Grant agreement no.211714

neuGrid

A GRID-BASED e-INFRASTRUCTURE FOR DATA ARCHIVING/ COMMUNICATION AND COMPUTATIONALLY INTENSIVE APPLICATIONS IN THE MEDICAL SCIENCES

Combination of Collaborative Project and Coordination and Support Action

Objective INFRA-2007-1.2.2 - Deployment of e-Infrastructures for scientific communities

Deliverable reference number and title: **D6.1 Distributed Medical Services Provision (Provenance Service)**

Due date of deliverable: **Month 12**

Actual submission date: **31st January 2009**

Start date of project: **February 1st 2008** Duration: **36 months**

Organisation name of lead contractor for this deliverable: **University of the West of England, Bristol UK**

Revision: Version **1**

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Table of Contents

.....2

Intended Recipients 3

8 The Provenance Service.....4

8.1 Introduction.....4

8.2 The User Requirements for the Provenance Service 5

8.2.1 Essential Requirements 6

8.2.2 Desirable Requirements 6

8.2.3 Optional Requirements 7

8.3 Existing Provenance Capturing Techniques and their usage in Provenance Service 7

8.3.1 Wings/Pegasus Provenance System 8

8.3.2 Provenance Aware Service Oriented Architecture (PASOA) 9

.....10

.....11

8.3.3 Job Provenance (JP) in gLite 12

.....13

.....13

8.3.4 Provenance Query and Answer (ProQA) 13

8.3.5 Matching of User requirements 14

8.3.6 Technical Requirements 16

8.4 The Provenance Service Design 17

8.5 The Provenance Service: An Example Case Study 21

Intended Recipients

The WP6 workpackage entitled **"Distributed Medical Services Provision"** aims to design a group of *generic* services that can be used in a number of related medical applications. These will then be implemented in order to fulfil the neuGrid specific project requirements. The services will be built according to the design philosophy presented in the WP6 deliverable. This will help to enhance and promote their re-usability in other related applications.

This deliverable document presents a design philosophy that the generic services will follow, maps user requirements against suitable services and briefly presents a list of the services. An initial implementation of the services and their detailed API descriptions will be delivered in the year 2 deliverable.

The WP leaders, technical users and neuGrid developers are the intended recipients of this document. To a lesser extent, since indirectly concerned (through the natural abstraction of Workflow/ Pipeline authoring environments such as the ones proposed in WP6), neuro-scientists and prospective users (e.g. Pharmaceutical companies) as well as internal and external reviewers of the project activities, are anticipated as potential readers of this document.

8 The Provenance Service

8.1 Introduction

The aim of the neuGrid project is to provide a user-friendly grid-based e-infrastructure, which will enable the European neuroscience community to carry out research necessary for the study of degenerative brain diseases. neuGrid will enable neuroscientists to archive large volumes of brain imaging data and perform analysis using a range of post-processing algorithms. Analysis is carried out using combinations of algorithms, which are integrated to form neuroimaging pipelines. For example in order to determine the thickness of a cortex, MINC [52] executables are combined together to form a cortical thickness pipeline. Such processes have a level of complexity that may allow small errors to occur, which cumulatively have a large impact on the validity of the results that are produced. Researchers therefore require a means of tracking the execution of given pipelines so they can ensure that important results are accurate. This is a manual task that is generally carried out before research is released to the wider community and is published. The provenance service is primarily intended to capture and provide the information that is necessary during this process.

The infrastructure will offer a Grid-based solution for such compute intensive pipelines, which will help to reduce the computational cost. In order to support neuroimaging analysis, a range of medical services will be developed. These will provide support for archiving the image data, creating pipelines in a user-friendly environment, and planning and gridification of pipelines. Furthermore such services will support the Grid execution of pipelines as well as provisions for capturing analysis information and querying it. Services will be designed in a generic way so that they have the potential to facilitate biomedical analysis in other projects (also described in Design Philosophy Document.) The process of neuroimaging analysis may involve a number of issues, which can cause an execution failure or undesired execution results. These may include incorrect pipeline specifications, inappropriate links between pipeline components, execution failures because of the dynamic nature of the Grid and others. A real problem in this scenario is tracking faults as and when they happen. This is mainly because of the absence of an information capturing mechanism during the pipeline specification, gridification and execution. Thus a user is unable to track errors in past neuroimaging analyses. This may lead to a loss of user control or repetition of errors during subsequent analyses. Users may face a range of problems in such cases, which may prevent them from being able to:

- Reconstruct a past pipeline or parts of it to view the errors at the time of specification.
- Validate a pipeline against a reference specification.
- Validate pipeline execution results against a reference dataset.
- Query information of his interest from the past analysis.
- Compare different analyses.
- Search annotations associated with a pipeline or its components for future reference.

To address the aforementioned problems, the generic medical services layer of neuGrid involves a process of keeping track of the origins of the data and its evolution between different stages and services. This process is called provenance and it will allow users to query analysis information, automatically generate analysis pipelines, detect errors and unusual behaviours in past analyses, and validate analyses. Tracking provenance data is important as it is useful in identifying the problems and errors that commonly occur during neuroimaging analysis. Provenance will support the continuous fine-tuning and refinement of the pipelines by capturing:

1. Pipeline specifications.
2. Data or inputs supplied to each pipeline component.
3. Annotations added to the pipeline and individual pipeline component.
4. Links and dependencies between pipeline components.
5. Execution errors generated during analysis.
6. Output produced by the pipeline and each pipeline component.

The neuGrid infrastructure will incorporate a provenance service, which will interact with other medical services through standardized interfaces. This service-oriented approach also allows centralized management and the on-line availability of the provenance service (Design Philosophy Document explains the features of SOA-based design of services in neuGrid). A conceptual model of the provenance service and a scenario of user interaction is shown in the Figure 40. The next few sections of this document explain the user requirements of the provenance service, the design, service components and relevant technologies that have been evaluated in the light of user requirements.

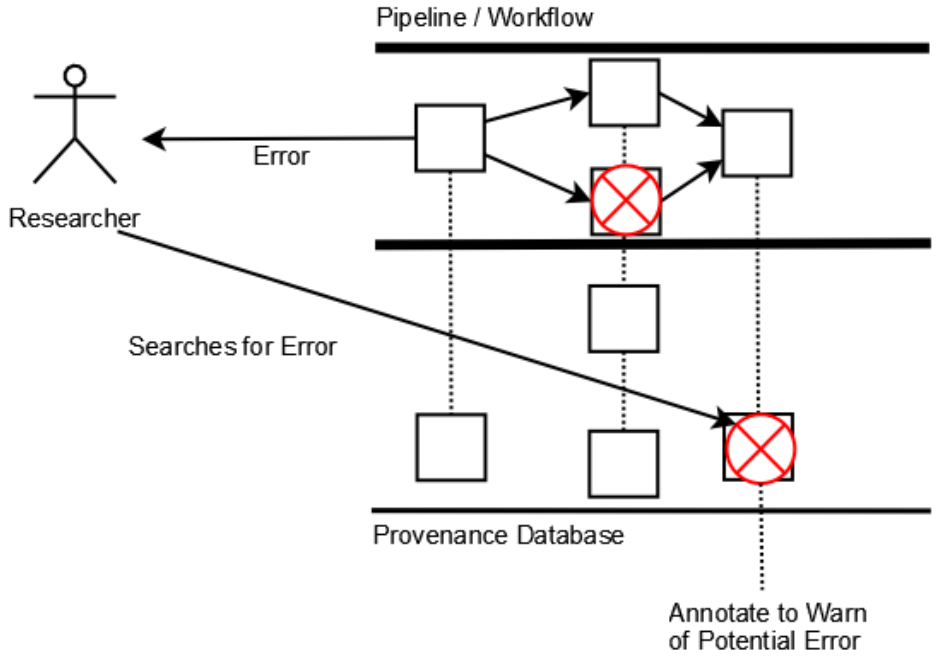

Figure 40: A conceptual model of Provenance Service

8.2 The User Requirements for the Provenance Service

The neuGrid consortium has identified a number of user requirements that are relevant to the development of the provenance service. These provide a set of functionality that end users of the infrastructure will expect to be available to them in the final system. The group of requirements mentioned in this section have been gathered by WP9 during discussions with user communities. Surveys and meetings have led to the creation of a range of stories, with each story explaining a process of user interaction with the service component/interface. Individual Use-cases have been extracted from the stories and this has allowed us to identify a number of individual user requirements. Requirements engineering is an iterative process and new requirements may be added as the provenance service evolves. Prototypes of the provenance service will be evaluated by user communities and the requirements will provide a means of evaluating the functionality that is developed. This process will help in identifying and meeting any elements that are missing. The user requirements are categorized in terms of priority as essential, desirable and optional

requirements.

8.2.1 Essential Requirements

Essential requirements are those that must be met by the provenance service. These define the core provenance service functionalities that are necessary in order to build an operational system. The major characteristics of such a system will be:

a. The business logic should be correctly implemented and tested to fulfil basic user needs.
b. It should be well deployed and available to the user communities.
c. Provide a preliminary user interface for interaction with the service.

