Neuroinformatics

1539-2791/07/11–34/$30.00 (Online) 1559-0089
DOI: 10.1385/NI:5:1:11

Original Article

The Extensible Neuroimaging Archive Toolkit

An Informatics Platform for Managing, Exploring, and Sharing Neuroimaging Data

Daniel S. Marcus,[*,1] Timothy R. Olsen,[1] Mohana Ramaratnam,[1] and Randy L. Buckner[2–5]*

[1]Department of Radiology, Washington University School of Medicine, St. Louis, MO; [2]Department of Psychology, Center for Brain Science, Harvard University, Cambridge, MA; [3]Department of Radiology, Harvard Medical School, Boston, MA; [4]Athinoula A. Martinos Center for Biomedical Imaging, Massachusetts General Hospital, Charlestown, MA; and [5]Howard Hughes Medical Institute

Abstract

The Extensible Neuroimaging Archive Toolkit (XNAT) is a software platform designed to facilitate common management and productivity tasks for neuroimaging and associated data. In particular, XNAT enables quality-control procedures and provides secure access to and storage of data. XNAT follows a three-tiered architecture that includes a data archive, user interface, and middleware engine. Data can be entered into the archive as XML or through data entry forms. Newly added data are stored in a virtual quarantine until an authorized user has validated it. XNAT subsequently maintains a history profile to track all changes made to the managed data. User access to the archive is provided by a secure web application. The web application provides a number of quality control and productivity features, including data entry forms, data-type-specific searches, searches that combine across data types, detailed reports, and listings of experimental data, upload/download tools, access to standard laboratory workflows, and administration and security tools. XNAT also includes an online image viewer that supports a number of common neuroimaging formats, including DICOM and Analyze. The viewer can be extended to support additional formats and to generate custom displays. By managing data with XNAT, laboratories are prepared to better maintain the long-term integrity of their data, to explore emergent relations across data types, and to share their data with the broader neuroimaging community.

Index Entries: Data sharing; database; genetics; informatics; metadata; neuroinformatics; open source; quality control; workflow; XML schema.

(Neuroinformatics DOI: 10.1385/NI:5:1:11)

*Author to whom all correspondence and reprint requests should be addressed.
E-mail: dmarcus@wustl.edu

Introduction

Neuroimaging laboratories are faced with a number of data management challenges. Within the laboratory, data must be passed through a series of capture, quality control, processing, and utilization steps. Maintaining the long-term usability and integrity of data require investigators to maintain vigilant oversight of the data through each of these steps. In the broader scope, subsets of these data often need to be shared with specific collaborating colleagues, and limited or anonymized versions of the data shared with the general community. Data sharing and privacy policies instituted by funding agencies and publishers are making data exchange commonplace (e.g., NIH data sharing statement,* HIPAA privacy standards†) (Martone et al., 2004; Van Horn et al., 2004; Gardner et al., 2003; Toga, 2002). Various end users—laboratory personnel, collaborators, and the general community—also require appropriate tools to explore the data. These tools should make the data highly available to intended users whereas enforcing security protocols that restrict unintended access. These issues are particularly challenging in the context of neuroimaging because imaging data sets are large, require complex processing pipelines, and follow complicated experimental protocols. Additionally, neuroimaging studies routinely incorporate measures from a range of other experimental approaches—genetic, clinical, neuropsychological—which must be integrated with the imaging measures into a unified data set.

Here, we present the Extensible Neuroimaging Archive Toolkit (XNAT), a software platform designed to address the data management challenges that neuroimaging laboratories face. In particular, XNAT was designed to capture data from multiple sources, to maintain the data in a secure repository, and to distribute the data to approved users (Fig. 1). User interaction with XNAT mirrors laboratory best practices for maintaining the quality and integrity of data. The XNAT user interface provides users with tools to manage the data from entry and storage through processing, access, and distribution. Data entry and upload forms as well as scriptable command-line programs, allow data to be easily captured into the archive. Quality-control tools enable users to inspect, validate, and process the data. Search, display, and download tools facilitate exploration of the archive. By maintaining direct control of the data through all of these actions, XNAT protects the overall integrity of the data and minimizes accidental or intentional misuse of the data. Backup processes can be scheduled and security protocols can be enforced. User interaction with the data is logged. Modifications to data are recorded and validated. Common processing routines can be automated and resulting-derived measures can be programmatically captured back into the archive. The resulting archive is a clean and rich resource for use within the laboratory and for sharing with the broader community.

Several other applications provide similar, though not entirely overlapping, functionality as XNAT for managing neuroimaging and similar data. The functional magnetic resonance imaging (fMRI) Data Center's Data Management Tool§ , based on the Protégé¶ "knowledge-base" and ontology editor (Noy et al., 2003), provides a detailed fMRI ontology onto which experimental data and metadata are mapped. Laboratory-specific plug-ins can be

*http://grants.nih.gov/grants/policy/data_sharing/.
†http://www.hhs.gov/ocr/combinedregtext.pdf.
§ http://www.fmridc.org/f/fmridc/dmt/index.html.
¶http://protege.stanford.edu/.

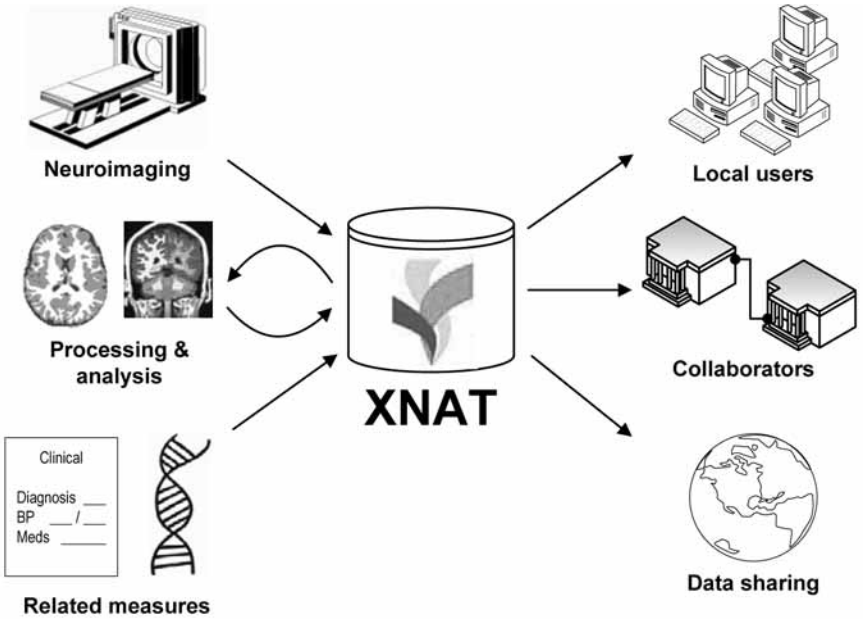


Fig. 1. XNAT serves as a hub for capturing data from multiple sources and distributing it to a variety of end users.

