Published in final edited form as:

Neuroimage. 2008 August 1; 42(1): 178–195. doi:10.1016/j.neuroimage.2008.04.186.

Provenance in Neuroimaging

Allan J. MacKenzie-Graham, John D. Van Horn, Roger P. Woods, Karen L. Crawford, and Arthur W. Toga

Laboratory of Neuro Imaging (LONI), Department of Neurology, University of California Los Angeles School of Medicine, 635 Charles E. Young Drive South, Suite 225, Los Angeles, CA 90095-7334.

Phone: (310) 206-2101, Fax: (310) 206-5518, E-mail: toga@loni.ucla.edu

Abstract

Provenance, the description of the history of a set of data, has grown more important with the proliferation of research consortia-related efforts in neuroimaging. Knowledge about the origin and history of an image is crucial for establishing data and results quality; detailed information about how it was processed, including the specific software routines and operating systems that were used, is necessary for proper interpretation, high fidelity replication and re-use. We have drafted a mechanism for describing provenance in a simple and easy to use environment, alleviating the burden of documentation from the user while still providing a rich description of an image's provenance. This combination of ease of use and highly descriptive metadata should greatly facilitate the collection of provenance and subsequent sharing of data.

Introduction

Neuroimaging is a crucial tool for both research and clinical neuroscience. A significant challenge in neuroimaging, and in fact all biological sciences, concerns devising ways to manage the enormous amounts of data generated using current techniques. This challenge is compounded by expansion of collaborative efforts in recent years and the necessity of not only sharing data across multiple sites, but making that data available and useful to the scientific community at large.

The need for solutions that facilitate the process of tool and data exchange has been recognized by the scientific community and numerous efforts are underway to achieve this goal (Murphy et al., 2006). To be meaningful, the tools employed and data considered must be adequately described and documented. The metadata describing the origin and subsequent processing of biological images is often referred to as "provenance" (also "lineage" or "pedigree") (Simmhan et al., 2005).

Provenance in neuroimaging has often been discussed, but few solutions have been suggested. Recently, Simon Miles and Luc Moreau (University of Southampton), and Mike Wilde and Ian Foster (University of Chicago/Argonne National Laboratory) proposed a provenance challenge to determine the state of available provenance systems (Moreau et al., 2007). The challenge consisted of collecting provenance information from a simple neuroimaging workflow (Zhao et al., 2005) and documenting each system's ability to respond to a set of

Correspondence to: Arthur W. Toga.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

predefined queries. Some of these existing provenance systems have previously been proposed as mechanisms for capturing provenance in neuroimaging, though they have not been widely adopted (Zhao et al., 2006). The main difficulty appears to be the need of a system to capture provenance information accurately, completely, and with minimal user intervention. Minimizing the burden on the provenance compiler, a comprehensive system will dramatically improve compliance and free the user to focus on performing neuroimaging research rather than collecting provenance.

Provenance can be used for determining data quality, for interpretation, and for interoperability (Simmhan et al., 2005; Zhao et al., 2004). In the biological sciences, provenance about how data was obtained is crucial for assessing the quality and usefulness of information, as well as enabling data analysis in an appropriate context. It is therefore imperative that the provenance of biological images be easily captured and readily accessible. In multiple sclerosis research, for example, increasingly complex analysis workflows are being developed to extract information from large cross-sectional or longitudinal studies (Liu et al., 2005). This is true of Alzheimer's disease (Fleisher et al., 2005; Mueller et al., 2005; Rusinek et al., 2003), autism (Langen et al., 2007), depression (Drevets, 2000), schizophrenia (Narr et al., 2007), and even studies of normal populations (Mazziotta et al., 1995). The implementation of the complex workflows associated with these studies requires the establishment of quality-control practices to ensure the accuracy, reproducibility, and reusability of the results. In effect, provenance.

Provenance can be divided into two subtypes, data provenance and processing provenance. Data provenance is the metadata that describes the subject, how an image of that subject was collected, who acquired the image, what instrument was used, what settings were used, and how the sample was prepared. However, most scientific image data is not obtained directly from such measurements, but rather derived from other data by the application of computational processes. Processing provenance is the metadata that defines what processing an image has undergone; for example, how the image was skull-stripped, what form of inhomogeneity correction was used, how it was aligned to a standard space, etc. Even data that is presented as "raw" often has been subjected to reconstruction software or converted from the scanner's native image format to a more commonly used and easily shared file format (Van Horn et al., 2004). A complete data provenance model would capture all this information, making the history of a data product transparent, enabling the free sharing of data across the neuroimaging community.

Some data provenance is typically captured at the site where the data is collected, in the headers of image files or in databases that record image acquisition (Erberich et al., 2007; Martone et al., 2003). An abbreviated form of this kind of provenance is often reported in method descriptions or even in the image files themselves (Bidgood et al., 1997). However, this data is seldom propagated with the images, since it is commonly removed or ignored in the course of file conversion for further processing.

Processing provenance can be collected about any resource in the data processing system and may include multiple levels of detail. Two major models for collecting processing provenance have been described, a process-oriented model (Zhao et al., 2004) and a data-oriented model (Simmhan et al., 2005). The process-oriented model collects lineage information from the deriving processes and provenance is inferred from the processing and by inspection of the input and output data. This mechanism is well suited for situations where individual data products are tracked within comprehensive frameworks and where the deriving processes can easily be reapplied to the original data to reproduce the data product. In the data-oriented model, lineage information is explicitly gathered about the set of data. This method is better suited for situations where data sharing occurs across heterogeneous environments and intermediate data

products may not be available for reproduction. This would be the case, for example, when data is shared between two laboratories.

The analysis of raw data in neuroimaging has become a computationally rich process with many intricate steps run on increasingly larger datasets (Liu et al., 2005). Many software packages exist that provide either complete analyses or specific steps in an analysis. These packages often have diverse input and output requirements, utilize different file formats, run in particular environments, and have limited abilities for certain types of data. The combination of these packages to achieve more sensitive and accurate results has become a common tactic in brain mapping studies, but requires much work to ensure valid interoperability between programs. To address this issue we have developed an XML schema (XSD) to guide the production and validation of XML files to capture data and processing provenance and incorporate it into a simple and easy to use environment. In this report we describe this XSD and a simple tool for documenting data provenance and demonstrate that minor differences in compilation can lead to measurable differences in results; reinforcing the need for comprehensive provenance collection. The details and assessment of this provenance schema are also presented.

Methods

LONI Opteron Grid

The LONI Opteron grid is a 306-node, dual-processor V20z cluster (Sun Microsystems Inc., Santa Clara, CA). Each compute node has dual 64-bit 2.4 gigahertz Opteron processors (Advanced Micro Devices Inc., Sunnyvale, CA) with four gigabytes of memory. The LONI grid runs the N1 Grid Engine 6.0u6 (Sun Microsystems Inc.) on top of the Solaris operating system 5.10 (Sun Microsystems Inc.) Applications were compiled with either Sun Studio 11 or Gnu Compiler Collection 3.4.3 (Gcc; Free Software Foundation; <http://gcc.gnu.org>) using no additional options.

Mac OS computers

The iMac is a 2 gigahertz PowerPC G5 processor computer with two gigabytes of memory (Apple, Cupertino, CA). The iMac runs Mac OS X 10.5.1 and Darwin Kernel Version 9.10.0 (Apple).

The MacBook Pro is a 1.83 gigahertz Intel Core Duo processor computer with two gigabytes of memory (Apple, Cupertino, CA). The MacBook Pro runs Mac OS X 10.5.1 and Darwin Kernel Version 9.10.0 (Apple).

Executables

The Automated Image Registration 5.2.5 source code was downloaded from the LONI website (http://www.loni.ucla.edu/Software/Software_Detail.jsp?software_id=8) and compiled according to the author's recommendations (<http://bishopw.loni.ucla.edu/AIR5/config.html>).

The Brain Imaging Software Toolbox source code was downloaded from the McConnell Brain Imaging Centre website (<http://packages.bic.mni.mcgill.ca>) and compiled according to the author's recommendations (<http://www.bic.mni.mcgill.ca/software/distribution>).

The Functional Magnetic Resonance Imaging of the Brain (fMRIB) Software Library 4.0 Mac OS universal binaries were downloaded from the fMRIB website (<http://www.fmrib.ox.ac.uk/fsl/downloads>) and installed according to the developer's recommendations (<http://www.fmrib.ox.ac.uk/fsl/fsl/macosx.html>).