8.2.1.1 Capturing Specification and Execution Information *(Ref. Prioritized use-cases and requirements: 6.1.1 and 6.1.2)*

This is an important requirement, which will allow a user to store a complete pipeline creation process. It will involve capturing each and every pipeline actor with its input and output details. The links between actors will need to be stored in order to make it possible for users to retrieve parts of a given workflow. All the intermediary steps of pipeline planning will also be captured. These deal with the gridification of pipelines in preparation for their execution on Grid resources. After the creation and planning processes, pipelines will be executed over the Grid. The result of an execution can be a success or failure. Therefore, it is important to store these results, error logs and other monitoring information that is provided by the Grid middleware. This will help a user to associate a particular analysis result with the specification.

8.2.1.2 Version Control Management *(Ref. Prioritized use-cases and requirements: 4.4.1, 4.4.2 and 4.4.3)*

Managing different versions of pipelines is necessary in order for experimental workflows to be gradually developed in a controlled process and eventually reach full maturity. A provenance store will need to make it possible for a user to track changes in different versions of a single pipeline specification. If several scientists/users are working on a particular analysis, versioning should help them to maintain the history of changes that occur in a pipeline. Therefore version control management is important for maintaining the ownership of pipelines and their evolutions with time. Versioning will also help users to analyse the difference in the outputs produced by different versions of a pipeline. These differences will be highlighted by the provenance data. All of this clearly requires a convenient means of capturing, storing and using versioning information.

8.2.1.3 Browsing and Validation *(Ref. Prioritized use-cases and requirements: 5.1.1 - 5.1.5)*

Provenance data is only useful if the information that is collected is made viewable to the user. This will include enabling the browsing of past traces of workflow specification and their respective execution information. Inaccurate specification at the time of pipeline creation or a resource failure during execution may produce erroneous results. Therefore the validation process will need to assist users to validate pipeline specifications against existing reference blueprints. This will also allow a user to validate execution results against a reference dataset. The integration of different aspects of provenance data is clearly useful in providing users with information that is of use to them during their research work.

8.2.2 Desirable Requirements

These requirements are clearly important in driving the provision of a well integrated and fully featured provenance service. Such requirements represent features which will enrich the functionality of the system. This will also enable users to benefit from the best possible mixture of features from the service. Desirable user requirements represent useful features which should

be supported once all the essential requirements have been addressed.

8.2.2.1 Querying and Searching interesting information *(Ref. Prioritized use-cases and requirements 6.3.1 - 6.3.3)*

The ability to query provenance data is an important requirement, as it will assist users in accessing information. This may involve the querying of data through SQL like statements. A user should be able to select a particular view of the data for specific analysis purposes. This requirement could be addressed through the provision of a querying interface on the top of provenance database. Users should be able to query data in a generic fashion, irrespective of which database is used at the backend and how the data is stored or maintained. The querying interface will help users to retrieve a selected dataset from the provenance store. Users are likely to want to search for interesting information such as what failure happened at what point and the reason of failure etc. This requirement could be met by making different views of provenance database, based on the search categories. It is clear that this will require the definition of what categorises interesting information. Such definitions will need to be extendable and flexible in order for them to evolve over time as the requirements of users develop.

8.2.2.2 Storing User annotations *(Ref. Prioritized use-cases and requirements 6.5.3)*

A user may want to add extra information into the provenance database regarding an actor or a pipeline. Enabling users to add annotations in the provenance database will satisfy this requirement. User annotations will serve as the metadata for the provenance data. It is likely that such information can be exploited elsewhere in the system.

8.2.2.3 Comparative analysis *(Ref. Prioritized use-cases and requirements 6.4)*

A user should be able to perform comparative analysis of the output produced in relation to a reference output. This requirement is similar to validation but in a comparative analysis a user does not know the authenticity of the reference output/dataset in advance.

8.2.3 Optional Requirements

Optional requirements in the provenance service will be addressed if the time and resources are available after meeting essential and desirable requirements.

8.2.3.1 Downloading Provenance Data *(Ref. Prioritized use-cases and requirements 6.4.3)*

A user should be able to download data objects in the provenance database, which may include input/output images and their associated metadata. This will enable a user to perform a statistical analysis on data or images.

8.2.3.2 Format Conversion *(Ref. Prioritized use-cases and requirements 6.4.5)*

Users will be provided with a tool to convert images into different formats so that they can visualize the results in a tool of their choice.

8.3 Existing Provenance Capturing Techniques and their usage in Provenance Service

Several projects are currently working to provide methods of maintaining an execution history for distributed workflows. These often differ from each other in scope and approach. A comprehensive survey of some popular provenance systems has been carried out. Such systems support the documentation of the process of scientific analyses. A thorough review and analysis of existing technologies has identified some individual components from within them, which are relevant to the design of the provenance service. Functionalities which are not provided by

existing systems nor have very specific requirements will be developed and integrated within the final service. This section focuses on some of these existing provenance techniques and evaluates them against what is required by the provenance service.

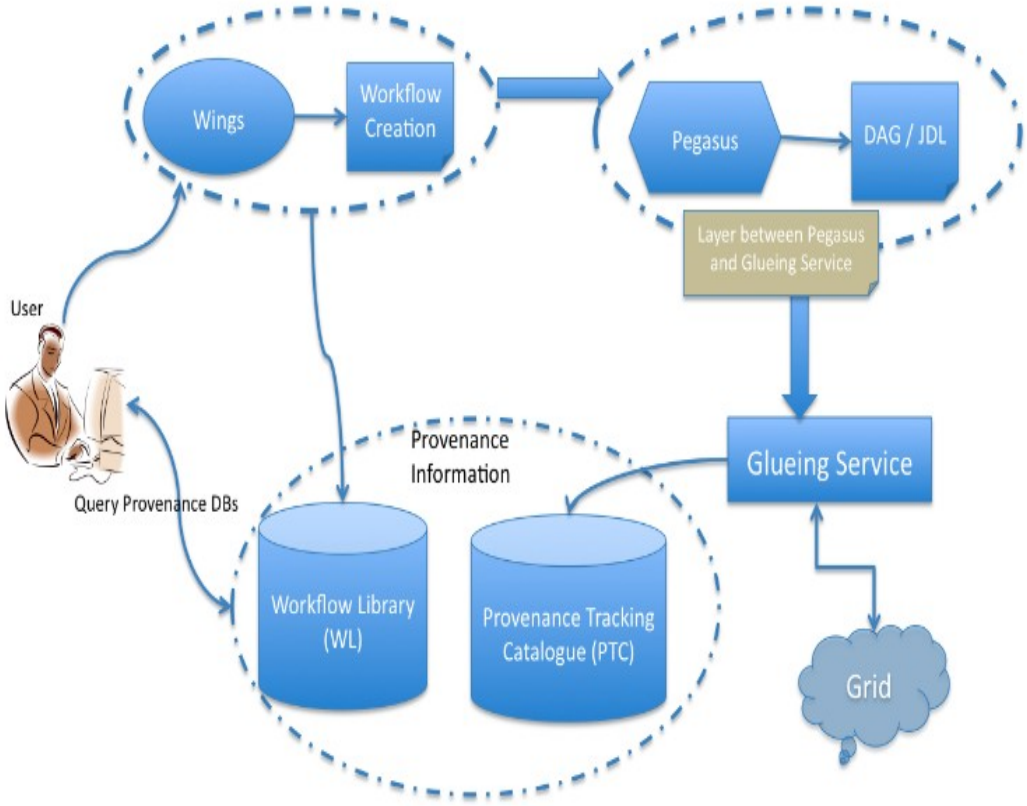

Figure 41: Workflow Refinement Process in Wings/Pegasus Provenance System

8.3.1 Wings/Pegasus Provenance System

The Wings/Pegasus framework [43] supports scientific workflows that involve a large number of computations on huge scientific datasets. The focus of this framework is to provide automatic generation, validation and resource selection facilities to deal with increasing computational jobs and data sources in scientific workflows. This system [44] produces provenance information at application and execution levels. Wings is used as workflow specification environment and uses semantic representations [45] to reason about application-level constraints and user annotations associated with the workflow. This execution independent workflow information/metadata not only helps in generating valid workflow specifications but also producing new workflow data products. All the provenance information regarding the workflow specifications is stored in workflow library (WL). Wings sends the workflow specifications to Pegasus, which then maps the specifications over the available Grid resources. This process involves various refinement processes such as workflow partition, site selection, data staging, data registration and job clustering. The refinement process in Pegasus is shown in the diagram below. The initial workflow specification is partitioned into various workflow instances, considering the dependencies in them. The individual workflow instances are then passed through the process of refinement and are eventually submitted to Condor DAGMan execution engine. This process continues for all the workflow instances. The execution level provenance information is stored in the provenance tracking catalogue (PTC).

Upon successful executions the job executable name, arguments start time for execution, duration of each job and compute element (CE) information is stored in PTC. For an unsuccessful execution the error message and exit status is stored. Wings/Pegasus provenance framework is shown in Figure 41.