added to extend the base ontology. Yale University's SenseLab* provides an object-oriented database that flexibly stores experimental data and metadata about how the data should be displayed within a Web-based user interface (Nadkarni et al, 1999; Marenco et al., 2003). The Human Imaging Database, developed by the Biomedical Informatics Research Network (BIRN), also provides a web-based user interface and a relational database backend (Ozyurt et al., 2004). Access to the actual data files within the Human Imaging Database is provided by the Storage Resource Broker† (Baru et al., 1998), which maintains a virtual directory system of local and remote files. The GridPACS§ system, developed at Ohio State University similarly provides a distributed data archive and utilizes grid technology for implementing processing pipelines (Hastings et al., 2005). Data management systems targeted at more generic data that use similar XML-based technologies as XNAT include Silkroute (Fernandez et al., 2002), DataServer (Bui et al., 2002), and ShreX (Du et al., 2004).

The XNAT framework relies heavily on XML¶ and XML Schema** for its data representation, security system, and generation of user interface content. XML provides a powerful tool for building extensible data models. This extensibility is particularly important in rapidly advancing fields like neuroimaging, where the managed data types are likely to change and evolve quickly. XML Schema has become the standard language for defining open and extensible XML data formats. As a result, many biomedical organizations have

*http://senselab.med.yale.edu/senselab/.
†http://www.npaci.edu/DICE/SRB/.
§http://bmi.osu.edu/areas_and_projects/mobius.cfm.
¶http://www.w3.org/XML/.
**http://www.w3.org/XML/Schema.

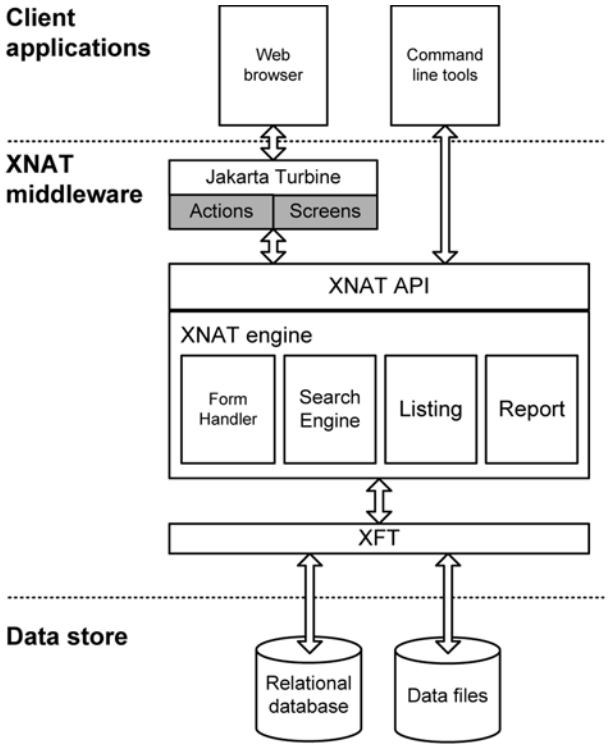


Fig. 2. The XNAT architecture consists of three basic tiers: the data store, middleware, and client applications. The Extensible Formatting Tool (XFT) and XNAT engine implement behind-the-scenes operations such as querying and writing to the data store. Developers wishing to customize XNAT implementations need only to familiarize themselves with the XNAT API, which provides a simple interface to the engine. The XNAT API can be accessed directly from Velocity-based HTML pages within the web application.

developed or are currently developing standards in XML (Keator et al., 2006; Westbrook et al., 2005; Zhao et al., 2005; Wang et al., 2002; Dolin et al., 2001). By focusing on XML Schema, XNAT allows laboratories to leverage these industry-standard schemas, to extend XNAT's included neuroimaging schema, or to build their own schemas from scratch. XNAT deployments that use common schemas can easily interoperate with one another and with other systems that support these schemas. Using XML transformation languages like XSLT, data can also be easily imported and exported in additional formats like HTML, PDF, spreadsheets, and alternative XML models.

The remainder of this article focuses on the informatics contributions of the XNAT platform. After a brief technical overview, we describe how XNAT abstracts laboratory best practices into a standard workflow and implements user interface, quality control, data storage, and security features to support this workflow. Subsequently, we describe how XNAT can be customized to meet site specific needs. Finally, we discuss real-world use and limitations of XNAT, and directions for future development.

Technical Overview

XNAT's architecture follows a widely used three-tier design pattern that includes a relational database backend, Java-based middleware, and a web-based user interface (Fig. 2) (Aarsten et al., 1996). XML schema documents (XSDs)—including the schema provided with XNAT and/or schemas supplied by a site that is deploying XNAT—define the data types that are to be handled by the deployed system. XNAT uses these schemas to generate custom components, content, and logical for each of the tiers: a relational database is generated that contains tables, columns, and relations equivalent to the structures defined in the XSDs; middleware classes are generated that can be used by developers to implement custom functionality that utilizes the XNAT database; and user interface content is generated, including navigation menus, search options, and data tables. The schemas are also used extensively in XNAT's security, validation, and search tools. The resulting system represents a fully operational data management system tailored specifically to the site's data model. During operation, XNAT mediates incoming and outgoing data requests, translating as necessary between the relational database, XML, and web-based formats. In addition to the XML-based data, image