All elements of the provenance schema, the Provenance Editor, and the LONI Pipeline can be downloaded from the LONI software website (<http://www.loni.ucla.edu/Software>).

Language

To begin the discussion we have defined some terms in order to prevent any ambiguity when software is discussed. To facilitate the current discussion we define the following terms; a binary is a pre-compiled program that is ready to run under a given operating system, a script is a simple program written in a utility language that is interpreted at runtime, and an executable is either a binary or script.

XSDs

An XML schema document (XSD) describes the structure of an XML document. An XSD defines the elements and attributes that can appear in a document and defines their order and data types. XSDs are an XML-based alternative to Document Type Definitions (DTD), but are extensible, support data types and namespaces. The flexibility and extensibility of XSDs make them ideally suited to define provenance description XML documents.

Data Provenance

An important aspect of provenance is the description of the subject. Subject provenance includes birth and death dates (for post-mortem studies), in addition to the age of the subject at the time of the data collection (or death). Sex and species are captured, further qualified by strain and genetic manipulation in the case of non-human subjects. Treatments, such as disease induction in experimental models, drug treatment, and combinations of treatments can be documented in the schema. Subject name has explicitly been excluded in order to protect patient privacy (<http://www.hhs.gov/ocr/index.html>), SubjectID standing in as a unique identifier for a given subject. These elements are extensible, allowing for multiple treatments or clinical evaluations. Subject provenance has been described in a simple, yet flexible format in order to make it easily accessible to the community with a minimum of work to adapt it for specialized use (Appendix A). However, the subject provenance is defined in its own independent XSD, making it easily modified or even replaced with a definition that is better suited to a specific kind of study, such as the XCEDE data exchange schema (Keator et al., 2006) for functional MRI (fMRI) studies.

The description of how a set of data was acquired is of critical importance for data provenance. Crucial elements of the acquisition provenance are captured by extracting that information from the image header or by requesting information from the user. Different information is required from the user based on the kind of data acquired. For example, when collecting acquisition provenance about an MRI image, information about the acquisition type (2D vs. 3D), weighting (proton density, T1, T2, etc.), pulse sequence, flip angle, echo time (TE), repetition time (TR), inversion time (TI), matrix dimensions, step sizes, magnet field strength, coil used, equipment manufacturer and model are explicitly captured in the XSD. These elements are far from exhaustive, but are easily expanded and/or extended to accommodate other imaging modalities from diffusion tensor imaging (DTI) to positron emission tomography (PET) (Appendix A).

LONI Provenance Editor

The LONI Provenance Editor is a self-contained, platform-independent application that automatically extracts the provenance information from an image header (such as a DICOM image) and generates a data provenance XML file with that information. The Provenance Editor (http://www.loni.ucla.edu/Software/Software_Detail.jsp?software_id=57) also allows the user to edit data prior to saving the provenance file, correcting inaccuracies or adding additional

information (Fig. 1). This provenance information is stored in .prov files, an XML formatted file that contains both the data and processing provenance and follows the XSD definition (http://www.loni.ucla.edu/Software/Software_Detail.jsp?software_id=58) (Table 1). These files must be structured so that the description of the project, subject, and acquisition come before the processing provenance. Sequential processing steps appear sequentially in the file.

Executable Provenance

The approach we used to document provenance combines both the data-oriented model and the process-oriented model. In the process-oriented model, binary provenance describes how a piece of software was compiled. It is comprised of two parts, a description of the environment and a description of the binary itself. The environment description includes the operating system, environment variables, compiler used, and libraries installed. The binary description includes configuration flags and/or modifications made to configuration or makefiles. Our goal is to provide the user with the ability to reproduce not only the binary, but the environment in which it was run.

A fundamental difference between executables is the hardware platform on which they were compiled. Differences in floating-point performance across different architectures can have a profound impact on outcome of a calculation and have been widely publicized in the popular media (Halfhill, 1995). The XSD captures not only architecture, but also the specific processor and the flags that are enabled on it.

Capturing pertinent details about the operating system is complicated, especially for Linux distributions, since each distribution contains many individually updated components. Essential information must be captured such as the operating system name, version, distribution, kernel name, and kernel version. For example, an application running on Ubuntu Dapper Drake (<http://www.ubuntu.com>) must have the following operating system metadata: Linux, 6.06, Ubuntu Desktop, #1 PREEMPT, 2.6.15-27-386; whereas an application built on the Mac OS X Leopard platform must have the following operating system metadata: Mac OS, 10.5.1, n/a, Darwin, 9.10.0.

The compiler used and libraries linked during compilation are a crucial aspect of the environment. In addition to compiler name and version, a list of which updates have been applied is also captured. This section of the provenance metadata also records which flags were used when the compiler was invoked, architecture and optimization flags being of special interest. Libraries used for compilation are described similarly to the binary itself and are recursive. That is to say that a library that is in turn linked to other libraries are also captured in the library's provenance.

Binaries also can be configured prior to compilation. Some packages are distributed in a format for use with the GNU build system or Autotools (Vaughan, 2000). Modification of the configure script or the makefile can yield substantially different results after compilation. The provenance XSD captures flags to the configure script, modifications to configure scripts and makefiles.

The concept of provenance can extend to knowledge of the behavior of executables, such as describing their function. The Brain Surface Extractor (BSE) (Shattuck and Leahy, 2002), the Brain Extraction Tool (BET) (Smith, 2002), and MRI Watershed (Dale et al., 1999) are all brain extraction algorithms, however, their internal functions may not be evident to a naive user, especially since they are commonly referred to by their abbreviations. This information, in addition to a short description of the executable, is also captured in the XSD and may be added to provenance XML files (Appendix B).

Executable provenance need only be collected once, when a binary is compiled or when a script is written. It must then be collected and recorded manually, then appended to the provenance XML. The LONI Pipeline is currently being extended to store and display executable provenance, eliminating the need for manual file editing in the future.

Workflow Provenance

Binary provenance describes the creation of a binary, but workflow provenance describes the actual invocation of that individual binary or the invocation of a binary in the context of a series of steps or a workflow. In its simplest form, the workflow provenance XSD captures the names of the data files used as input and output, the options used to invoke the single binary, and the environment in which the binary was run. Arguments to the binary are captured by recording the command-line that was used to invoke the binary. The processing environment is described similarly to the environment for compilation, but also includes environmental variables that may modify the behavior of the binary. For example, the FSL tools (Smith et al., 2004) use an environment variable called “FSLOUTPUTTYPE” to define the file format of the output image.

Often image processing is complex and non-linear and cannot be represented in a simple script or directed acyclic graph. Data may converge along several lines of processing only to diverge again after a common step. These complex workflows are difficult to document, either for publication or later re-use. Capturing the provenance for these workflows is equally complex, not only requiring the execution order of the individual steps, but how those steps are related to one another, especially in the case of multiple lines of data being processed simultaneously. In order to address this issue we have used the LONI Pipeline Processing Environment (LONI Pipeline; <http://pipeline.loni.ucla.edu>) (Rex et al., 2003) to capture workflow provenance and description.

The LONI Pipeline is a simple, efficient, and distributed computing solution to these problems, enabling software inclusion from different laboratories in different environments. It provides a visual programming interface for the design, execution, and dissemination of neuroimaging analyses. Individual executables are represented as “modules” that can be added, deleted, and substituted for other modules within a graphical user interface. Connections between the modules that establish an analysis methodology are represented by a “workflow”. The environment handles bookkeeping, controls the details of the computation, and information transfer between modules and within the workflow. It allows files, intermediate results, and other information to be accurately passed between individually connected modules. Modules and workflows can be saved to disk at any stage of development and recalled at a later time for modification, use, or distribution.

Individual modules are representations of programs that are present on a LONI Pipeline server (which can also be the client). The modules are XML descriptions of the executables and how they should be invoked from a command-line interface. The module contains text descriptions of all the arguments for the program and a description of the program itself (Fig. 2). Using the LONI Pipeline as an example of workflow software, we have designed the provenance framework to take advantage of context information that can only be kept while using workflow software. Specifically, the use of conditionals between executables, and loops can all be represented in a higher workflow language and associated with a series of executable events in the provenance. More generally, we want to be able to track how data is derived with sufficient precision that one can create or recreate it from this knowledge. Workflow provenance can then be added to the provenance XML file by copying the entire workflow into the file (Appendix B).