8.3.1.1 Wings/Pegasus-based Architecture

The Wings/Pegasus framework passes workflow DAGs to the Condor DAGMan execution engine. This is not in line with the technical requirements of project as the workflow specification and execution should be middleware independent. Therefore, a software layer can be added to interface Pegasus with the Glueing Service. This architecture is shown in figure 42.

Pros	Cons
Support Task-based WFs	Ontology and semantic representation have no use in the project
Specification and execution logging	Different query interfaces for WL and PTC
	Extra overhead of interfacing Pegasus with Glueing Service
	Provenance information is stored in a non-customized database schema

Figure 42: Wings/Pegasus-based Provenance Architecture

8.3.2 Provenance Aware Service Oriented Architecture (PASOA)

PASOA [46] is a provenance capturing mechanism in web services environment. Therefore, this architecture mainly supports provenance in service-based workflow management systems [42]. The workflow dataset transformations are recorded during workflow execution. PASOA manages all the provenance recordings in a provenance server. It provides a client API to allow client applications to interface with the server. The provenance server holds the provenance information and provides methods to access and update this information through a web service. The provenance information is stored in a relational database, as shown in Figure 43.

The client API is responsible for submitting the web service invocation credentials, as the workflow enactment engine executes the workflow. The service credentials are extracted from the workflow script, represented in a workflow language such as WSFL or BPEL4WS. The workflow script describes the order in which the web services are invoked and also defines the control and data flow between services. PASOA also provides a browsing interface, which allows users to navigate through provenance traces of past workflow executions. A provenance reasoning facility is also provided in order to validate the workflow execution results. The validation process allows a user to check if the web service invocations produce the same results as in the past workflow executions.

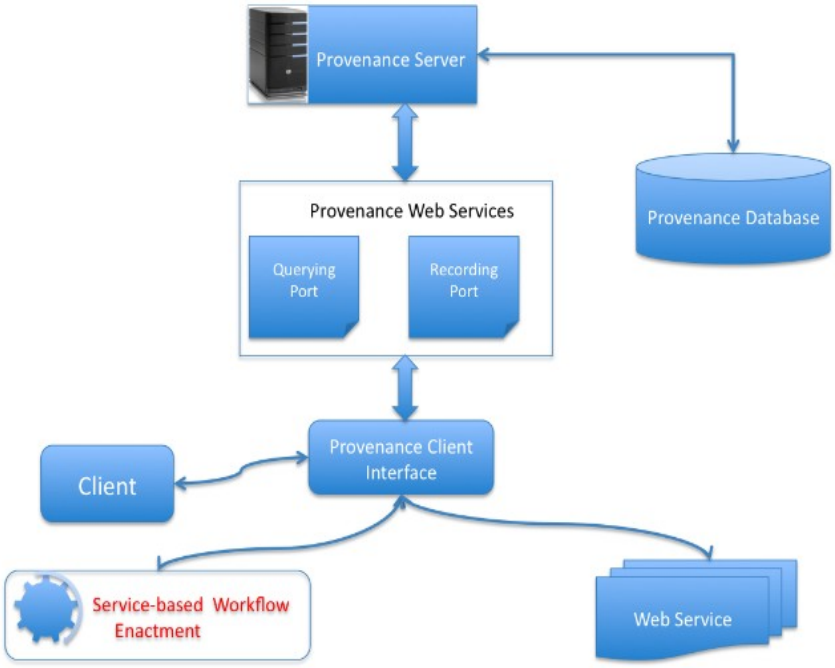

Figure 43: PASOA Architecture

PASOA handles this process by re-invoking a service and a difference in results in notified, which is also logged in provenance database. A screenshot of browsing and validation interface is shown in Figure 44.

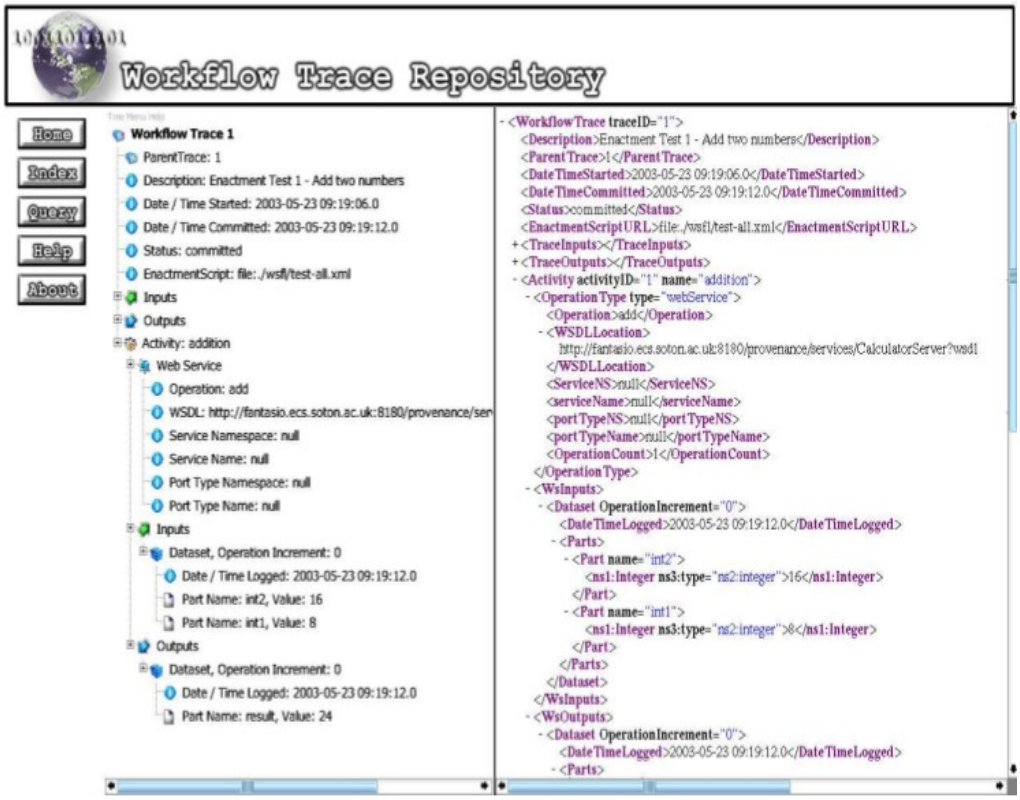

Figure 44: Browsing interface in PASOA

8.3.2.1 PASOA-based Architecture

PASOA stores execution logs and specification provenance in an XML database, called Provenance Database. PASOA's client interfaces provide APIs to populate XML database with provenance information. As PASOA supports remote execution in a service-based environment therefore its APIs are tied with service-based workflow systems. Whereas, technical requirements of project require a provenance system to capture execution logs of task-based workflows. Moreover in neuGrid the pipeline execution takes place via glueing service therefore client APIs should be able to interface with glueing service. This section proposes an architecture, which uses PASOA client APIs for capturing specification provenance only. The execution logs are stored in a relational database and this database is populated by glueing service, through the client interface shown in Figure 45. Moreover specification level provenance can be merged with the relational database through XML to relational translations, to have a single database view.

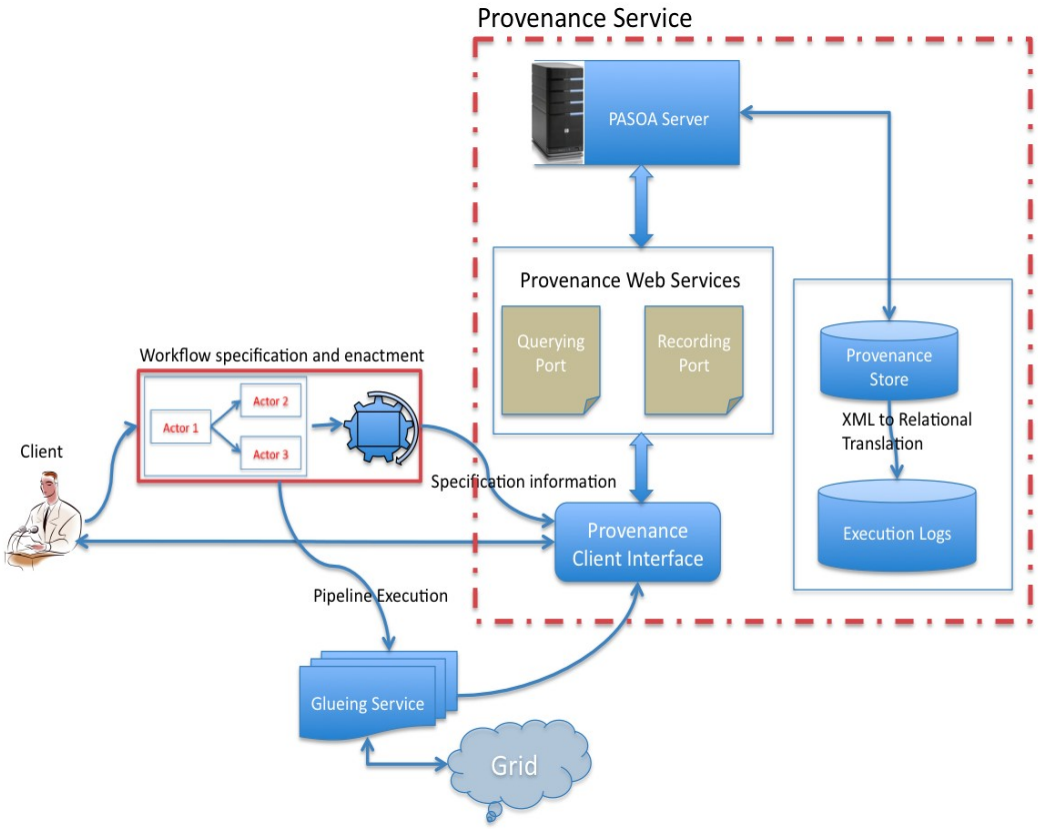

Figure 45: PASOA-based Provenance Architecture

Pros	Cons
Support Task-based WFs	Partial usage of PASOA Client APIs
Specification and execution logging	Overhead of writing XML to relational translator
Database schema is customizable	
PASOA provides a browsing interface to browse provenance traces	