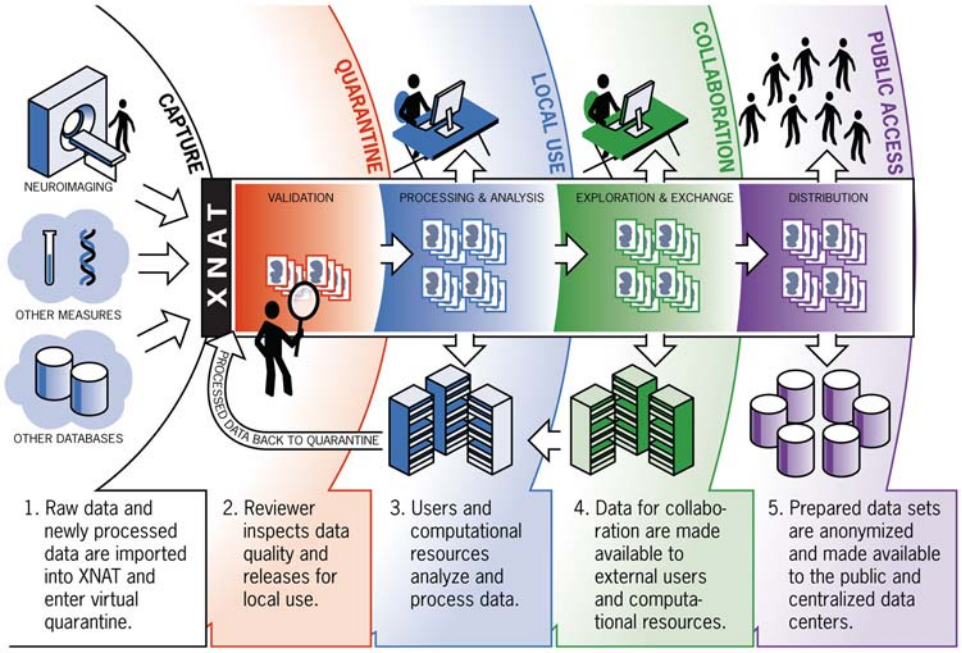


Fig. 3. The XNAT workflow.

data are stored in their native format on the platform file system. Links to these files in the database provide XNAT with access to information for distributing and displaying the images. XNAT's architecture is covered in more detail in later sections and comprehensively on the XNAT website (www.xnat.org).

The XNAT Workflow

A primary focus of the XNAT project has been to identify the practices that neuroimaging laboratories commonly use to manage their data from acquisition to publication. As a result of this effort, we have developed XNAT around a standardized workflow that includes data capture, quality control and automation, local use, collaboration, and public access (Fig. 3). The XNAT platform implements ergonomic tools to support each component of this workflow.

1. *Capture.* The first step in the XNAT workflow is to capture imaging data and metadata.

After scanning is complete, the laboratory member responsible for the session data first logs into the XNAT web application and enters session metadata, including an ID and sequence type for each scan run during the session, through a web-based form. Using standard transfer tools (e.g., DICOM "pushes,"* FTP, portable hard media) or XNAT's web-based *Upload* tool, she then loads the image data into a user-specific *prearchive*. The user's prearchive serves as a temporary cache, where the user can inspect the data to ensure that it was loaded without loss or corruption. The files are then transferred from the user's prearchive into XNAT's secure central archive using XNAT's *Transfer* tool. The distinction between user-owned local copies vs central archive copies of data is a fundamental strategy in XNAT for maintaining data integrity (Fig. 4). Central copies are protected from accidental or malicious manipulation. Standard processing workflows are run on these copies, and resulting

*http://medical.nema.org/dicom/2004.html.

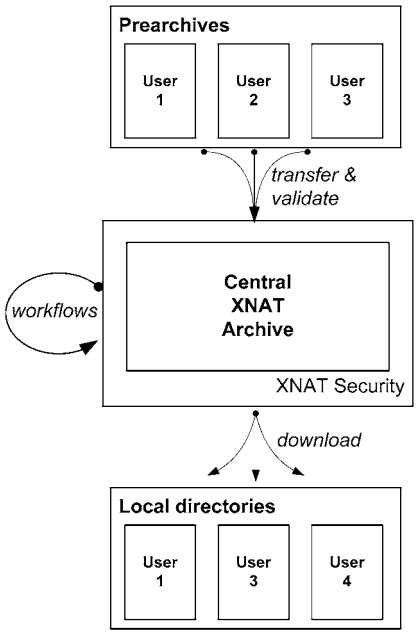


Fig. 4. The XNAT file infrastructure includes individual user prearchives and a central archive. Web application tools enable transferring data from the prearchives into the archive. Data can be downloaded from the archive for local use, whereas standard processing is executed within the archive using XNAT's workflow tools.

files become a part of the protected archive. Local copies, on the other hand, allow users to prepare image files before archiving and to run individualized analyses on data retrieved from the archive.

XNAT also provides methods for capturing nonimaging data into the archive. One method is to use datatype-specific web-based forms for hand entering the data. When XNAT is deployed, these forms are automatically generated for each of the data types included in a site's XSDs. Web-based data entry is appropriate for capturing data that does not already exist in a digital form (e.g., transcribing data from paper forms). A second method for capturing nonimage data is through XML documents. Spreadsheets, for example, can be output to XML (using Microsoft Excel 2003's "Map to XML" feature,* for example). Similarly, data can be directly exported as XML from many applications or translated into XML using custom scripts. XML documents can be loaded automatically into the archive using XNAT's *StoreXML* tool. This method of data entry has the benefit that it can be automated and scripted.

2. *Quality control and automation.* XNAT implements additional quality-control features at the most vulnerable points in the data life cycle. When nonimaging data are newly entered (using either form-based or XML entry methods) or edited, they are placed into a *virtual quarantine*. Similar to the concept of the image file prearchive, virtual quarantine serves as a mechanism to temporarily withhold data from typical archive actions (e.g., search, download) until they are inspected and approved by an authorized user. If the user identifies errors in a quarantined record, she can clean the data using XNAT's online forms or trace the error back to its point of origin and reupload the data file. Data can also be obsoleted, effectively removing them from all archive activities. As a course of routine, XNAT does not permanently delete data from the archive, although data can be removed through a separate process and thereby allowing compliance with certain regulatory requirements.

As data are transferred from the user cache into the central archive, XNAT compares the file count, file sizes, and file header values against the metadata already entered by the user and against a site-wide validation file that describes the expected values for the various protocols used at the site. The validation file can include entries for any of the fields included in the header section of a particular file format. The validation results—whether the data

*http://msdn.microsoft.com/library/en-us/odc_xl2003_ta/html/OfficeExcelXMLMappingScenarios.asp.

match the validation criteria and, if not, what discrepancies occurred—are written to the database and emailed to appropriate users. In addition to validating the data, this process also writes the protocol and scan parameters from the header fields into XNAT's database. Currently, XNAT's image validation process supports DICOM and Siemens's IMA file formats. Nonimaging data are validated by XNAT using standard XML Schema validation.