Processing Provenance

By documenting the executable provenance of each module and the workflow itself as workflow provenance, any workflow application can become a mechanism for capturing processing provenance. We have used a combination of an executable provenance XSD with LONI Pipeline workflows to capture processing provenance and description. One of the real strengths of this system is the capacity to easily recreate the processing applied to a file by viewing its provenance file, extracting the workflow, and rerunning it in the LONI Pipeline.

Results

In order to document the utility of this model of provenance documentation, we performed three tests; one to demonstrate that differences in processor architecture can affect results, one to demonstrate that even a small change in the compilation of a binary can alter its results, and one to demonstrate the capacity to independently recreate a workflow and its output data using only the provenance documentation.

For the first test, a workflow (workflow A) was created in the LONI Pipeline from Mac OS universal binaries downloaded from the FMRIB website and installed according to the developer's recommendations on an Apple PowerPC iMac computer. A second workflow (workflow B) was created in the LONI Pipeline from Mac OS universal binaries downloaded from the FMRIB website and installed according to the developer's recommendations on an Apple Core Duo MacBook Pro computer. Both computers were running Mac OS 10.5.1 (Darwin Kernel 9.1.0). With the exception of the difference in hardware platform, all elements of module creation and binary compilation were the same as those used for workflow A. In fact, both computers used the same downloaded binaries (albeit universal).

These first test workflows were independent components analysis (ICA) workflows (Fig. 3). The workflow first aligns a set of images to the first element in that set by calculating a full-affine transformation with FMRIB's Linear Image Registration Tool (FLIRT 5.4.2)(Jenkinson et al., 2002; Jenkinson and Smith, 2001), merging the files with FSLmerge (FSL 4.0) (Smith et al., 2004), and then applying an independent components analysis using Multivariate Exploratory Linear Optimised Decomposition into Independent Components (MELODIC 3.05) (Beckmann and Smith, 2004). Workflow A was run on the PowerPC iMac computer, workflow B was run on the Core Duo MacBook Pro, and the results were subtracted from one another. In both cases the input data was identical. Not surprisingly, the result of the difference of the first independent component image created with the first set of executables and the corresponding image created with the second set of executables was non-zero (Fig. 4). This difference in resulting images was due strictly to the change in architecture.

We selected an analysis that made extensive use of floating-point operations, since differences in floating-point math performance have been anecdotally observed, but not reported in the literature. The difference in the values of the independent component images were on the same order as the actual images themselves, making knowledge of the architecture that the results were computed on crucial. Exact architecture is seldom reported, and in the case outlined above would result in substantially different interpretations of the data. In order for these analyses to be comparable, they would have to be performed on a machine with the same architecture.

Automated Image Registration (AIR) has a number of configuration options that are set a compile-time. The value of the configuration option "AIR_CONFIG_THRESHOLD2" controls the default pixel value threshold used by the registration programs in AIR and are set in a configuration file prior to compilation. The default values can be optionally overridden on the command line at execution time. For the second test, a workflow (workflow C) was created in the LONI Pipeline from binaries compiled on the LONI Opteron Grid by the LONI system

administrator and modules in the pipeline library constructed by the LONI Pipeline developers. In this case, the default value of 7000 for “AIR_CONFIG_THRESHOLD2” was used in compilation. Data and processing provenance was captured and recorded in a provenance file using the mechanism described above.

Another workflow (workflow D) was constructed from modules for executables that were compiled on that machine using the provenance information captured by the mechanism we described above as a guide for compilation. However, for this workflow, the value of “AIR_CONFIG_THRESHOLD2” was set to 1 for compilation. With the exception of this change in one configuration option, all elements of module creation and binary compilation were the same as those used for workflow C.

These workflows were non-linear alignments (Fig. 5). The workflows align two files by calculating a full-affine transformation with Alignlinear (AIR 5.2.5) (Woods et al., 1998a; Woods et al., 1998b), using this linear alignment as the starting point for a 5th order non-linear warp with Align_warp (AIR 5.2.5), and then applying the resulting transformation to the reslice image with Reslice_warp (AIR 5.2.5) using 3D windowed sinc interpolation. The default threshold values were not overridden in either workflow. Both workflows were then run on the LONI grid and the results were subtracted from one another. In all cases the input data was identical. Not surprisingly, the result of the difference of the image created with the first set of executables and the image created with the second set of executables was non-zero (Fig. 6). This difference in resulting images was due strictly to this change in a single, but important configuration option. Minor changes in commonly modified configuration options are rarely reported, though in this case it clearly affected the final results.

In the third test, a workflow (workflow E) was created in the LONI Pipeline from binaries compiled on the LONI Opteron grid by the LONI system administrator and modules in the pipeline library constructed by the LONI Pipeline developers. Data and processing provenance was captured and recorded in provenance file using only the mechanism described above.

Another workflow (workflow F) was constructed from scratch with modules defined by the authors for a second set of executables compiled for the LONI grid, also compiled by the authors. These executables were compiled using only the provenance information captured by the mechanism described above as a guide for compilation.

The workflows used for the third test were also non-linear alignment workflows (Fig. 5). In this case, both sets of executables were compiled with the same options. Both workflows were then run on the LONI grid and the results were subtracted from one another. The same input data was used for both workflows. We compared the aligned images resulting from each workflow by subtracting them from one another and verified that the difference between any combination of two of the three images was 0 at every voxel (data not shown).

Workflows are not limited to single image processing packages. Packages can be combined and provenance information can be captured from those workflows. Combining package elements allows the user the greatest flexibility for their analyses. For example, a workflow could correct motion artifact using tools from Freesurfer (Dale et al., 1999), perform skull stripping using the BSE (Shattuck and Leahy, 2002), calculate and apply the N3 field inhomogeneity correction (Sled et al., 1998), and then align a magnetic resonance image to the ICBM 452 atlas (Rex et al., 2003) using FMRIB’s Linear Image Registration Tool (FLIRT) (Jenkinson et al., 2002; Jenkinson and Smith, 2001) from the FMRIB Software Library (FSL) (Smith et al., 2004) (Fig. 7).

Discussion

Recent interest has arisen in the field of neuroscience, and particularly in neuroimaging, in identifying or creating standards to facilitate software tool interoperability. The NIMH Neuroimaging Informatics Technology Initiative (NIFTI; <http://nifti.nimh.nih.gov>) was formed to aid in the development and enhancement of informatics tools for neuroimaging. Though best known for the Data Format Working Group (DFWG) that has defined the NifTI image file format standard, this effort has recently turned its attentions to how provenance metadata might be standardized. The Biomedical Informatics Research Network (BIRN; <http://nbirn.net>) is another high profile effort working to develop standards among its consortia membership, including the development of study data provenance.

Descriptions of data provenance have been used successfully in other fields of endeavor. For example, the Dublin Core Metadata Initiative (DCMI) is an organization dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing resources that enable more intelligent information discovery systems (<http://dublincore.org>). This also includes meta-data related to workflow provenance. Also, the “Minimum Information About a Microarray Experiment” (MIAME) standard describes the minimum information that is needed to enable the interpretation of the microarray results of the experiment unambiguously and potentially to reproduce the experiment. This standard includes essential experimental and data processing protocols (http://www.mged.org/Workgroups/MIAME/miame_2.0.html). Efforts such as these examples have sought to capture data and workflow information sufficient to reproduce reported study findings and that enable cross-study comparison. Specific workflow description frameworks also exist in other fields that help to sequence data processing steps and that can be used to populate provenance descriptions. These frameworks are highly sophisticated tools that require substantial investment to learn and deploy. They are generally divided in to either data- or process-oriented approaches (Table 2).

The Collaboratory for Multi-scale Chemical Science (CMCS) project is a data-oriented informatics toolkit for collaboration and data management for multi-scale chemistry (Myers et al., 2005). CMCS collects pedigree information about individual data objects by defining input and output data and capturing “pedigree chains” describing the processing that the data has undergone (<http://cmcs.org>). The provenance data is explicitly defined in associations, placing the burden of documentation upon the user.

Other data workflow systems, such as the virtual data system (formerly known as Chimera and incorporating Pegasus) (Zhao et al., 2006), are process-oriented provenance models. The Virtual Data System (VDS) provides middleware for the GriPhyN project (<http://www.griphyn.org>), expressing, executing, and tracking the results of workflows. Provenance is used for regeneration of derived data, comparison of data, and auditing data derivations. Users construct workflows using a standard virtual data language (VDL) describing “transformations” (executable programs) that are executed by a VDL interpreter producing a “derivation” (the execution of a transformation). “Data objects” are entities that are consumed or produced by a derivation. In the VDS model, provenance is inferred from the processing by inverting the processing to associate the output data with the input data. This approach places very little burden on the user to document data provenance.