8.3.3 Job Provenance (JP) in gLite

Job Provenance (JP) [47] is a component of gLite Grid middleware and developed with in EU EGEE project. The motivation of JP is to verify the workflow execution results by redoing the experiment. JP is a job-centric service and collects enough information about job life cycle, inputs/outputs, user annotations etc. This information is kept in a Logging and Bookkeeping (L&B) service, which enables JP to re-execute a workflow/job. JP keeps a long-term trace on the completed workflow computations, irrespective of space constraints in L&B. JP organizes the data collection, for re-executing the job/workflow, in three ways (i) It stores all the inputs to the job/workflow, JDL (Job Description Language) and the job input files, which are fetched from the middleware sandbox (ii) It keeps a complete execution trace on the compute element (CE) i.e. when and where the job is planned and executed, job submission count and environment settings on CE (iii) It keeps a record of user annotations added on the job or workflow. A data flow in gLite job provenance is shown in Figure 46.

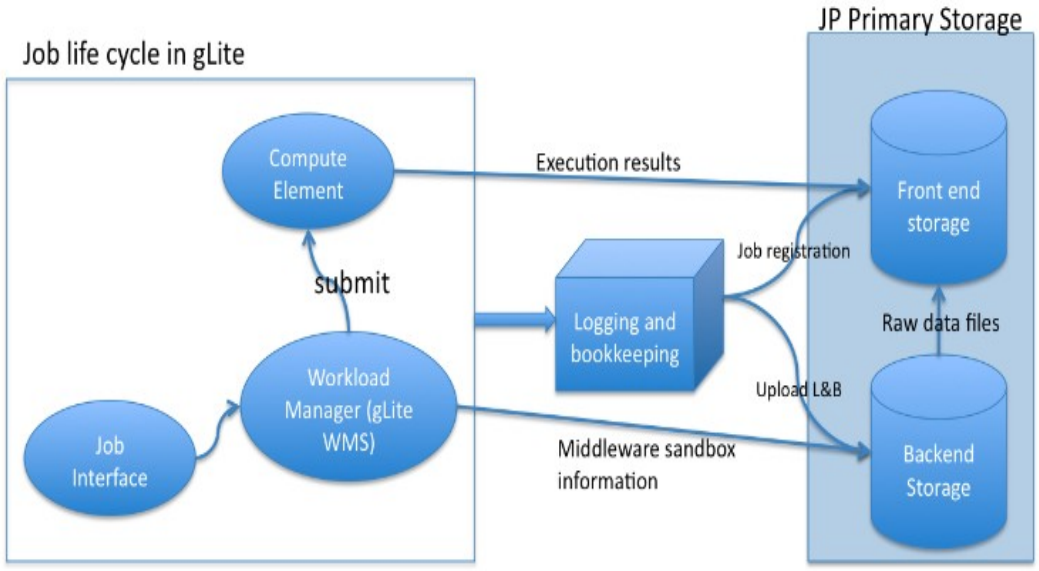

Figure 46: Data flow in JP

JP mainly focuses on the job re-execution of a job from the provenance data and does not provide any intelligent information from its provenance store. This provenance system is also tightly integrated with gLite middleware and cannot interoperate with other Grid middleware.

8.3.3.1 JP-based Architecture

This architecture is tied with gLite and therefore it does not involve glueing service for workflow execution. The pipeline specifications are directly passed to the user interface of gLite, after appropriate translations into JDL/DAG. The gLite Workload Manager System (WMS) schedules the workflows/jobs for their execution on CE. The execution logs are stored in L&B where JP adds JDL/DAG related data. JP also adds workflow annotations in L&B, after the execution is performed. This architecture is shown in Figure 47.

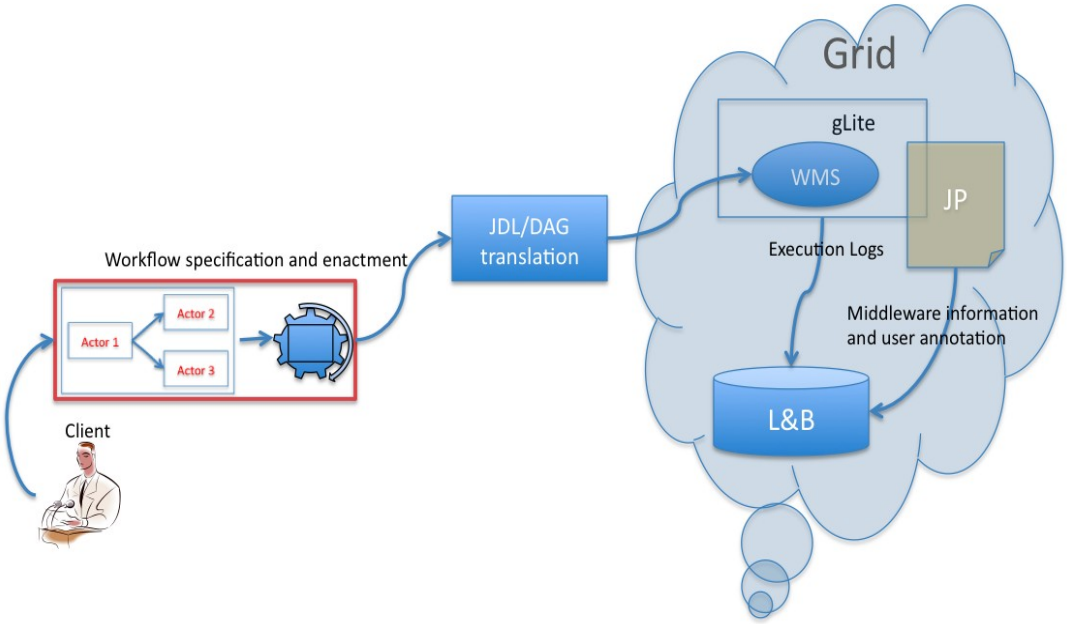

Figure 47: JP-based architecture

Pros	*Cons*
Support Task-based WFs	Middleware specific approach
Simple to implement	Execution only provenance

8.3.4 Provenance Query and Answer (ProQA)

ProQA [49] is a prototype provenance system and implemented under the context of Taverna [42] workflow workbench, which is targeted for bioinformaticians. ProQA supports provenance retrieval as well as provenance abstractions, aggregations and semantic reasoning. This system uses various third party tools to support its provenance mechanism such as RDF Access API, Analysis API, Core API and knowledge template plug-in. ProQA defines ontological representation of provenance data and this information is then represented in graph structure using resource description framework (RDF). A provenance graph is generated from the initial execution of workflow and upon the subsequent executions the graphs are merged into multi and mega graphs. This way ProQA forms a Provenance Web, where everything is represented by a Life Science Identifier (LSID) [50]. A detailed architecture of ProQA is shown in Figure 48.