Raw imaging data are typically passed through a number of preprocessing (e.g., motion correction, atlas transformation) and automated analysis steps as part of a laboratory's standard analysis pipeline. Directing and monitoring the execution of each step in the pipeline is critical for producing usable postprocessed images and derived measures. XNAT provides a number of tools to facilitate this oversight. The web-based *Build* tool allows authorized users to select a set of scans on which a processing pipeline will be executed. XNAT passes the selected scans to the *ArcBuild* script. ArcBuild is essentially a wrapper script that executes a prespecified sequence of processing steps set up by the site administrator. As each step completes, ArcBuild writes a *data provenance* record to the database. The provenance record details the process that was run, the command line parameters that were used, versioning information, and the computing hardware the process was executed on. Information about images that were generated in the pipeline step, including precursor images and the location of the generated image, is also written to the database. These records represent a detailed accounting of the entire workflow. XNAT's online image viewer allows the users to inspect the intermediate and final images described in the records.

3. *Local use.* Although XNAT's centralized processing tools enable standard processing pipelines to be executed on archived data, most studies require additional nonstandard analyses. Early on in XNAT's development, we concluded that we were unlikely to improve upon the many high-quality statistics and analytic applications available to perform such analyses. So instead of building lesser versions of these functions into XNAT, we focused on developing the capture, validation, quality control, and exploration tools that simplify making high-quality data available to these applications. The XNAT web application includes a "Download Spreadsheet" link on every page after a user browses or searches for data. The downloaded spreadsheets can be imported automatically into typical statistics packages. Similarly, local copies of image data can be retrieved from the archive by authorized users for further analysis.

Determining which analyses should be "local" and which should be made part of the central archive is not always straightforward. Generally, groupwise analysis is the best conducted outside of XNAT as it requires numerous combinations of data sets, including sets selected after filtering the data and removing outliers. Exploratory and nonstable methods should also be excluded from the central archive. However, as these methods mature, the resulting measures can be modeled in XML Schema and imported into the archive. Subsequently, acquired data can be automatically evaluated with these methods by adding the methods to a centralized processing pipeline.

XNAT's web-based interface provides a range of tools to enhance user experience. Data can be easily browsed, sorted, searched, and downloaded. As data are explored, they can be annotated and additional measures can be entered. Burns et al. (2005), for example, used XNAT's online tools to assess structural MR images in older adults for white matter lesions (modified from Scheltens et al., 1993). They used XNAT's online viewer to evaluate the images and subsequently entered their assessments in a customized XNAT form. Because XNAT maintains links between related data types—subject, MR data, assessments—their assessments were automatically integrated with clinical, psychometric, and demographic data available for the assessed subjects.

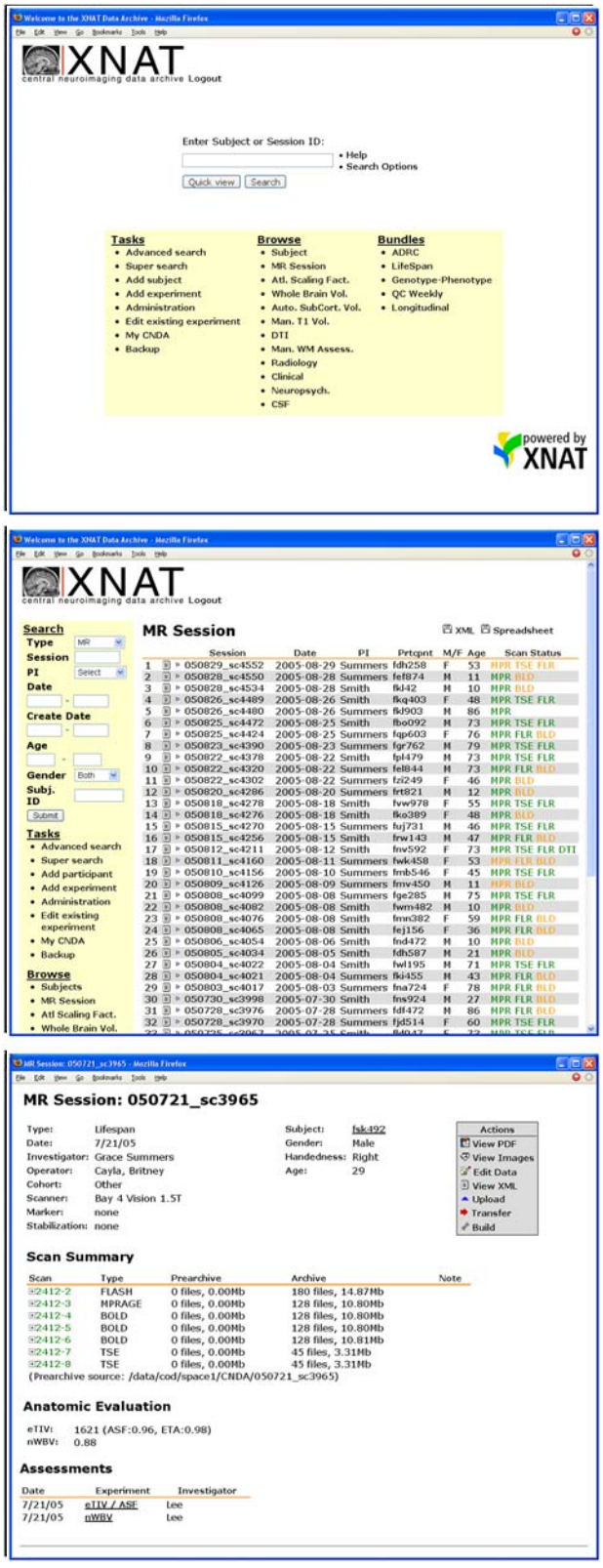
4. *Collaboration.* XNAT also facilitates collaboration. Subsets of the archive can be grouped together into "bundles" that reflect projects, research groups, or other collaborative entities. The bundled data are made available to select users through a prominent link when they login to the site's webpage. As new data are generated that meet a bundle's criteria, they are automatically added to the bundle, thus providing a current view into state of the collaborative endeavor. *Ad hoc* bundles can be created by searching the archive for data meeting-specific criteria and clicking the "Email to colleague" link available on all XNAT webpages. The email recipient can then follow a link in the sent email to view the data of interest (to maintain security, no actual data are sent in the email). Additional collaboration tools are currently under development, including an integrated Wiki page editing system (Leuf and Cunningham, 2001) that will allow discussion and laboratory notebook entries to be created around individual and group records.
5. *Public access.* A number of scientific and sociological trends, including increased collaboration across sites and fields of expertise, guidelines from external stakeholders such as funding sources and industry, and policies from publishers, have added to the incentive to share data. In early stages of a project, data sharing with a closed group of investigators is supported by XNAT's collaboration tools. When a study reaches later stages, likely after results have been published, data sharing with the general community is easily implemented with XNAT's web-based exploration tools. However, because archive data may contain sensitive and identifying data, care must be taken to prepare data for open access. Tools to automatically generate anonymized views of archive data are currently under development. XNAT's XML export and image bundling tools also make it simple to prepare data for upload into federated databases like the fMRI Data Center. An important step to facilitate this type of exchange is for the various database entities to establish standard XML formats and services for exchanging compliant XML documents.