The myGrid project (Oinn et al., 2004) provides middleware in support of computational experiments in the biological sciences, modeled as workflows in a grid environment. Users construct workflows written in XScufl language using the Taverna engine. The LogBook is a plugin for Taverna engine that allows users to log their experiments in a MySQL database and browse, rerun, and maintain previously run workflows

(<http://www.mygrid.org.uk/wiki/Mygrid/LogBook>). This provenance log contains the executables invoked, the parameters used, data used and derived, and is automatically produced when the workflow executes. This process-oriented provenance log is also inverted to infer the provenance for the intermediate and final set of data.

Within the neuroimaging community, the XCEDE (XML-based Clinical Experiment Data Exchange) schema (Keator et al., 2006) also provides for the storage of data provenance information. Provenance information manually captured includes hardware, compilation and libraries linked, operating system and software versions, and parameters used to generate and document results. XCEDE is a data-oriented system where the provenance metadata is associated with the actual data files.

However, the tools described above do not provide a simple mechanism for the capture of provenance metadata from multiple packages, the capacity to represent complex, non sequential analyses, nor at a sufficient level of detail to allow the reproduction of a derived set of data on a new platform. Hence the need for the development of a provenance framework that can easily be applied to complex neuroimaging analyses.

The eXtensible Neuroimaging Archive Toolkit (XNAT) (Marcus et al., 2007) is a data management system for the collection of data from multiple sources, storage of the data in a repository, and to facilitate access to the data by authorized users. It is a software platform that comprises a data archive, a user interface, and the middleware to support both. As such, XNAT is not a provenance system, but a data archiving system that makes use of an ArcBuild script to preprocess data and writes a provenance record to the database. Data must be processed within ArcBuild within the XNAT system in order for provenance metadata to be recorded and captured. It is possible that the provenance schema we have described could be incorporated into the XNAT archiving system, permitting users to process or submit their previously processed data from outside of the ArcBuild/XNAT system.

By maintaining software provenance, it will be possible to build intelligent systems that guide users in the design of analytic strategies. These tools can capture the expertise of algorithm developers, as well as the experience of experts at local institutions who have spent significant periods of time learning how best to apply specific tools to the analysis needs of the laboratory. The tools will inform the users of missing processing stages, suggest available and verified processing modules, and warn of incompatible data types.

The development of the provenance framework described here will continue to be an open discussion for the neuroimaging community to continue its evolution (<http://provenance.loni.ucla.edu>). All software tools and documentation will be available for distribution, discussion, and modification through our website. We encourage the community to become involved in the development of not only the schema, but of software tools that can take advantage of it.

Future directions include the adaptation of existing processing tools to make use of the provenance framework. The LONI Pipeline was developed to facilitate neuroimaging analyses. It is currently being extended to incorporate executable provenance into its modules and to intelligently propagate provenance files with its imaging results. The LONI Pipeline can accommodate almost any form of workflow, the underlying architecture is module agnostic, not limiting the kind of applications that can be run within it. LONI Pipeline workflows can therefore serve to document workflow provenance in almost any field of endeavor.

Workflows can be enrolled in a database, creating a processing and provenance database. Having a readily searchable database of commonly used (and rarely used) workflows would

greatly aid investigators in recreating the conditions of a particular analysis, reproducing previous results and rerunning analyses with small modifications.

In an era where digital information underlies much of the scientific enterprise and the manipulation of that data has become increasingly complex, the recording of data and methods provenance takes on greater importance. In this article, we describe an XML-based neuroimaging provenance description that can be implemented in any workflow environment. We envision the LONI Pipeline as fulfilling a role for neuroimaging similar to other frameworks in chemistry or high energy physics. We believe that data and workflow provenance form a major element of the program that promotes data processing methods description, data sharing, and study replication.

References

- Beckmann CF, Smith SM. Probabilistic independent component analysis for functional magnetic resonance imaging. *IEEE Trans Med Imaging* 2004;23:137–152. [PubMed: 14964560]
- Bidgood WD Jr, Horii SC, Prior FW, Van Syckle DE. Understanding and using DICOM, the data interchange standard for biomedical imaging. *J Am Med Inform Assoc* 1997;4:199–212. [PubMed: 9147339]
- Dale AM, Fischl B, Sereno MI. Cortical surface-based analysis. I. Segmentation and surface reconstruction. *Neuroimage* 1999;9:179–194. [PubMed: 9931268]
- Drevets WC. Neuroimaging studies of mood disorders. *Biol Psychiatry* 2000;48:813–829. [PubMed: 11063977]
- Erberich SG, Silverstein JC, Chervenak A, Schuler R, Nelson MD, Kesselman C. Globus MEDICUS - Federation of DICOM Medical Imaging Devices into Healthcare Grids. *Stud Health Technol Inform* 2007;126:269–278. [PubMed: 17476069]
- Fleisher AS, Houston WS, Eyler LT, Frye S, Jenkins C, Thal LJ, Bondi MW. Identification of Alzheimer disease risk by functional magnetic resonance imaging. *Arch Neurol* 2005;62:1881–1888. [PubMed: 16344346]
- Halfhill TR. The Truth Behind the Pentium Bug. *Byte*. 1995
- Jenkinson M, Bannister P, Brady M, Smith S. Improved optimization for the robust and accurate linear registration and motion correction of brain images. *Neuroimage* 2002;17:825–841. [PubMed: 12377157]
- Jenkinson M, Smith S. A global optimisation method for robust affine registration of brain images. *Med Image Anal* 2001;5:143–156. [PubMed: 11516708]
- Keator DB, Gadde S, Grethe JS, Taylor DV, Potkin SG. A general XML schema and SPM toolbox for storage of neuro-imaging results and anatomical labels. *Neuroinformatics* 2006;4:199–212. [PubMed: 16845169]
- Langen M, Durston S, Staal WG, Palmen SJ, van Engeland H. Caudate nucleus is enlarged in high-functioning medication-naïve subjects with autism. *Biol Psychiatry* 2007;62:262–266. [PubMed: 17224135]
- Liu L, Meier D, Polgar-Turcsanyi M, Karkocha P, Bakshi R, Guttmann CR. Multiple sclerosis medical image analysis and information management. *J Neuroimaging* 2005;15:103S–117S. [PubMed: 16385023]
- Marcus DS, Olsen TR, Ramaratnam M, Buckner RL. The Extensible Neuroimaging Archive Toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data. *Neuroinformatics* 2007;5:11–34. [PubMed: 17426351]
- Martone ME, Zhang S, Gupta A, Qian X, He H, Price DL, Wong M, Santini S, Ellisman MH. The cell-centered database: a database for multiscale structural and protein localization data from light and electron microscopy. *Neuroinformatics* 2003;1:379–395. [PubMed: 15043222]
- Mazziotta JC, Toga AW, Evans A, Fox P, Lancaster J. A probabilistic atlas of the human brain: theory and rationale for its development. The International Consortium for Brain Mapping (ICBM). *Neuroimage* 1995;2:89–101. [PubMed: 9343592]

