T. Payne, D. Marvin, M. Surridge, and S. Taylor: 2004, ‘Towards a Semantic Web Security Infrastructure’. In: *Semantic Web Services, AAAI Spring Symposium Series*.
12. Becker, R. A. and J. M. J. M. Chambers: 1988, ‘Auditing of data analyses’. *SIAM Journal of Scientific and Statistical Computing* **9**(4), 747–760.
13. Buneman, P., S. Khanna, K.Tajima, and W. Tan: 2002, ‘Archiving scientific data’. In: *Proc. of the 2002 ACM SIGMOD International Conference on Management of Data*. pp. 1–12.
14. Buneman, P., S. Khanna, and W. Tan: 2001, ‘Why and Where: A Characterization of Data Provenance’. In: *Int. Conf. on Databases Theory (ICDT)*.
15. Crawford, M. J., J. G. Frey, and T. J. VanderNoot: 1996, ‘Investigation of transport across an immiscible liquid/liquid interface - Electrochemical and second harmonic generation studies’. *J. Chem. Soc., Faraday Trans.* **92**(1369).
16. Cui, Y., J. Widom, and J. L. Wiener: 2000, ‘Tracing the lineage of view data in a warehousing environment’. *ACM Trans. Database Syst.* **25**(2), 179–227.
17. Fan, H. and A. Poulovassilis: 2003, ‘Tracing data lineage using schema transformation pathways’. In: B. Omelayenko and M. Klein (eds.): *Knowledge transformation for the Semantic Web*. IOS Press, pp. 64–79.
18. Foster, I., C. Kesselman, and S. Tuecke: 2001, ‘The Anatomy of the Grid: Enabling Scalable Virtual Organizations’. In: *Int. J. Supercomputer Applications*. pp. 15–18.
19. Foster, I., J. Vockler, M. Wilde, and Y. Zhao: 2003, ‘The virtual data grid: A new model and architecture for data-intensive collaboration’. In: *In Proc. of the CIDR 2003 First Biennial Conference on Innovative Data Systems Research*.
20. Foster, I., J. Voeckler, M. Wilde, and Y.Zhao: 2002, ‘Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation’. In: *Proc. of the 14th Conf. on Scientific and Statistical Database Management*.
21. Greenwood, M., C. Goble, R. Stevens, J. Zhao, M. Addis, D. Marvin, L. Moreau, and T. Oinn: 2003, ‘Provenance of e-Science Experiments - experience from Bioinformatics’. In: S. J. Cox (ed.): *Proc. UK e-Science All Hands Meeting 2003*. pp. 223–226.
22. Groth, P., M. Luck, and L. Moreau, ‘A protocol for recording provenance in service-oriented Grids’. In: *Proceedings of the 8th International Conference on Principles of Distributed Systems (OPODIS’04)*. Grenoble, France.

23. Groth, P., S. Miles, W. Fang, S. C. Wong, K.-P. Zauner, and L. Moreau: 2005, 'Recording and Using Provenance in a Protein Compressibility Experiment'. Submitted for publication.
24. Lanter, D.: 1991a, 'Design of a Lineage-Based Meta-Data Base for GIS'. *Cartography and Geographic Information Systems* **18**(4), 255–261.
25. Lanter, D.: 1991b, 'Lineage in GIS: The Problem and a Solution'. Technical Report 90-6, National Center for Geographic Information and Analysis (NCGIA), UCSB, Santa Barbara, CA.
26. Lanter, D. and R. Essinger: 1991, 'User-Centered Graphical User Interface Design for GIS'. Technical Report 91-6, National Center for Geographic Information and Analysis (NCGIA). UCSB.
27. Marathe, A. P.: 2001, 'Tracing Lineage of Array Data'. *J. Intell. Inf. Syst.* **17**(2-3), 193–214.
28. Myers, J., A. Chappell, M. Elder, A. Geist, and J. Schwidder: 2003a, 'Re-integrating the research record'. *IEEE Computing in Science & Engineering* pp. 44–50.
29. Myers, J. D., C. Pancerella, C. Lansing, K. L. Schuchardt, and B. Didier: 2003b, 'Multi-scale science: supporting emerging practice with semantically derived provenance'. In: *ISWC 2003 Workshop: Semantic Web Technologies for Searching and Retrieving Scientific Data*. Sanibel Island, Florida, USA.
30. Pope, A.: 1997, *The CORBA Reference Guide: Understanding the Common Object Request Broker Architecture*. Addison Wesley Publishing Company.
31. Ruth, P., D. Xu, B. K. Bhargava, and F. Regnier: 2004, 'E-notebook Middleware for Acccountability and Reputation Based Trust in Distributed Data Sharing Communities'. In: *Proc. 2nd Int. Conf. on Trust Management, Oxford, UK*, Vol. 2995 of *LNCS*.
32. Szomszor, M. and L. Moreau: 2003, 'Recording and Reasoning over Data Provenance in Web and Grid Services'. In: *Int. Conf. on Ontologies, Databases and Applications of Semantics*, Vol. 2888 of *LNCS*.
33. Tan, V. H. K.: 2004, 'Interaction tracing for mobile agent security'. Ph.D. thesis, University of Southampton.
34. Townend, P., P. Groth, and J. Xu: 2005, 'A Provenance-Aware Weighted Fault Tolerance Scheme for Service-Based Applications'. In: *Proceedings of the 8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC 2005)*.
35. Vahdat, A. and T. Anderson: 1998, 'Transparent Result Caching'. In: *Proc. of the 1998 USENIX Technical Conference*. New Orleans, Louisiana.
36. Waldo, J.: 2000, *The Jini Specifications*. Addison-Wesley Professional, 2nd edition.
37. Woodruff, A. and M. Stonebraker: 1997, 'Supporting Fine-grained Data Lineage in a Database Visualization Environment'. In: *Proc. of the 13th International Conference on Data Engineering*. Birmingham, England, pp. 91–102.
38. Woodruff, A. G.: 1998, 'Data Lineage and Information Density in Database Visualization'. Ph.D. thesis, University of California at Berkeley.
39. Zhao, J., C. Goble, M. Greenwood, C. Wroe, and R. Stevens: 2003, 'Annotating, linking and browsing provenance logs for e-Science'. In: *Proc. of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*.