The XNAT User Interface

Web-Based Tools

The XNAT web application provides an ergonomic view into each component of the XNAT workflow (Fig. 5). The application home page, displayed after logging in, features a streamlined search tool and a set of menus. The search tool allows users to find existing subjects and imaging sessions by simply entering a full or partial identifier into the search tool. The menus provide links to three basic categories of information and operations: browsable listings of the primary data types, tools to perform various productivity and administrative tasks, and "bundles" of preconfigured data sets. The overall "look and feel" of the application follows evidence-based usability principles that emphasize rapid load times, simple layouts, and minimal clutter (Nielsen, 2000). While navigating the site, the archive content is displayed in two basic layouts, listings and reports. Listings are tabular displays containing rows of similar records; reports are detailed displays, containing information related to a particular record. A search for all male subjects, for example, would yield a list, whereas a search for a subject with a particular ID would yield a report. The vast majority of the application's content builds on these two basic views, and both are highly customizable.

Interconnected links between the various reports and listings provide much of the navigation structure for the application. In a typical used case of website navigation, a researcher would click on the bundle link for a project she is involved in. She would then browse through the resulting listing to get a sense of the project's overall progress and click on particular items to view more detailed reports. From individual reports, she could



check on the status of the data within the XNAT workflow, use tools to move the data through the workflow, and view quality-control images and derived measures. An "Actions" box on each report lists all of the operations that the user has authorization to perform on the data represented in the report.

In addition to browsing tools, the XNAT web application includes a powerful search tool that allows users to find data using custom search forms generated from the site's XSD data model. If a search returns a single record, the result is displayed as a report. Result sets with multiple records are displayed as listings. A search box in the listing allows users to further filter the data. XNAT's *Super Search* tool allows users to create a query as earlier but to also include related data types in the search criteria and result set. For example, a typical archive may include genetic, clinical, and MRI data that are related by the subject from whom they were obtained. A user could enter a search for 60–80 yr old subjects with a diagnosis of dementia and a particular genotype. The user could then request that the result set includes columns for the demographic, clinical, and genetic data as well as for volumetric measures that were derived from the MR images. When more than

Fig. 5. The XNAT web application gives users access to the data archive from any computer with a web browser and internet connection. The snapshots shown here are taken from a local XNAT deployment. The actual data were scrambled to preserve subject privacy. The home page (top) provides links to all of the data types to which the logged-in user has access, including preloaded "bundles" of commonly requested data. XNAT listings provide (middle) an overview of a particular data type. This listing of MR session data has been customized to include color-coded indicators of the processing status of its constituent scans. XNAT reports (bottom) provide detailed views of a particular data set. This MR session report includes synopses of derived anatomic measures and links to a number of XNAT tools.