- Moreau L, Ludäscher B, Altintas I, Barga RS, Bowers S, Callahan S, Chin G, Clifford B, Cohen S, Cohen-Boulakia S, Davidson S, Deelman E, Digiampietri L, Foster I, Freire J, Frew J, Futrelle J, Gibson T, Gil Y, Goble C, Golbeck J, Groth P, Holland DA, Jiang S, Kim J, Koop D, Krenek A, McPhillips T, Mehta G, Miles S, Metzger D, Munroe S, Myers J, Plale B, Podhorszki N, Ratnakar V, Santos E, Scheidegger C, Schuchardt K, Seltzer M, Simmhan YL, Silva C, Slaughter P, Stephan E, Stevens R, Turi D, Vo H, Wilde M, Zhao J, Zhao Y. Special Issue: The First Provenance Challenge. *Concurrency and Computation: Practice & Experience* 00. 2007
- Mueller SG, Weiner MW, Thal LJ, Petersen RC, Jack CR, Jagust W, Trojanowski JQ, Toga AW, Beckett L. Ways toward an early diagnosis in Alzheimer's disease: The Alzheimer's Disease Neuroimaging Initiative (ADNI). *Alzheimers Dement* 2005;1:55–66. [PubMed: 17476317]
- Murphy SN, Mendis ME, Grethe J, Gollub R, Kennedy D, Rosen BR. A Web Portal that Enables Collaborative Use of Advanced Medical Image Processing and Informatics Tools through the Biomedical Informatics Research Network (BIRN). *AMIA Annu Symp Proc* 2006:579–583. [PubMed: 17238407]
- Myers JD, Allison TC, Bittner S, Didier B, Frenklach M, Green WH, Ho YL, Hewson J, Koegler W, Lansing C, Leahy D, Lee M, McCoy R, Minkoff M, Nijsure S, Von Laszewski G, Montoya D, Oluwole L, Pancerella C, Pinzon R, Pitz W, Rahn LA, Ruscic B, Schuchardt K, Stephan E, Wagner A, Windus T, Yang C. A collaborative informatics infrastructure for multi-scale science. *Cluster Computing-the Journal of Networks Software Tools and Applications* 2005;8:243–253.
- Narr KL, Bilder RM, Luders E, Thompson PM, Woods RP, Robinson D, Szeszko PR, Dimtcheva T, Gurbani M, Toga AW. Asymmetries of cortical shape: Effects of handedness, sex and schizophrenia. *Neuroimage* 2007;34:939–948. [PubMed: 17166743]
- Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, Carver T, Glover K, Pocock MR, Wipat A, Li P. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* 2004;20:3045–3054. [PubMed: 15201187]
- Rex DE, Ma JQ, Toga AW. The LONI Pipeline Processing Environment. *Neuroimage* 2003;19:1033–1048. [PubMed: 12880830]
- Rusinek H, De Santi S, Frid D, Tsui WH, Tarshish CY, Convit A, de Leon MJ. Regional brain atrophy rate predicts future cognitive decline: 6-year longitudinal MR imaging study of normal aging. *Radiology* 2003;229:691–696. [PubMed: 14657306]
- Shattuck DW, Leahy RM. BrainSuite: an automated cortical surface identification tool. *Med Image Anal* 2002;6:129–142. [PubMed: 12045000]
- Simmhan YL, Plale B, Gannon D. A survey of data provenance in e-science. *Sigmod Record* 2005;34:31–36.
- Sled JG, Zijdenbos AP, Evans AC. A nonparametric method for automatic correction of intensity nonuniformity in MRI data. *IEEE Trans Med Imaging* 1998;17:87–97. [PubMed: 9617910]
- Smith SM. Fast robust automated brain extraction. *Hum Brain Mapp* 2002;17:143–155. [PubMed: 12391568]
- Smith SM, Jenkinson M, Woolrich MW, Beckmann CF, Behrens TE, Johansen-Berg H, Bannister PR, De Luca M, Drobnjak I, Flitney DE, Niazy RK, Saunders J, Vickers J, Zhang Y, De Stefano N, Brady JM, Matthews PM. Advances in functional and structural MR image analysis and implementation as FSL. *Neuroimage* 2004;23 Suppl 1:S208–219. [PubMed: 15501092]
- Van Horn JD, Grafton ST, Rockmore D, Gazzaniga MS. Sharing neuroimaging studies of human cognition. *Nat Neurosci* 2004;7:473–481. [PubMed: 15114361]
- Vaughan, GV. GNU Autoconf, Automake, and Libtool. Vol. 1. New Riders; Indianapolis: 2000.
- Woods RP, Grafton ST, Holmes CJ, Cherry SR, Mazziotta JC. Automated image registration: I. General methods and intrasubject, intramodality validation. *Journal of Computer Assisted Tomography* 1998a;22:139–152.
- Woods RP, Grafton ST, Watson JD, Sicotte NL, Mazziotta JC. Automated image registration: II. Intersubject validation of linear and nonlinear models. *Journal of Computer Assisted Tomography* 1998b;22:153–165. [PubMed: 9448780]
- Zhao J, Goble C, Stevens R, Bechhofer S. Semantically linking and browsing provenance logs for e-science. *Semantics of a Networked World: Semantics for Grid Databases* 2004;3226:158–176.

Zhao Y, Dobson J, Foster I, Moreau L, Wilde M. A notation and system for expressing and executing cleanly typed workflows on messy scientific data. *Sigmod Record* 2005;34:37–43.

Zhao Y, Wilde M, Foster I. Applying the virtual data provenance model. *Provenance and Annotation of Data* 2006;4145:148–161.

Appendix A: The Neuroimaging Data Provenance XSD

```
<?xml schema xmlns:xsd="http://www.w3.org/2001/XMLSchema#">

<xsd:annotation>
  <xsd:documentation xml:lang="en-US"> Neuroimaging data provenance schema
</xsd:documentation>
</xsd:annotation>

<xsd:element name="NeuroimagingDataProvenance">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Project" type="Project"/>
      <xsd:element name="Subject" type="Subject"/>
      <xsd:element name="AcquisitionProvenance" type="AcquisitionProvenance" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ProcessingProvenance" type="ProcessingProvenance" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="Project">
  <xsd:sequence>
    <xsd:element name="Description" type="xsd:string" />
    <xsd:element name="Funding" type="xsd:string" />
    <xsd:element name="Comment" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="ProjectID" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Subject">
  <xsd:sequence>
    <xsd:element name="Birth" type="xsd:date"/>
    <xsd:element name="Age" type="xsd:string"/>
    <xsd:element name="Death" type="xsd:date"/>
    <xsd:element name="Sex" type="xsd:string"/>
    <xsd:element name="Species" type="xsd:string"/>
    <xsd:element name="Strain" type="xsd:string"/>
    <xsd:element name="GeneticModification" type="xsd:string"/>
    <xsd:element name="Treatment" type="xsd:string"/>
    <xsd:element name="Comment" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="SubjectID" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="AcquisitionProvenance">
  <xsd:sequence>
    <xsd:element name="MRIAcquisition" type="MRIAcquisition" minOccurs="0" maxOccurs="1"/>

```

```

<xsd:element name="PETAcquisition" type="PETAcquisition" minOccurs="0" maxOccurs="1"/>
<xsd:element name="AcquisitionTime" type="xsd:dateTime"/>
<xsd:element name="MatrixX" type="xsd:int" />
<xsd:element name="MatrixY" type="xsd:int" />
<xsd:element name="MatrixZ" type="xsd:int" />
<xsd:element name="PixelSizeX" type="xsd:decimal" />
<xsd:element name="PixelSizeY" type="xsd:decimal" />
<xsd:element name="PixelSizeZ" type="xsd:decimal" />
<xsd:element name="BitDepth" type="xsd:int" />
<xsd:element name="Manufacturer" type="xsd:string"/>
<xsd:element name="Model" type="xsd:string"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MRIAcquisition">
  <xsd:sequence>
    <xsd:element name="AcquisitionType" type="xsd:string"/>
    <xsd:element name="MRATechnique" type="xsd:string" />
    <xsd:element name="Weighting" type="xsd:string" />
    <xsd:element name="PulseSequence" type="xsd:string" />
    <xsd:element name="TE" type="xsd:decimal" />
    <xsd:element name="TR" type="xsd:int" />
    <xsd:element name="TI" type="xsd:int" />
    <xsd:element name="FlipAngle" type="xsd:string" />
    <xsd:element name="AcquisitionPlane" type="xsd:string" />
    <xsd:element name="Coil" type="xsd:string" />
    <xsd:element name="FieldStrength" type="xsd:decimal" />
    <xsd:element name="Comment" type="xsd:string" />
    <xsd:element name="DescriptionDump" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="AcquisitionName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="PETAcquisition">
  <xsd:sequence>
    <xsd:element name="Radiopharmaceutical" type="xsd:string"/>
    <xsd:element name="Radioisotope" type="xsd:string"/>
    <xsd:element name="CountsSource" type="xsd:string"/>
    <xsd:element name="ConvolutionKernel" type="xsd:string"/>
    <xsd:element name="RandomsCorrection" type="xsd:string"/>
    <xsd:element name="AttenuationCorrection" type="xsd:string"/>
    <xsd:element name="DecayCorrection" type="xsd:string"/>
    <xsd:element name="ScatterCorrection" type="xsd:string"/>
    <xsd:element name="Reconstruction" type="xsd:string"/>
    <xsd:element name="Comment" type="xsd:string"/>
    <xsd:element name="DescriptionDump" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="AcquisitionName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ProcessingProvenance">
  <xsd:sequence>