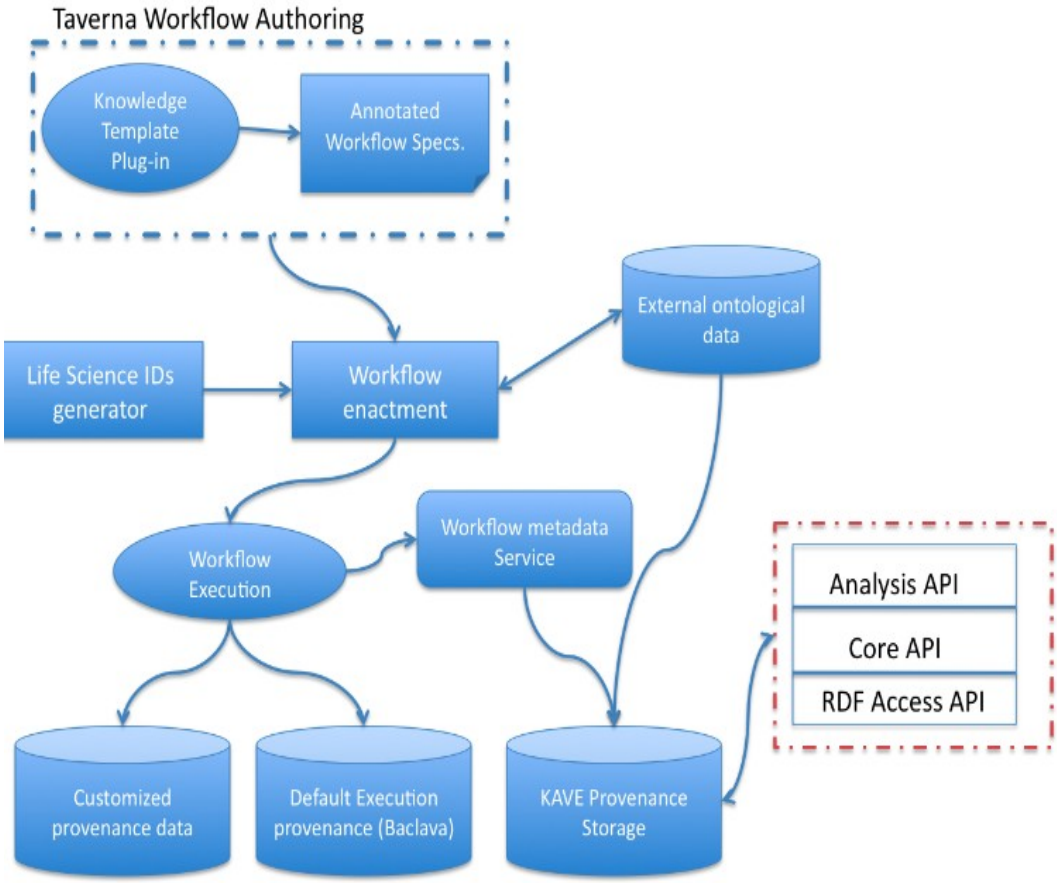

Figure 48: ProQA Architecture

The knowledge template plug-in, provided by Taverna, allows the users to annotate additional metadata with the workflow. The workflow specifications along with the annotated metadata are passed to workflow enactment engine for execution. Taverna assigns an LSID to each workflow component before its execution, and this LSID serves as a reference for each workflow component. The execution information is stored either in a customized relational database or in Baclava storage, which stores default execution information. Workflow metadata or external ontological representations are stored in KAVE [51] storage. An external API set, as shown in Figure 48, are used to query provenance information, build multi and mega provenance graphs and analyse the provenance information in these graphs.

8.3.5 Matching of User requirements

Previous sections describe a summary of the state of the art provenance systems. The possible architectures for the development of provenance service and pros and cons of each architecture were also described. This section focuses on matching user requirements against each of the proposed architecture.

Essential Requirements

		User Requirements			
		Specification Level Provenance	Logging execution information	Browsing execution information	Validate a workflow using provenance data
Possible Provenance Architectures	Wings / Pegasus System	**Supported**	**Supported**	**Not Supported**	**Supported**
	PASOA	**Supported**	**Supported**	**Supported** [2]	**Supported**
	JP	**Not Supported** [1]	**Supported**	**Not Supported**	**Supported**
	ProQA	**Supported**	**Supported**	**Not Supported**	**Supported**

Table 8: Matching Essential requirements with existing provenance system

1. JP has no interface for the workflow authoring environment and therefore it does not support capturing workflow specification provenance.
2. PASOA provides a web interface to browse workflow traces, of past workflow executions. The provenance trace browser is also shown in Figure 44.

Desirable Requirements

		User Requirements				
		Query Interface	Annotate workflows	Comparative analysis of output data produced [1]	Control versions of workflows	Searching interesting information for user [2]
Possible Provenance Architectures	Wings / Pegasus System	**Supported**	**Supported**	**Not Supported**	**Supported**	**Not Supported**
	PASOA	**Supported**	**Supported**	**Not Supported**	**Supported**	**Not Supported**
	JP	**Supported**	**Supported**	**Not Supported**	**Supported**	**Not Supported**
	ProQA	**Supported**	**Supported**	**Not supported**	**Supported**	**Not supported**

Table 9: Matching Desirable requirements with existing provenance system

1. All the provenance techniques mentioned in section 8.3 do not support a comparative analysis of the output data to a reference data set. This is an internal project requirement, which can be addressed by developing a tool on the top of provenance database.
2. Enabling a user to search interesting information, such as execution failure at a certain point, is another internal project requirement.

Optional Requirements

	User Requirements	
	Statistical Analysis of	Format Conversion Tools [2]

		provenance data	
Possible Provenance Architectures	Wings / Pegasus System	**Not Supported**	**Not Supported**
	PASOA	**Not Supported**	**Not Supported**
	JP	**Not Supported**	**Not Supported**
	ProQA	**Supported** [1]	**Not Supported**

1.

Table 10: Matching Optional requirements with existing provenance system

1. The multi and mega graphs in ProQA, built from ontological data, help in the analysis of provenance data.
2. Format conversion tools allow a user to convert the output into appropriate formats, which can then be used for visualizing the execution results.

8.3.6 Technical Requirements

Besides the user requirements, the neuGrid project has few technical requirements too, which should be considered when designing any generic medical service. These requirements are:
1. In neuGrid pipelines are gridified and then executed over the Grid. Therefore the provenance service should be able to store remote execution results.
2. The services should be middleware agnostic that is the pipelines should be able to run on any Grid middleware. This implies that the provenance service should also be able to store job execution information without being tied to any particular middleware.
3. The processes/components in a pipeline are tasks rather than services. Therefore, the provenance service should have support for task-based workflows.
4. The storage system should be a relational database, enabling a user to retrieve provenance information/results through SQL queries.

The following table matches technical requirements with the existing provenance systems, discussed in section 8.3.

1. The provenance systems, mentioned in section 8.3, are mainly integrated with some middleware to capture execution level information.
2. PASOA and ProQA support capturing provenance of service based workflows

Technical Requirements	Wings/Pegasus	PASOA	JP	ProQA
Distributed execution	Yes	Yes	Yes	Yes
Grid Middleware Agnostic	No[1]	No[1]	No[1]	No[1]
Task-based Workflow	Yes	No[2]	Yes	No[2]
Relational Database	Yes	No	Yes	Yes

Table 11: Matching technical requirements with existing provenance systems

The survey of provenance literature has helped in identifying essential components of an effective provenance system. The objective of this review was to explore that how some popular provenance capturing techniques have been implemented. Different possible architectures, for the provenance service, have been studied and analyzed based upon their advantages and disadvantages. Each provenance architecture, using existing systems, has been matched with all three sets of user requirements. This practice has helped in finding provenance architectures, which are closer to or inline with the project requirements. This has also provided suggestions that which of the components in existing systems can be used in the design and which need to be developed. Therefore an analysis of provenance systems, their proposed architectures and their match with the user requirements has played an important role in designing the provenance service.

8.4 The Provenance Service Design

The design of the provenance service has been derived from the initial set of requirements that are described in section 8.2. The current design focuses on fulfilling all the essential and desirable requirements. Once these requirements are met, optional features will be added in the provenance service as described in section 8.2.3. Figure 49 shows the design diagram where on one side of the provenance service is client applications and on the other side of it are grid resources. The client applications are essentially other WP6 services which would be responsible for the creation of analysis pipelines and their execution over the Grid. These along with grid resources will populate the provenance database through recording interfaces. Later the analysis history of pipelines will be provided by the querying interfaces of the provenance service. A case study of the interaction of the provenance service with different WP6 services is shown in section 8.5.

Generic medical services (WP6) in neuGrid are based on a Service Orientated Architecture (SOA). An SOA means that developers need to deploy services within a service layer. The core business logic may reside on different application servers and interfaces are published in the system. SOA is a commonly used architecture for the component-based development of a specific application process. This enhances the mobility of code, as the user transparently accesses services without knowing where they are actually located. This process is facilitated by the lookup service and dynamic binding. Location transparency enables multiple instances of a service running on different servers. Therefore if one server goes down the requests are redirected to another one without users being unduly affected. Moreover, SOA improves the structuring of the development process by encouraging the development of service layers. This allows specific roles to be defined for different developer communities. For example, business logic developers' work may work within the services layer whilst at the same time, front-end designers are developing application interfaces in the user layer. The benefit of this approach is that different development teams can work in parallel. This also improves application maintainability and abates the difficulty of finding and correcting errors in code. The service logic in an SOA model will enable neuGrid services to communicate with each other through their standardized interfaces. This will also help other services to use/reuse functionalities provided by a specific service to accomplish a particular task. The provenance service is designed in such a way that its components will be available to all the other generic medical services in WP6. These components will be handled through separate interfaces so that each interface is targeted for a specific group of users, although these interfaces will be available to all user communities.