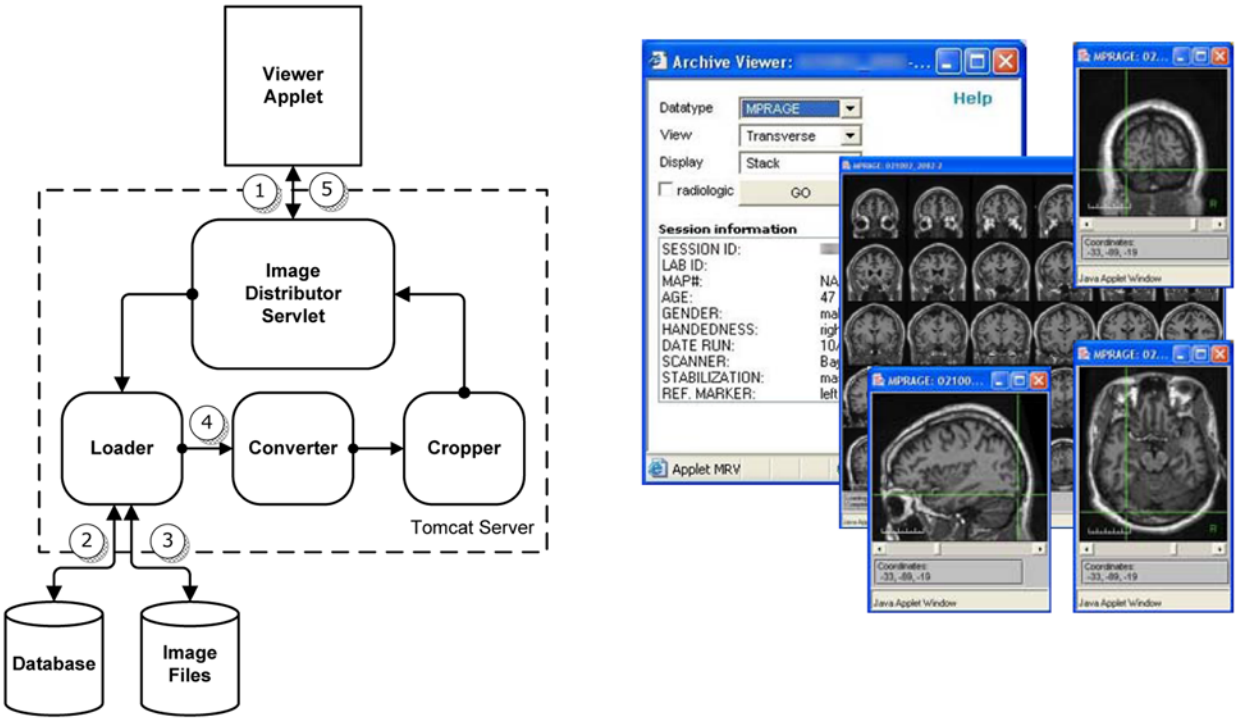


Fig. 6. The image viewer architecture (left) includes a client-side Java applet (right), a server-side Java servlet, and supporting Java classes. The client and server components are built using ImageJ,* an open source image processing software package. (1) Requests from the applet are received by the servlet, (2) The Loader class queries the database to determine where the files are located, and (3) Reads the image data from the appropriate files. In typical display mode, the viewer loads a low-resolution version of the requested image, which speeds transfer of the image. In cases where full resolution displays are required, removing the low-resolution entry from the viewer's specification file forces the viewer to load the full-resolution image. (4) The server components crops the image according to the configuration file entry to further minimize the image size. (5) The image is sent to the applet using HTTP tunneling. Because HTTP tunneling uses the web server port, the viewer can use SSL-based encryption and communicate with servers behind department firewalls and virtual private networks (Juric et al., 2004). The actual image data are placed into a Java object that implements the Serializable interface, which allows the object to be broken into transportable pieces and reassembled by the client. The applet provides an ergonomic view of the images.

one potential match is present in the database—multiple clinical assessments for a subject, for example—an additional field can be specified to direct which records are merged. Typically, a date field is used so that data that were obtained in closest proximity to one another are joined. As with all XNAT listings, search results can be downloaded as a comma delimited file.

The web application allows data to be entered and edited through HTML forms. During setup, XNAT automatically generates these forms based on the site's included XSDs. XNAT automatically parses submitted form data and validates it against the XSD. XNAT writes valid submissions to the database, and returns invalid submissions to the user with a descriptive error message. Sites can edit the

*http://rsb.info.nih.gov/ij/.

autogenerated forms to improve their ergonomics, remove undesired fields, and add additional fields.

Image Viewing Tools

In addition to the text and numeric measures displayable in the web application's lists and reports, XNAT also includes an integrated online viewer that provides access to all of the images in the archive (Fig. 6). The viewer allows users to inspect raw and postprocessed images for acquisition quality and algorithm errors, to conduct studies such as lesion counts (e.g., Scheltens et al., 1993) and radiological assessments, and to share particular cases with colleagues. The main viewer window provides access to basic display features like file selection, image orientation, and viewing mode (either montage or stacks). Once the desired display settings are configured, the user clicks a "Go" button to view the image. The viewer relies on plug-ins to implement image type-specific functionality. The default structural MRI plug-in, for example, implements basic features like contrast adjustment, whereas the overlay plug-in implements more complex views that include base-structural images and color-mapped overlays. Additional plug-ins can be developed by sites to support their local requirements.

Command Line Tools

Whereas the web application is useful for interactive exploration of a data archive, command-line tools provide the ability to script common tasks and run them in batch mode. The *Browse* and *Search* programs can be used to retrieve data from the archive. Browse returns all instances of a specified schema element, whereas Search returns subsets of the data according to specified criteria. Data retrieved using these tools can be formatted as XML, CSV, or HTML. The *StoreXML* program can be used to load XML documents into the archive. Combining Search with StoreXML within an automation script allows the script to pull a particular data set (e.g., structural MR images) from the archive, run some processing routine on the retrieved data, and subsequently load the resulting-derived data (e.g., regional volumes) into the archive.

Modeling and Storing XNAT Data

XNAT's Storage Architecture

XNAT uses a hybrid storage architecture that leverages the strengths of XML, relational databases, and standard file systems. Data stored by XNAT are modeled in XML using XML Schema. From the XSDs supplied by a site, XNAT generates a corresponding relational database that actually stores all of the nonimage data. XNAT automatically imports and exports compliant XML to and from the generated database. Image data remain as flat files in their native format (e.g., DICOM) on the file system. These files are represented as URI* links in the database and XML.

This hybrid XML/relational/file system architecture has a number of advantages. By building on a data model in the XML domain, the XNAT platform is able to generate a great deal of content from the known structure of XML documents and XNAT sites can easily utilize the growing set of XML-based services and technologies. By storing the text and numeric data in a relational database, the typical drawbacks of XML data representations—inefficient storage and querying—are avoided. By storing the image data in flat files, the cumbersome nature of binary types in XML and databases is avoided and the images can be directly accessed by users and applications.

However, XNAT's hybrid architecture does potentially come at some cost. In particular,

*http://www.w3.org/Addressing/.