```



```

<xsd:element name="ProcessingTime" type="xsd:dateTime"/>
<xsd:element name="ProcessingDescription" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="ImageDescription" type="ImageDescription" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="ExecutableProvenance" type="ExecutableProvenance" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="PipelineProvenance" type="PipelineProvenance" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="ProcessingProvenance" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ImageDescription">
<xsd:sequence>
<xsd:element name="FileFormat" type="xsd:string"/>
<xsd:element name="FileSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="VoxelDimensionX" type="xsd:int"/>
<xsd:element name="VoxelDimensionY" type="xsd:int"/>
<xsd:element name="VoxelDimensionZ" type="xsd:int"/>
<xsd:element name="StepSizeX" type="xsd:float"/>
<xsd:element name="StepSizeY" type="xsd:float"/>
<xsd:element name="StepSizeZ" type="xsd:float"/>
<xsd:element name="StartX" type="xsd:float"/>
<xsd:element name="StartY" type="xsd:float"/>
<xsd:element name="StartZ" type="xsd:float"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="HeaderDump" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="ImageID" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ExecutableProvenance">
<xsd:sequence>
<xsd:element name="ExecutableDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Invocation" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="AncillaryFiles" type="FileDescription" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Script" type="Script" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="ExecutableName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="PipelineProvenance">
<xsd:sequence>
<xsd:element name="PipelineDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Invocation" type="xsd:string"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="PipelineDump" type="xsd:string"/>
</xsd:sequence>

```

```

    <xsd:attribute name="PipelineName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="FileDescription">
    <xsd:sequence>
      <xsd:element name="FileDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileFormat" type="xsd:string"/>
      <xsd:element name="FormatSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileCreateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="FileName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Script">
    <xsd:sequence>
      <xsd:element name="ScriptDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ScriptVersion" type="xsd:string"/>
      <xsd:element name="ScriptAuthor" type="xsd:string"/>
      <xsd:element name="ScriptCreateTime" type="xsd:dateTime"/>
      <xsd:element name="ScriptModifyAuthor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptModifyTime" type="xsd:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptText" type="xsd:string"/>
      <xsd:element name="ScriptShell" type="xsd:string"/>
      <xsd:element name="ScriptShellVersion" type="xsd:string"/>
      <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ScriptName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Binary">
    <xsd:sequence>
      <xsd:element name="BinaryDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="BinaryVersion" type="xsd:string"/>
      <xsd:element name="BinaryPackage" type="Package"/>
      <xsd:element name="CompilationTime" type="xsd:dateTime"/>
      <xsd:element name="SystemConfig" type="SystemConfig"/>
      <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="BinaryName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Package">
    <xsd:sequence>
      <xsd:element name="PackageSource" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PackageName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="SystemConfig">
    <xsd:sequence>

```

```

<xsd:element name="Architecture" type="xsd:string"/>
<xsd:element name="OS" type="OS"/>
<xsd:element name="Compiler" type="Compiler"/>
<xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>

```

```

<xsd:element name="PETAcquisition" type="PETAcquisition" minOccurs="0" maxOccurs="1"/>
<xsd:element name="AcquisitionTime" type="xsd:dateTime"/>
<xsd:element name="MatrixX" type="xsd:int" />
<xsd:element name="MatrixY" type="xsd:int" />
<xsd:element name="MatrixZ" type="xsd:int" />
<xsd:element name="PixelSizeX" type="xsd:decimal" />
<xsd:element name="PixelSizeY" type="xsd:decimal" />
<xsd:element name="PixelSizeZ" type="xsd:decimal" />
<xsd:element name="BitDepth" type="xsd:int" />
<xsd:element name="Manufacturer" type="xsd:string"/>
<xsd:element name="Model" type="xsd:string"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MRIAcquisition">
  <xsd:sequence>
    <xsd:element name="AcquisitionType" type="xsd:string"/>
    <xsd:element name="MRATechnique" type="xsd:string" />
    <xsd:element name="Weighting" type="xsd:string" />
    <xsd:element name="PulseSequence" type="xsd:string" />
    <xsd:element name="TE" type="xsd:decimal" />
    <xsd:element name="TR" type="xsd:int" />
    <xsd:element name="TI" type="xsd:int" />
    <xsd:element name="FlipAngle" type="xsd:string" />
    <xsd:element name="AcquisitionPlane" type="xsd:string" />
    <xsd:element name="Coil" type="xsd:string" />
    <xsd:element name="FieldStrength" type="xsd:decimal" />
    <xsd:element name="Comment" type="xsd:string" />
    <xsd:element name="DescriptionDump" type="xsd:string" />
  </xsd:sequence>
  <xsd:attribute name="AcquisitionName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="PETAcquisition">
  <xsd:sequence>
    <xsd:element name="Radiopharmaceutical" type="xsd:string"/>
    <xsd:element name="Radioisotope" type="xsd:string"/>
    <xsd:element name="CountsSource" type="xsd:string"/>
    <xsd:element name="ConvolutionKernel" type="xsd:string"/>
    <xsd:element name="RandomsCorrection" type="xsd:string"/>
    <xsd:element name="AttenuationCorrection" type="xsd:string"/>
    <xsd:element name="DecayCorrection" type="xsd:string"/>
    <xsd:element name="ScatterCorrection" type="xsd:string"/>
    <xsd:element name="Reconstruction" type="xsd:string"/>
    <xsd:element name="Comment" type="xsd:string"/>
    <xsd:element name="DescriptionDump" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="AcquisitionName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ProcessingProvenance">
  <xsd:sequence>

```

```

<xsd:element name="ProcessingTime" type="xsd:dateTime"/>
<xsd:element name="ProcessingDescription" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="ImageDescription" type="ImageDescription" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="ExecutableProvenance" type="ExecutableProvenance" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="PipelineProvenance" type="PipelineProvenance" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="ProcessingProvenance" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ImageDescription">
<xsd:sequence>
<xsd:element name="FileFormat" type="xsd:string"/>
<xsd:element name="FileSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="VoxelDimensionX" type="xsd:int"/>
<xsd:element name="VoxelDimensionY" type="xsd:int"/>
<xsd:element name="VoxelDimensionZ" type="xsd:int"/>
<xsd:element name="StepSizeX" type="xsd:float"/>
<xsd:element name="StepSizeY" type="xsd:float"/>
<xsd:element name="StepSizeZ" type="xsd:float"/>
<xsd:element name="StartX" type="xsd:float"/>
<xsd:element name="StartY" type="xsd:float"/>
<xsd:element name="StartZ" type="xsd:float"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="HeaderDump" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="ImageID" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ExecutableProvenance">
<xsd:sequence>
<xsd:element name="ExecutableDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Invocation" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="AncillaryFiles" type="FileDescription" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Script" type="Script" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="ExecutableName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="PipelineProvenance">
<xsd:sequence>
<xsd:element name="PipelineDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Invocation" type="xsd:string"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="PipelineDump" type="xsd:string"/>
</xsd:sequence>

```

```

    <xsd:attribute name="PipelineName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="FileDescription">
    <xsd:sequence>
      <xsd:element name="FileDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileFormat" type="xsd:string"/>
      <xsd:element name="FormatSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileCreateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="FileName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Script">
    <xsd:sequence>
      <xsd:element name="ScriptDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ScriptVersion" type="xsd:string"/>
      <xsd:element name="ScriptAuthor" type="xsd:string"/>
      <xsd:element name="ScriptCreateTime" type="xsd:dateTime"/>
      <xsd:element name="ScriptModifyAuthor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptModifyTime" type="xsd:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptText" type="xsd:string"/>
      <xsd:element name="ScriptShell" type="xsd:string"/>
      <xsd:element name="ScriptShellVersion" type="xsd:string"/>
      <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ScriptName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Binary">
    <xsd:sequence>
      <xsd:element name="BinaryDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="BinaryVersion" type="xsd:string"/>
      <xsd:element name="BinaryPackage" type="Package"/>
      <xsd:element name="CompilationTime" type="xsd:dateTime"/>
      <xsd:element name="SystemConfig" type="SystemConfig"/>
      <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="BinaryName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Package">
    <xsd:sequence>
      <xsd:element name="PackageSource" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PackageName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="SystemConfig">
    <xsd:sequence>

```



```

    <xsd:element name="Architecture" type="xsd:string"/>
    <xsd:element name="OS" type="OS"/>
    <xsd:element name="Compiler" type="Compiler"/>
    <xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>
</xsd:schema>