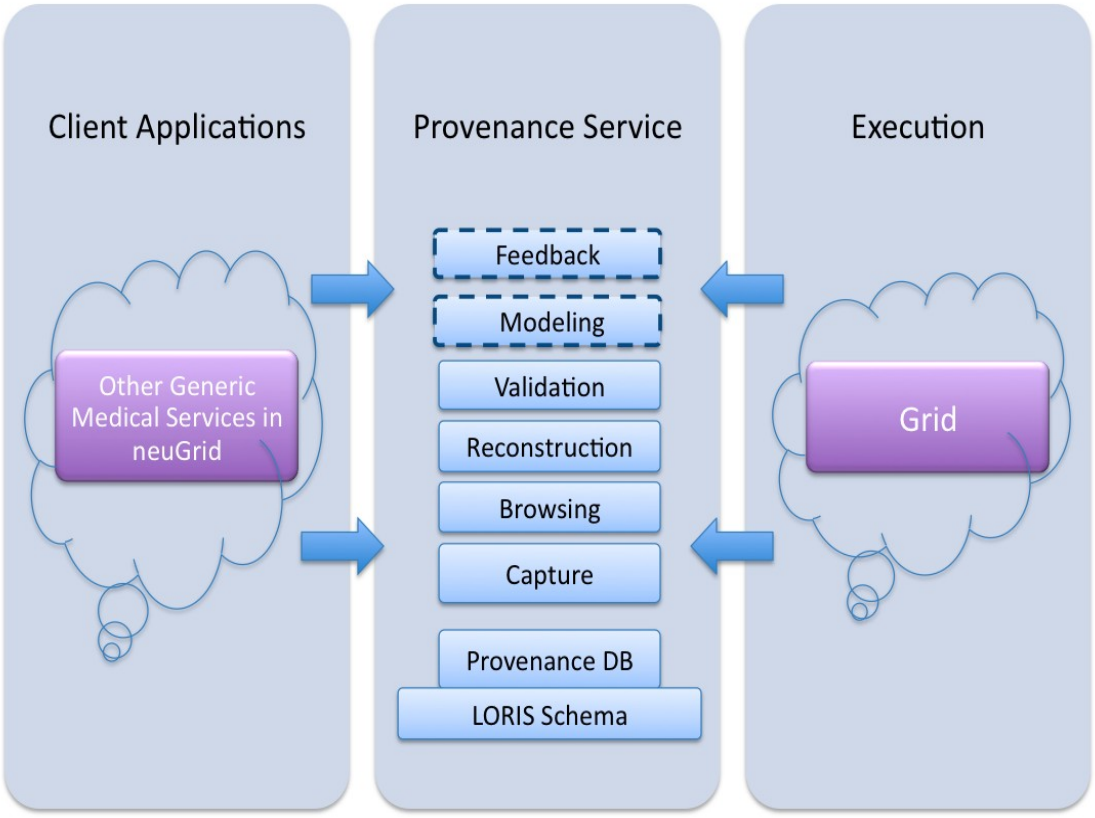

Figure 49: Provenance Service Design

The individual components of the provenance service are shown in Figure 49 and explanation of each is given below:

8.4.1 LORIS Schema and Provenance DB

LORIS (On-line Research Imaging System) is a system which was designed for the collection, management and processing of brain imaging data. It has been developed using a range of open source software such as Apache, PHP and MySQL. The system has two major components, a database schema for storing brain images and their associated metadata and a web-based portal which provides access to the LORIS database. In the context of the provenance service the primary role of LORIS will be to input the required MRI scans and metadata to neuroimaging pipelines, and then store the results/images of pipeline executions. The database schema of LORIS follows a relational model and therefore it can be easily extended. The LORIS schema has been specifically designed in order to simplify its customization for different tasks. This may also allow it to serve as a repository for pipeline specifications, relations between pipeline components, user annotations, and the input files that are supplied to them.

8.4.2 Capturing

Capturing provenance will be a key component of the provenance service, as its interface will allow a user/client service to store specification-level provenance and execution logs into the provenance database. Section 8.2.1.1 identifies the main components that will be captured by the

provenance service. Section 8.6.1 highlights some important API functions, which will facilitate the pipeline capturing/recording process. A basic overview of the pipeline capturing process is shown in Figure 49-A. The capturing component of the provenance service will be able to store:

- Pipeline descriptions and version information.
- The head node of the pipeline.
- The input data supplied to the head node.
- A script or process associated with a workflow node.
- The type of a workflow node i.e. single process node or composite node.
- The successors of a workflow node.
- The predecessors of a workflow node and input data supplied to it.
- Metadata associated with each workflow node.

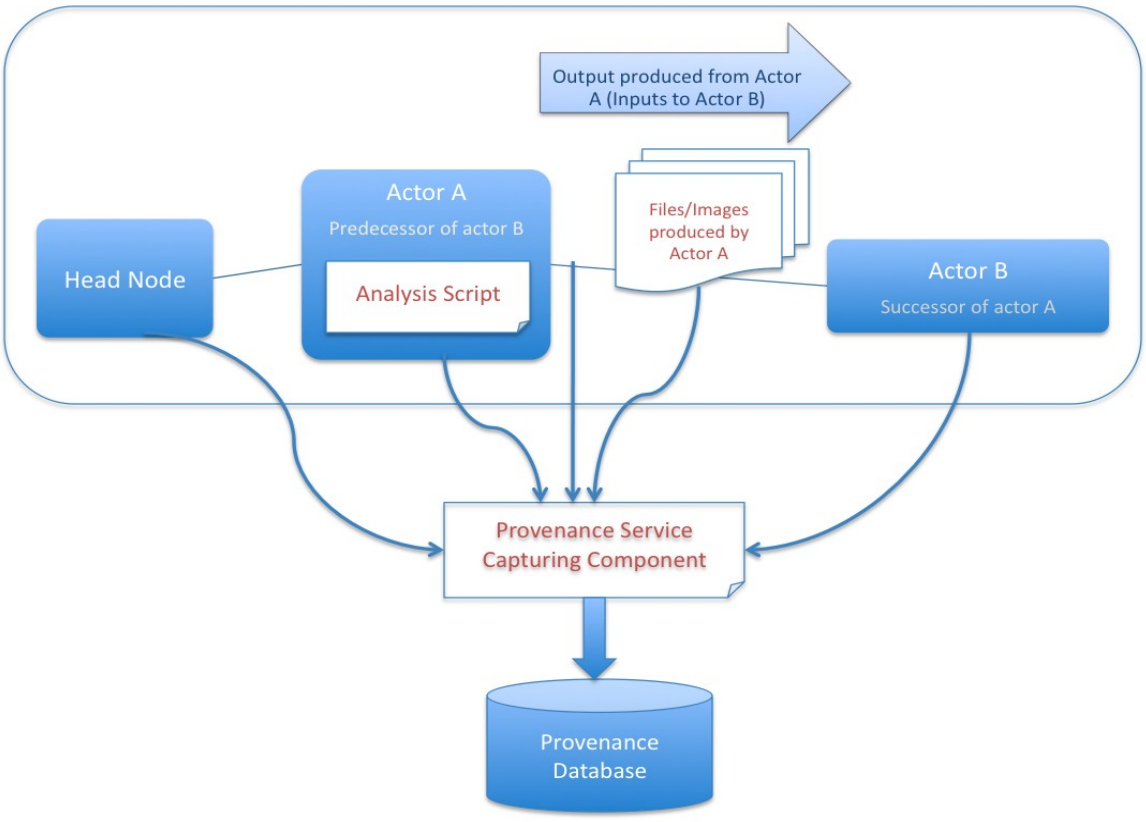

Figure 49-A: Capturing components of a pipeline

8.4.3 Browsing and Reconstruction

The browsing component will be built on the top of the provenance database, which will serve as

a utility for the users to browse the past pipeline traces. Browsing is not itself a core component of the provenance service, rather it is a project requirement which will help users to interact with and use the provenance database in a simplified way. A screenshot of the browsing component in PASOA [46] is shown in Figure 45.

A pipeline comprises a start node, tasks or actors, successors and predecessors of an actor, links/relations among actors, input data and files supplied to each actor, and a final output of complete pipeline. The provenance database will store each of the pipelines constituent parts. This will enable a user to retrieve a complete pipeline from the provenance store and also examine sections of it with the appropriate dependencies. This is important, as it will allow users to examine the various stages in the pipeline creation process even if they cannot remember each and every step that they originally took. The reconstruction APIs will help a user to reconstruct a pipeline or part of it for different purposes such as:

- Observing the pipeline creation process in past
- Re-executing a pipeline or part of it
- Modifying a pipeline and storing it with a different version

Section 8.6.2 highlights some important API functions for reconstructing and re-executing the pipelines.

8.4.5 Validation

Users will specify neuroimaging pipelines by combining different analysis algorithms. The pipelines will be gridified and the algorithms will then be executed over the Grid. The pipeline designer/creator may define inappropriate links between different components. A change to an analysis algorithm, residing on grid sites, may not always propagate to the user end. Therefore at the time of creating pipeline its author is not aware of any change in the logic of an algorithm. In these situations a user may receive corrupted or outdated execution results. The validation component of the provenance service will enable a user to verify a current pipeline specification against a reference blueprint. It will also allow a user to verify the results of an execution using a reference dataset. This type of validation will be performed in two ways: a. Re-executing algorithms in the pipeline and then comparing the results of the execution with the reference/expected output will perform an online validation of the results. b. Offline validation will verify the results of an already executed pipeline, in the provenance database, with a reference dataset.