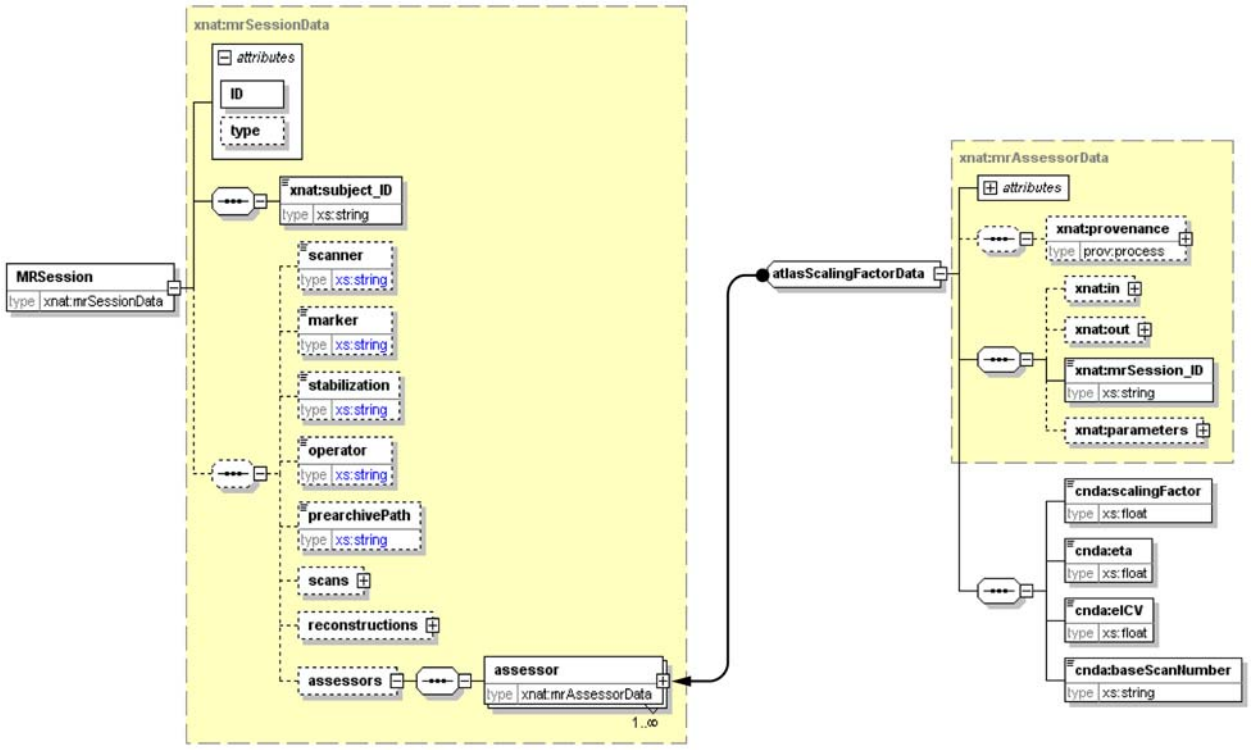


Fig. 7. This schematic snippet from the XNAT schema illustrates the extensible design of the schema. The MRSession element (left), derived from ImageSession can include one or more "mrAssessorData" child elements. mrAssessorData is an abstract type that can be extended to capture specific experimental protocols that operate on MR data. The "atlasScalingFactorData" type (right) extends mrAssessorData to capture a set of atlas registration measures (Buckner et al., 2004) used in Washington University's ADRC.

generating a reasonably efficient and robust database from XSDs that were intended more for data exchange than data storage is a significant challenge. This issue is discussed in detail later. Another challenge of the hybrid architecture is that it requires site administrators to be aware of two separate data representations. While XNAT's fairly straightforward mapping between the schema and database representations mitigates this burden, it does nonetheless require some effort to learn the database structure. The dual representations also require that changes to the data model percolate through both the XSDs and the database. Update tools within the XNAT package ease this process, but some manipulation of the database by hand may be required to reflect changes to the data model.

Modeling Neuroimaging Data in XML

XNAT includes a base XSD that provides a framework for capturing both specific experimental protocols and the general hierarchy of metadata associated with these protocols (Fig. 7). The schema includes a set of abstract data types embedded within a hierarchy consisting of Projects, Subjects, and Experiments. Specific protocols are implemented by deriving concrete data types from the abstract types and using these in place of the abstract types within the hierarchy. For example, the abstract *SubjectAssessor* type has been extended to

implement an *MRSession* type that defines the details of an MR acquisition (e.g., scan types, scanner settings). Similarly, the abstract *MRAssessor* type can be extended to capture measures generated from the images described in *MRSession*. Some of the data types that have been derived from the abstract types include demographics, clinical assessments, neuropsychological exams, behavioral tests, brain-tissue segmentations, and brain region segmentations. Details of the core schema types and a number of derived types can be found at http://www.xnat.org/schema.

XNAT's XSD includes a number of concepts borrowed from relational database design that make it more efficient for storing data and more amenable to mapping into a database schema. In general, it follows the normalization guidelines recommended in the literature for approximating the BCNF (Lee et al., 2002; Hartmann and Link, 2003; Arenas and Libkin, 2004). As a result, its basic data types include ID fields, which are used like primary keys to uniquely identify instance data. These IDs can be used in related data types to refer to the full instance, similarly to how foreign keys are used in the relational model.

Schemas that do not derive from the base XNAT XSD can also be integrated into an archive's data model. Using a configuration file included in XNAT, types defined in these schemas can be related to types defined in the XNAT schema. For example, XNAT's Subject element could be linked to a Genotype element in a genetics schema, enabling XNAT to implement searches and content that integrates fields from both data types. The abstract types in the XNAT schema provide a minimal framework from which new types can be derived, giving implementers a great deal of flexibility in how they model their data. A primary design consideration is choosing between a general or specific data model. Neuropsychological data, for example, could be modeled as a single element that captures a broad range of tests or as a set of elements specific for each test. A more general data model leads to a more compact database structure and requires less frequent amending to capture new tests and assessments. A more specified data model can be more deeply validated, is more self-documenting, and is easier to search and transform. In our own deployments, we have found that highly specified models and the validation that can be done on them, are preferable. Nonetheless, the best practice for deciding between general and specific models for experimental data remains an open question and one worth substantial consideration.

This and many other issues associated with XML data models would be well served by a broad dialogue within the neuroinformatics community. Along these lines, the XNAT schema is currently being harmonized with similar data models developed by the BIRN (Keator et al., 2006) and the fMRI Data Center, with the goal of creating a data format that will be supported by a wide range of applications and databases. It is anticipated that this and other developments will lead to changes in the XNAT data model over time. As changes are made, XSLT-based scripts will be provided to aid sites in transforming their data to comply with newer versions of the XNAT schema. Depending on site requirements, these transformations could be applied to the entire archive or to individual documents as they are exchanged with other entities.