```

```

<xsd:element name="ProcessingTime" type="xsd:dateTime"/>
<xsd:element name="ProcessingDescription" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="ImageDescription" type="ImageDescription" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="ExecutableProvenance" type="ExecutableProvenance" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="PipelineProvenance" type="PipelineProvenance" minOccurs="0"
maxOccurs="1"/>
<xsd:element name="comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="ProcessingProvenance" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ImageDescription">
<xsd:sequence>
<xsd:element name="FileFormat" type="xsd:string"/>
<xsd:element name="FileSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="VoxelDimensionX" type="xsd:int"/>
<xsd:element name="VoxelDimensionY" type="xsd:int"/>
<xsd:element name="VoxelDimensionZ" type="xsd:int"/>
<xsd:element name="StepSizeX" type="xsd:float"/>
<xsd:element name="StepSizeY" type="xsd:float"/>
<xsd:element name="StepSizeZ" type="xsd:float"/>
<xsd:element name="StartX" type="xsd:float"/>
<xsd:element name="StartY" type="xsd:float"/>
<xsd:element name="StartZ" type="xsd:float"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="HeaderDump" type="xsd:string" minOccurs="0"/>
</xsd:sequence>
<xsd:attribute name="ImageID" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ExecutableProvenance">
<xsd:sequence>
<xsd:element name="ExecutableDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Invocation" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="AncillaryFiles" type="FileDescription" minOccurs="0"
maxOccurs="unbounded"/>
<xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Script" type="Script" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="ExecutableName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="PipelineProvenance">
<xsd:sequence>
<xsd:element name="PipelineDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
<xsd:element name="Invocation" type="xsd:string"/>
<xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="PipelineDump" type="xsd:string"/>
</xsd:sequence>

```

```

    <xsd:attribute name="PipelineName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="FileDescription">
    <xsd:sequence>
      <xsd:element name="FileDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileFormat" type="xsd:string"/>
      <xsd:element name="FormatSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileCreateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="FileName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Script">
    <xsd:sequence>
      <xsd:element name="ScriptDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ScriptVersion" type="xsd:string"/>
      <xsd:element name="ScriptAuthor" type="xsd:string"/>
      <xsd:element name="ScriptCreateTime" type="xsd:dateTime"/>
      <xsd:element name="ScriptModifyAuthor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptModifyTime" type="xsd:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptText" type="xsd:string"/>
      <xsd:element name="ScriptShell" type="xsd:string"/>
      <xsd:element name="ScriptShellVersion" type="xsd:string"/>
      <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ScriptName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Binary">
    <xsd:sequence>
      <xsd:element name="BinaryDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="BinaryVersion" type="xsd:string"/>
      <xsd:element name="BinaryPackage" type="Package"/>
      <xsd:element name="CompilationTime" type="xsd:dateTime"/>
      <xsd:element name="SystemConfig" type="SystemConfig"/>
      <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="BinaryName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Package">
    <xsd:sequence>
      <xsd:element name="PackageSource" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PackageName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="SystemConfig">
    <xsd:sequence>

```

```

    <xsd:element name="Architecture" type="xsd:string"/>
    <xsd:element name="OS" type="OS"/>
    <xsd:element name="Compiler" type="Compiler"/>
    <xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>
</xsd:schema>

```

```

    <xsd:attribute name="PipelineName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="FileDescription">
    <xsd:sequence>
      <xsd:element name="FileDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileFormat" type="xsd:string"/>
      <xsd:element name="FormatSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileCreateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="FileName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Script">
    <xsd:sequence>
      <xsd:element name="ScriptDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ScriptVersion" type="xsd:string"/>
      <xsd:element name="ScriptAuthor" type="xsd:string"/>
      <xsd:element name="ScriptCreateTime" type="xsd:dateTime"/>
      <xsd:element name="ScriptModifyAuthor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptModifyTime" type="xsd:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptText" type="xsd:string"/>
      <xsd:element name="ScriptShell" type="xsd:string"/>
      <xsd:element name="ScriptShellVersion" type="xsd:string"/>
      <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ScriptName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Binary">
    <xsd:sequence>
      <xsd:element name="BinaryDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="BinaryVersion" type="xsd:string"/>
      <xsd:element name="BinaryPackage" type="Package"/>
      <xsd:element name="CompilationTime" type="xsd:dateTime"/>
      <xsd:element name="SystemConfig" type="SystemConfig"/>
      <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="BinaryName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Package">
    <xsd:sequence>
      <xsd:element name="PackageSource" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PackageName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="SystemConfig">
    <xsd:sequence>

```

```

<xsd:element name="Architecture" type="xsd:string"/>
<xsd:element name="OS" type="OS"/>
<xsd:element name="Compiler" type="Compiler"/>
<xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>

```



```

    <xsd:element name="Architecture" type="xsd:string"/>
    <xsd:element name="OS" type="OS"/>
    <xsd:element name="Compiler" type="Compiler"/>
    <xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>
</xsd:schema>

```

Appendix B: The Workflow Provenance XSD

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en-US"> Processing provenance schema </xsd:documentation>
  </xsd:annotation>

  <xsd:complexType name="ProcessingProvenance">
    <xsd:sequence>
      <xsd:element name="ProcessingTime" type="xsd:dateTime"/>
      <xsd:element name="ProcessingDescription" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ExecutableProvenance" type="ExecutableProvenance" minOccurs="0"
maxOccurs="1"/>
      <xsd:element name="PipelineProvenance" type="PipelineProvenance" minOccurs="0"
maxOccurs="1"/>
      <xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ProcessingProvenance" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="ExecutableProvenance">
    <xsd:sequence>
      <xsd:element name="ExecutableDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Invocation" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="AncillaryFiles" type="FileDescription" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Script" type="Script" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ExecutableName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="PipelineProvenance">
    <xsd:sequence>
      <xsd:element name="PipelineDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="Invocation" type="xsd:string"/>
      <xsd:element name="Comment" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="PipelineDump" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PipelineName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="FileDescription">
    <xsd:sequence>
      <xsd:element name="FileDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileFormat" type="xsd:string"/>
      <xsd:element name="FormatSubtype" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="FileCreateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="FileName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Script">
    <xsd:sequence>
      <xsd:element name="ScriptDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="ScriptVersion" type="xsd:string"/>
      <xsd:element name="ScriptAuthor" type="xsd:string"/>
      <xsd:element name="ScriptCreateTime" type="xsd:dateTime"/>
      <xsd:element name="ScriptModifyAuthor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptModifyTime" type="xsd:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
      <xsd:element name="ScriptText" type="xsd:string"/>
      <xsd:element name="ScriptShell" type="xsd:string"/>
      <xsd:element name="ScriptShellVersion" type="xsd:string"/>
      <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ScriptName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Binary">
    <xsd:sequence>
      <xsd:element name="BinaryDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="BinaryVersion" type="xsd:string"/>
      <xsd:element name="BinaryPackage" type="Package"/>
      <xsd:element name="CompilationTime" type="xsd:dateTime"/>
      <xsd:element name="SystemConfig" type="SystemConfig"/>
      <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="BinaryName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Package">
    <xsd:sequence>
      <xsd:element name="PackageSource" type="xsd:string"/>
    </xsd:sequence>
    <xsd:attribute name="PackageName" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="SystemConfig">
    <xsd:sequence>
      <xsd:element name="Architecture" type="Architecture"/>
      <xsd:element name="OS" type="OS"/>
      <xsd:element name="Compiler" type="Compiler"/>
      <xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

```

```

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Architecture">
  <xsd:sequence>
    <xsd:element name="Processor" type="Processor"/>
  </xsd:sequence>
  <xsd:attribute name="ArchitectureName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Processor">
  <xsd:sequence>
    <xsd:element name="VendorID" type="xsd:string"/>
    <xsd:element name="ModelName" type="xsd:string"/>
    <xsd:element name="ProcessorFlags" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>

```