8.4.6 Modelling and Feedback

The Modelling and Feedback components are shown in dashed boxes in Figure 49. These are optional features of the design and will be developed when all other components are available. The Modelling component will analyse and group the provenance information that is stored in the provenance DB. This will lead to the identification of groups of data that represents different patterns that occur during pipeline specification and execution behaviours. The modelling process will harness machine learning algorithms in order to categorize information segments that are present in the provenance DB. Different techniques of evolutionary computing may be applied to identify new information chunks and thus modelling will be a dynamic and evolving process. The feedback component will be developed using the modelling information. The primary objective of this component will be to provide users with useful information that is driven by the modelling process. The role of this component will be similar to a decision support system, which will help in pipeline construction and the planning process. This may, for example, suggest the use of a specific analysis pipeline or recommend computational resources on which to run the job at a

given time. Both the modelling and feedback processes will be stochastic and their efficiency will improve as the provenance DB grows. The modelling and feedback processes are also described in section 8.7.

8.5 The Provenance Service: An Example Case Study

Section 8.4 mainly describes the components of the provenance service (ProS) and how these components will facilitate other generic medical services in neuGrid, as shown in Figure 49. This section explains a scenario in which two of the generic medical services (the Pipeline and Glueing Services) feed data to the provenance database and then retrieve the provenance information using ProS interfaces, as shown in Figure 50.

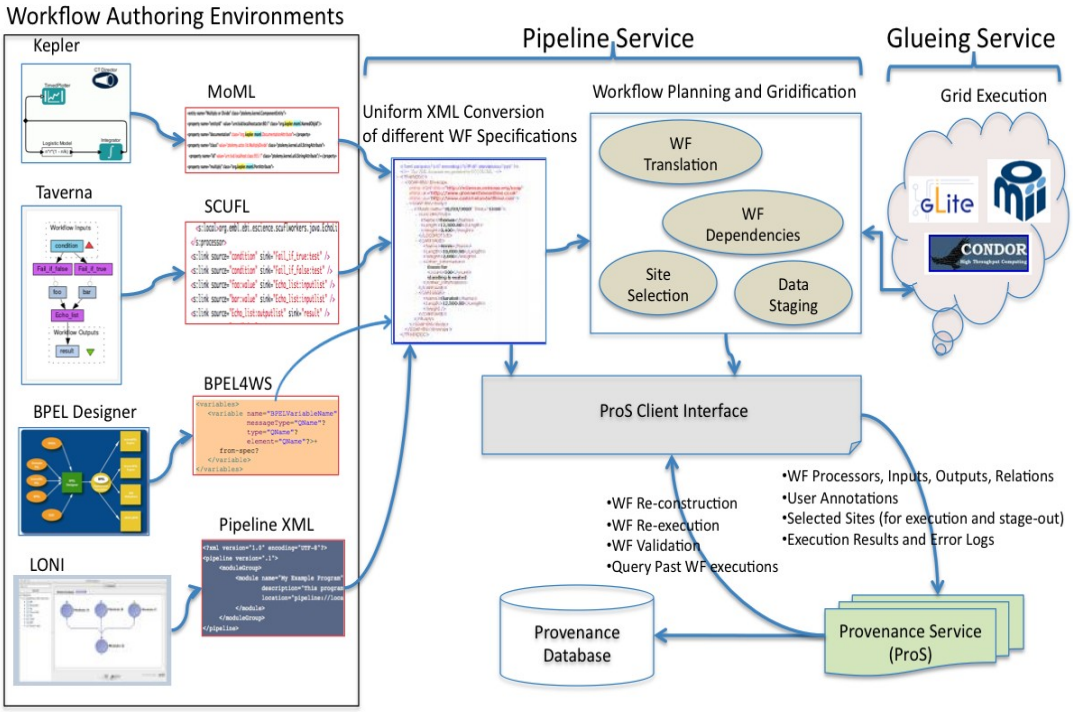

Figure 50: Provenance Service deployed in neuGrid

Pipelines will be constructed in different workflow/pipeline authoring environments. The workflow specifications are represented in different forms of XML, depending on the authoring environment such as MoML, SUFL, BPEL4WS etc. The Pipeline Service will translate all these representations into a uniform XML specification. This will then be passed to the planning component of the pipeline service, where the pipelines will be gridified in preparation for execution. The gridified tasks/components will be passed to the glueing service, which will perform a middleware agnostic execution of pipelines. The post execution logs, monitoring information or errors (if any) will be passed back to the pipeline service. The pipeline and glueing services design documents explain the process of pipeline gridification and middleware agnostic execution in detail. The pipeline specification, planning information and execution results will be stored in the provenance database through ProS client interfaces, which are described in section 8.6. Figure 50 explains the interactions of the Pipeline and Glueing services with the provenance service. The salient features of the provenance database are that it:

- Enables the maintaining of different versions of workflows.
- Links annotations and execution information with particular versions of workflows.
- Maintains a relational model of the different parts of a workflow and the associations

among them.
- Allows the retrieval of a complete workflow specification, for reconstruction, re-execution or validation.
- Helps in detecting unusual execution results.
- Could also support analysis and reasoning by modelling past workflow execution behaviours.

8.6 The Provenance Service Interface

The Provenance service will have two primary service interfaces. These interfaces are (1) the Recording Interface and (2) the Querying Interface. Each interface will provide a distinct set of functionalities as described in section 8.4. These interfaces will be exposed in the form of WSDLs, where each of them is focused for a specific user need. For example, a user who is interested in logging/capturing a pipeline construction process will interact with recording interface. On the other hand a user, who wishes to retrieve information about a particular analysis, will interact with querying interfaces only. This approach of keeping different set of functions in separate interfaces will make the service more manageable and easy to use. A detailed description of each service interface is given sections 6.1 and 6.2.

8.6.1 The Recording Interface

The recording interface will enable a user to store specification level information. Provenance APIs, for recording pipeline specification, facilitate a client application to log:

- Versioning information

Recording APIs will help a user to associate versioning descriptions with pipelines. This serves as the metadata for a given pipeline and will include information such as version number, name of the author, creation date and other details.

- Head node

Storing the head node of a pipeline is important in order to locate the start of an analysis. The provenance service recording APIs will allow a user to store the head node, which is essential for retrieving a complete pipeline with all its successor nodes and the links between them.

- Inputs to the head node

Input files and data supplied to head node will also be logged via recording APIs.

- Analysis scripts

Pipeline nodes/actors will represent a task in neuroimaging analysis. These tasks are usually algorithmic scripts, whose output is passed to the successor nodes in the pipeline. The recording APIs also help a user to store these scripts in the neuGrid database.

- Node types

Nodes in a pipeline can be of two types (i) a single process, (ii) a composite nodes. Single process nodes in a pipeline represent a granular task that is responsible for a specific part of the main

analysis. Whereas composite nodes are constituents of two or more single process nodes. The recording API will store the type of each node, which facilitates the reconstruction of a pipeline or its components.

- Predecessor nodes

Inputs to a node, in a pipeline, may be more than simple data literals or files and could include the output of predecessor node/nodes. This is shown in Figure 51 where MincDefrag is the predecessor node of MinDefrag2. In order to store pipelines at a fine grained level, such dependencies should be logged via the recording API.

- Successor nodes

Each node, single process or composite, in a pipeline can pass its output to a successor node. This is shown in Figure 51 where CorticalSurface is the successor node of MincDefrag2. In the process of regenerating a pipeline the successors of the start node and nodes after it will be retrieved recursively. Storing successors and predecessors in a pipeline is important as this will maintain a hierarchy during the reconstruction process.

- Annotations

The recording API will also provide a mechanism for the annotation of workflows. This will help users to associate metadata, in key-value pairs, with a pipeline or its components. Users will be able to retrieve annotations/metadata and view the past observations of an experimenter or workflow author. Annotations will also help users to define new rules for a given pipeline. For example a user might want to restrict a group (of users) to perform a particular analysis.

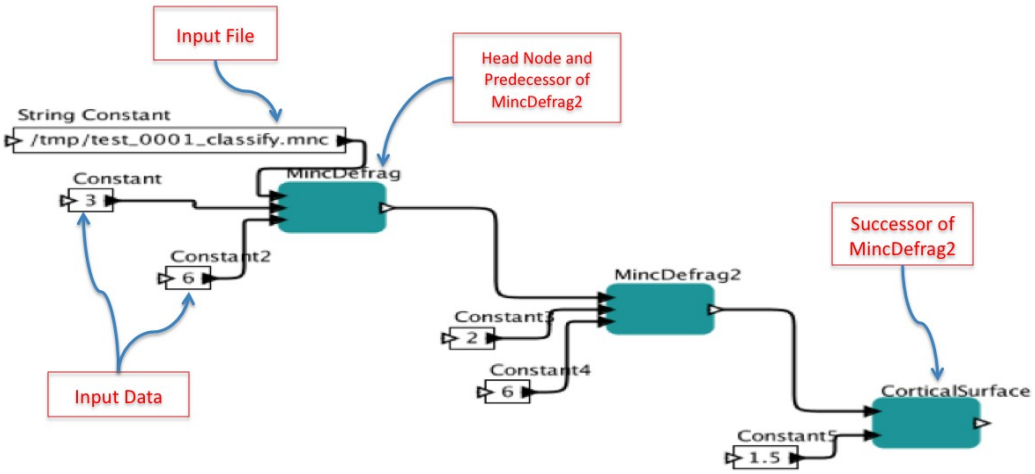

Figure 51: A sample pipeline specification

A list of some significant API functions of the recording interface is given in Appendix E.

8.6.2 The Query Interface

The query interface will enable users to retrieve provenance information, once an analysis is completed and logged into the provenance database. The main objectives of query interface are to:

- Enable a user to retrieve information from provenance database in SQL-like query format.
- Retrieve a complete workflow description for re-executing the workflow.
- Retrieve execution results of a specific version of workflow.
- Enable a user to retrieve a specific part of a workflow along with its associated data and files.
- Validate a workflow or part of it.

A list of some significant API functions of query interface is given in Appendix E.

8.7 Future Work

Current provenance systems commonly capture records of past workflow executions and their related behaviours. Users are then provided with a set of tools that allow them to query and analyse the data that has been captured. This means that provenance information cannot easily be used to fine-tune or refine the process of workflow specification. Clearly this has several significant drawbacks, which include the repetition of common mistakes and an inability to optimise workflow execution. Given the range and sheer size of the data that can be collected through a provenance system, it is difficult for users to make sense of what it all means for their individual research. In recent years techniques for data mining and integration have developed at a rapid pace. With this in mind, a direction for future work is to explore how such techniques can be applied within a provenance service. An intelligent feedback system, which is based on the provenance information, may enable a user to specify workflow components for their optimized execution during the specification phase.

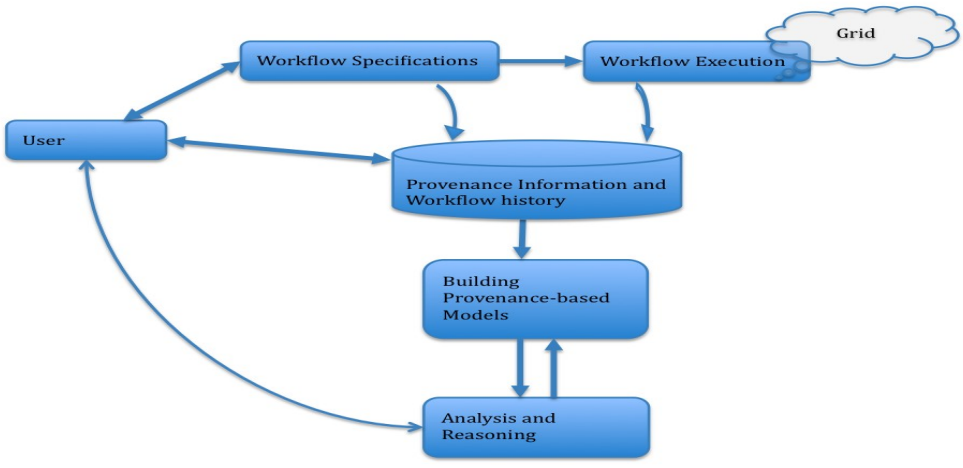

Figure 52: Modelling Provenance data and providing a Feedback mechanism

Wide-ranging provenance information, both at specification and execution levels, can help in building an effective suggestion/feedback based provenance system. Such a system can be built by logging pre and post workflow execution events and then applying different analysis techniques on the provenance information. These techniques can then help in building different execution models of past workflow executions. Different machine learning approaches can be used to build and identify the best execution models. These models will evolve with each iteration of workflow execution and provide feedback to the specification

part of WfMS. Hence, a model-driven provenance system can facilitate a user to specify workflows for their optimized execution. Figure 52 shows the process of modelling provenance information. Analysis and reasoning are applied on these models to provide effective feedback to the user. The need of an intelligent feedback mechanism is of paramount importance when the workflows are executed over the distributed/Grid resources. As users face a few performance and QoS issues when executing workflows in local environments where performance parameters are normally under control. Moreover because of the dynamic nature of Grid there may happen a number of transformations during the execution of a workflow. These all cause a control-less execution of WFs over the Grid. Thus a suggestion based process, built from the history data, is essential for the refinement of a workflow at the time of specification.

8.8 Conclusion

The requirements analysis process has clearly identified the fact that keeping track of how results are produced is important to users. In response to these demands, the Provenance service will provide a means of capturing and maintaining workflow specification and execution information in a workflow/provenance database. Existing provenance systems aim to provide a documented history of distributed workflow executions. These systems use different techniques for capturing and storing provenance information, depending on the nature of the WfMS that is deployed i.e. task-based or service-based workflows. The neuGrid project requires analysis to be performed by joining and linking together different scientific tasks/scripts, which form task-oriented workflows. Moreover workflows will be planned and gridified prior to their execution over distributed resources. This implies that the provenance service will be capable of storing each task information into the provenance database, along with pipeline creation steps. Each process in planning and gridification, by the pipeline service, will also be logged. The provenance database will also link post execution results of individual tasks with its information in a relational DB model. This data will be retrievable and query-able in SQL-like format, to perform customized search operations.

Provenance service will provide recording and querying interfaces to store pipeline information and retrieve already stored provenance information. These interfaces will be published to user communities, following SOA design principles. Therefore a set of APIs will be exposed, making the service functionality available online. The provenance service APIs will also allow a user to retrieve a complete workflow or parts of it from the provenance database. This information will be viewable in the workflow authoring environment, allowing a user to reconstruct a past analysis. This process will help in validating an analysis against a reference dataset and re-execute a pipeline or parts of it. The provenance service will also facilitate other WP6 services because of its SOA-based design. This is also inline with the neuGrid project requirements, which require all services to communicate with each other through their standardized interfaces. Besides aforementioned service features, workflow provenance has many potential research aspects. One such research area is the classification of provenance data in different categories and modelling provenance information using WF specification and execution details. Moreover finding the best specification and execution models and providing user a feedback with such information will enable him to specify workflows for their optimized execution. These research areas will be explored if time and resources permit after the completion of provenance service.

8.9 References

[40] Ilkay Altintas et al., "A Framework for the Design and Reuse of Grid Workflows" Intl. Workshop on Scientific Applications on Grid Computing (SAG'04), LNCS 3458, Springer, 2005

[41] I. Taylor, et al., "The Triana Workflow Environment: Architecture and Applications", Workflows for e-Science, pages 320-339. Springer, New York, Secaucus, NJ, USA, 2007

[42] Thomas Oinn et al., "Taverna: a tool for the composition and enactment of bioinformatics workflow", Bioinformatics Vol 20(17) 2004, Pages 3045-3054

[43] Deelman, E., Singh, G., Su, M., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, B., Good, J. Laiety, A., Jacob J., Katz, D., Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. In Distributed Systems Scientific Programming Journal, Vol. 13(3), 2005.

[44] J Kim, E Deelman, Y Gil, G Mehta, V Ratnakar, "Provenance trails in the Wings/Pegasus system", Concurrency and Computatio: Practice & Experience Vol 20(5) 2008, Pages 487-497

[45] Gil, Y., Ratnakar, V., Deelman, E., Mehta, G., Kim, J., Wings for Pegasus: A Semantic Approach to Creating Very Large Scientific Workflows. In The Eighteenth Conference on Innovative Applications of Artificial Intel ligence, Vancouver, BC, July 2007.

[46] M Szomszor, L Moreau, "Recording and Reasoning over Data Provenance in Web and Grid Services", Lecture Notes in Computer Science 2003, ISBN 978-3-540-20498-5.

[47] A Křenek, J Sitera, L Matyska, F Dvořák, M Mulač, "gLite Job Provenance–a job-centric view", Concurrency and Computation: Practice & Experience Vol 20(5) 2008, Pages 453-462

[48] L Moreau, B Ludascher, I Altintas, R Barga, "The First Provenance Challenge", Concurrency and Computation: Practice & Experience Vol 20(5) 2008, Pages 409-418

[49] Jun Zhao, Carole Goble, Robert Stevens and Daniele Turi, "Mining Taverna's semantic web of provenance", Concurrency and Computation: Practice and Experience 2007.

[50] Martin S, Hohman MM, Liefeld T. The impact of life science identifier on informatics data. *Drug Discovery Today* 2005; 10(22):1566 – 1572.

[51] Zhao J, Wroe C, Goble C, Stevens R, Quan D, Greenwood M. Using semantic web technologies for representing e-science provenance. *Proceedings of the 3rd International Semantic Web Conference*, Hiroshima, Japan, 2004; 92 – 106.

[52] http://www.bic.mni.mcgill.ca/software/