```

    <xsd:element name="FileCreateTime" type="xsd:dateTime" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="FileName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Script">
  <xsd:sequence>
    <xsd:element name="ScriptDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="ScriptVersion" type="xsd:string"/>
    <xsd:element name="ScriptAuthor" type="xsd:string"/>
    <xsd:element name="ScriptCreateTime" type="xsd:dateTime"/>
    <xsd:element name="ScriptModifyAuthor" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="ScriptModifyTime" type="xsd:dateTime" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="ScriptText" type="xsd:string"/>
    <xsd:element name="ScriptShell" type="xsd:string"/>
    <xsd:element name="ScriptShellVersion" type="xsd:string"/>
    <xsd:element name="Binary" type="Binary" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="ScriptName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Binary">
  <xsd:sequence>
    <xsd:element name="BinaryDescription" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="BinaryVersion" type="xsd:string"/>
    <xsd:element name="BinaryPackage" type="Package"/>
    <xsd:element name="CompilationTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="BinaryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Package">
  <xsd:sequence>
    <xsd:element name="PackageSource" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="PackageName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="SystemConfig">
  <xsd:sequence>
    <xsd:element name="Architecture" type="Architecture"/>
    <xsd:element name="OS" type="OS"/>
    <xsd:element name="Compiler" type="Compiler"/>
    <xsd:element name="Library" type="Library" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="ConfigurationFlags" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Architecture">
  <xsd:sequence>
    <xsd:element name="Processor" type="Processor"/>
  </xsd:sequence>
  <xsd:attribute name="ArchitectureName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Processor">
  <xsd:sequence>
    <xsd:element name="VendorID" type="xsd:string"/>
    <xsd:element name="ModelName" type="xsd:string"/>
    <xsd:element name="ProcessorFlags" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

```

```

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>

<xsd:complexType name="BinaryConfig">
  <xsd:sequence>
    <xsd:element name="ConfigurationOptions" type="ConfigurationOptions" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Library">
  <xsd:sequence>
    <xsd:element name="LibraryVersion" type="xsd:string"/>
    <xsd:element name="Package" type="Package"/>
    <xsd:element name="LibraryTime" type="xsd:dateTime"/>
    <xsd:element name="SystemConfig" type="SystemConfig"/>
    <xsd:element name="BinaryConfig" type="BinaryConfig" minOccurs="0"
maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="LibraryName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Architecture">
  <xsd:sequence>
    <xsd:element name="Processor" type="Processor"/>
  </xsd:sequence>
  <xsd:attribute name="ArchitectureName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Processor">
  <xsd:sequence>
    <xsd:element name="VendorID" type="xsd:string"/>
    <xsd:element name="ModelName" type="xsd:string"/>
    <xsd:element name="ProcessorFlags" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="OS">
  <xsd:sequence>
    <xsd:element name="OSVersion" type="xsd:string"/>
    <xsd:element name="OSdistribution" type="xsd:string"/>
    <xsd:element name="KernelName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="KernelVersion" type="xsd:string" minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="OSName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="Compiler">
  <xsd:sequence>
    <xsd:element name="CompilerVersion" type="xsd:string"/>
    <xsd:element name="CompilerUpdates" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xsd:element name="CompilerFlags" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="CompilerName" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>

<xsd:complexType name="ConfigurationOptions">
  <xsd:sequence>
    <xsd:element name="ConfigValue" type="xsd:string"/>
  </xsd:sequence>
  <xsd:attribute name="ConfigName" type="xsd:string"/>
</xsd:complexType>

</xsd:schema>

```

Provenance Editor	
File	
Project	
ProjectID	ICBM
Description	The goal of ICBM is the development...
Funding	NIBIB P01 EB001955
Comment	
Subject	
SubjectID	UCLA_1243
Birth	1987-05-18
Age	019Y
Death	
Sex	F
Species	Human
Strain	
Comment	
AcquisitionProvenance	
AcquisitionTime	2006-05-18T15:54:18
VoxelDimensionX	256
VoxelDimensionY	256
VoxelDimensionZ	1
StepSizeX	1.0
StepSizeY	1.0
StepSizeZ	1
BitDepth	16
Manufacturer	SIEMENS
Model	Sonata
MRIAcquisition	
AcquisitionName	SAG MPAGE 8 CHANNEL
AcquisitionType	3D
MRATechnique	
Weighting	PD
PulseSequence	tf13d1
TE	4.38
TR	1900
TI	1100
FlipAngle	15
AcquisitionPlane	
Coil	Body
FieldStrength	1.4939999580383
Comment	
DescriptionDump	
fileDescription	
fileFormat	DICOM
formatSubtype	
fileCreateTime	
fileName	ICBM_UCLA_1243_MR_SAG_MPAG...
BitDepth	16
VoxelDimensionX	256
VoxelDimensionY	256
VoxelDimensionZ	1
StepSizeX	1.0
StepSizeY	1.0
StepSizeZ	1
StartX	
StartY	
StartZ	
Comment	

Figure 1. Provenance Editor. A screen capture of the Provenance Editor demonstrating a subset of the information extracted from a DICOM header file and a number of user modifiable tags.

The screenshot shows the 'Module Definition' window with the 'Parameters' tab selected. The 'Program Location' is set to '/usr/local/air-5.2.5_64_16/bin/alignlinear' and 'Remote' is checked. The 'Server address' is 'cranium.loni.ucla.edu'. A table lists parameters: 'Standard Volume' (Required, Input), 'Reslice Volume' (Required, Input), 'Linear Transformation' (Required, Input), 'Model Number' (Required, Input), and 'Threshold Standard' (Required, Input). Below the table are 'Remove' and 'Add' buttons. The 'Description' field contains 'The atlas that the input file is resliced to match.' The 'Switch' field is empty and 'Space after switch' is checked. The 'Type' is 'File' and 'Arguments' is '1'. The 'File type' list includes 'Analyze Image' (checked), '1D File', '3D File', and '3DM file'. There are 'New...', 'Advanced...', 'Cancel', and 'OK' buttons.

Parameter Name	Required	Input
Standard Volume	☒	☒
Reslice Volume	☒	☒
Linear Transformation	☒	☒
Model Number	☒	☒
Threshold Standard	☒	☒

Figure 2. Pipeline Module Definition. A screen capture of a LONI Pipeline module definition for the Alignlinear (AIR 5.2.5) executable.

Figure 3. An Independent Components Analysis Workflow. A screen capture of a LONI Pipeline workflow for the calculation of the independent components of a series of BOLD images.

Figure 4.

Different Architectures Yield Different Results. **A**, The result of an independent components analysis of a series of BOLD functional MRI brain images using workflow A. The lower right panel shows the intensity distribution of the image. The scale is 10^5 voxels. **B**, The difference between images A and C rendered in false color to highlight differences. The lower right panel shows the intensity distribution of the image. The scale is 10^5 voxels. **C**, The result of an independent components analysis of a series of BOLD functional MRI brain images using workflow B. The lower right panel shows the intensity distribution of the image. The scale is 10^5 voxels.

Figure 5.

A Non-linear Alignment Workflow. A screen capture of a LONI Pipeline workflow for the non-linear alignment of two images.

Figure 6.

Different Compilation Options Yield Different Results. **A**, A T2-weighted mouse brain image that has been aligned to a minimum deformation atlas using workflow C. The lower right panel shows the intensity distribution of the image. The scale is 10^5 voxels. **B**, The difference between panels A and C rendered in false color to highlight differences. The lower right panel shows the intensity distribution of the image. The scale is 10^5 voxels. **C**, A T2-weighted mouse brain image that has been aligned to a minimum deformation atlas using workflow D. The lower right panel shows the intensity distribution of the image. The scale is 10^5 voxels.

Figure 7. Complex Heterogeneous Pipeline. A screen capture of a complex, heterogeneous LONI Pipeline workflow demonstrating that any number of executables can be described in the Pipeline and therefore represented by the provenance schema.

Table 1
Structure of Data Provenance File
 Outline describing the major elements of the data provenance file.

Provenance
Neuroimaging data provenance
Project
Subject
Species
Age
Sex
Acquisition
Scanner
Orientation
Weighting
Field Strength
TR
TE
TI
Workflow provenance
Executable provenance
Environment
Options
Input files
Output files
Binary provenance
Binary configuration
Configuration options
System configuration
Architecture
Operating system
Compiler
Libraries
Script provenance
Shell
Script
Binary provenance
Pipeline provenance
Pipeline workflow

Data Provenance Systems

A listing of currently active workflow systems that incorporate provenance collection. Data-oriented systems collect provenance information by directly associating provenance metadata with data. Process-oriented systems reconstruct provenance metadata by inverting the processes that created the data.

	CMCS	VDS	myGrid	XCEDE	LONI
URL	cmcs.org	vds.uchicago.edu	www.mygrid.org.uk	nbim.net	provenance.loni.ucla.edu
Primary Domain	chemistry	astronomy, physics	biology	functional neuroimaging	neuroimaging
Model	data-oriented	process-oriented	process-oriented	data-oriented	data-oriented
Format	DublinCore XML	XML/RDF	VDL	XML	XML