Generating the Database

XNAT's hybrid data storage architecture requires a mechanism for generating a database schema from specified XSDs. The challenge in such schema "shredding" is overcoming the impedance mismatch between XML's tree structure and normalized relational models. For example, an XML representation of an MR study would typically include a root study element

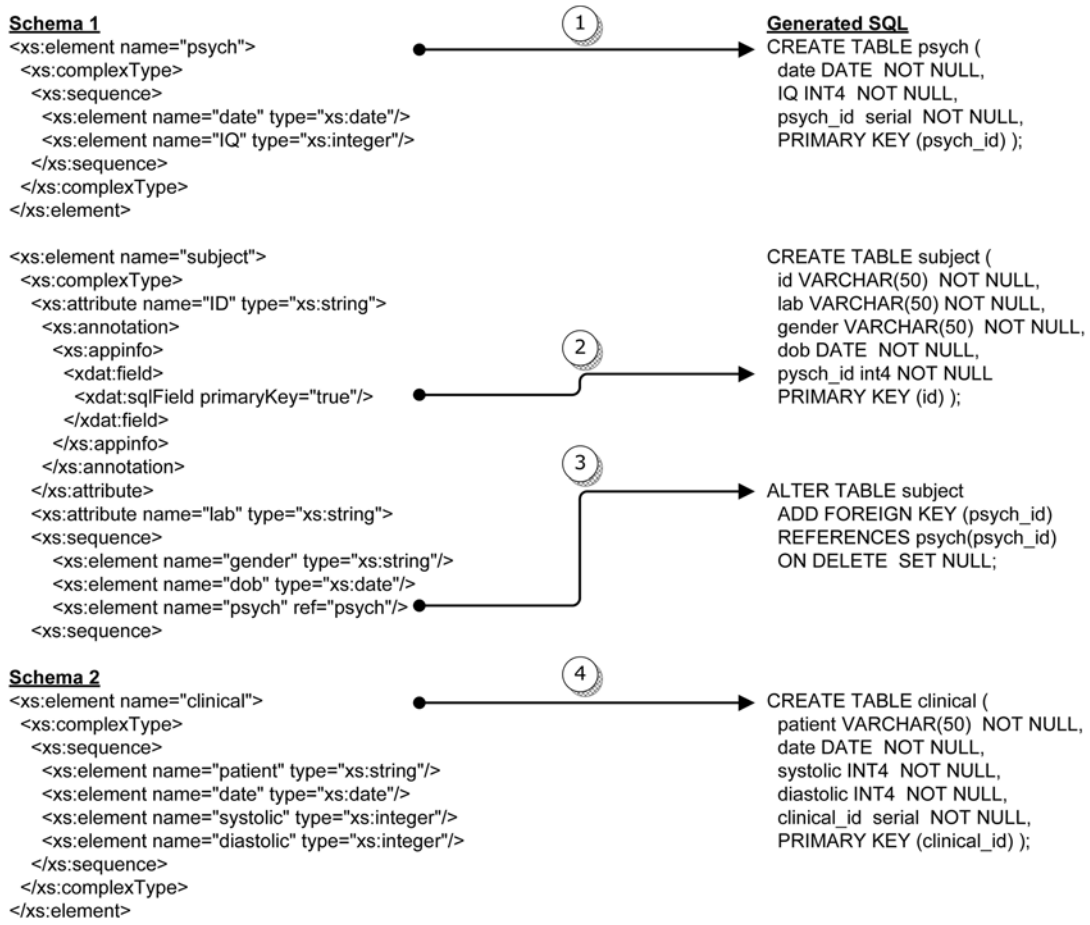


Fig. 8. XNAT maps XML Schema elements to database tables following a straightforward rule set as illustrated here. (1) Simple XML Schema elements map to a single table with a column for each child element and attribute. (2) Primary key columns are created if none are indicated within the schema. Using custom XNAT tags, appropriate fields like subject's ID attribute can be indicated as the primary key column in the corresponding table. References to global elements map to relations in the generated database. (3) A reference to the *psych* element in *subject* maps to a foreign key in the subject table to the psych table. (4) When more than one schema is present, additional tables are created in the database. Proper namespacing within the schemas is maintained in the database to prevent conflicts in table names.

with a child branch describing the scanner on which the study was run. If the shredder maps the scanner branch as an attribute of the study (i.e., as columns in a scanner table), then the resulting database will be denormalized. On the other hand, mapping the scanner as an independent entity requires information for generating keys that is not present in the schema.

A number of shredding approaches have been described in the literature (*see* Amer-Yahia et al., 2004 for a summary). Though independently developed, XNAT uses a shared inlining technique similar to those previously described (Shanmugasundaram et al., 1999; Varlamis and Vazirgiannis, 2001; Du et al., 2004; Chaudhuri et al., 2005). XNAT also implements a number of directives that can be added as schema annotations to guide the mapping process in handling the normalization issues described earlier. The basic mapping is as follows (Fig. 8). All global elements and types map to individual tables. Each of these elements' attributes and

single occurrence SimpleType child elements map to columns in the parent table. The table and column names follow a convention using the XSD path, with underscores separating path levels. Each column is assigned an appropriate datatype based on the XML Schema primitive type being mapped. SimpleType child elements that can occur more than once map to their own tables with foreign keys to the parent table, allowing multiple rows to relate to a single parent row (i.e., 1:M relation). ComplexType child elements map to their own tables, with columns and child tables generated as with top level elements. References to global elements map to relations in the database, either as an additional table (M:N) or as a foreign key in the table corresponding to the global element (1:M). Every table in the generated database is assigned a primary key. If an appropriate column already exists, it can be identified by adding a custom schema annotation. Otherwise, an auto-incrementing column is automatically added to the table.

Interacting With the Database

From an operational perspective, XNAT's functions are primarily in the XML domain: new data are imported as XML documents, data entry form fields use an XPath* naming style, and queries against the database are written using simple XQuery†-like structures. These operations are implemented by XNAT's middleware component, the Extensible Formatting Tool (XFT). XFT maintains an internal object-based representation of the site's data model. This representation is akin to an XML Infoset,§ in that it provides a decomposition of XML documents. However, it also provides a mapping into the auto-generated database model for each information item. This allows XNAT to accept XML-based data and queries on the frontend whereas actually storing data and implementing queries as SQL on the backend.

Queries against the archive can be expressed from the command line using XNAT's *Search* program. Arguments to the program include which XML element to return, which fields to search against, and the actual search criteria. As an example (using the XNAT base XSD), a search for MR sessions that include scans with repetition times of less than 3000 ms would be written as follows: Search –f MRSession.scans.scan.parameters.tr -v 3000 –c LESS_THAN. Based on XFT's internal information item for the "tr" element, an appropriate SQL query is generated and executed; the result set is stored within XFT; and finally the corresponding XML is output to the user.

The development of XNAT's query language is ongoing. An important goal is to develop full compatibility with XQuery. A concern introduced by Krishnamurthy et al., (2003) is that certain XML-based queries are difficult to implement in relational databases. This limitation is restricted to recursive XML constructs and has not had a practical impact on XNAT archives to date. However, continued consideration of this issue is warranted.

XNAT also allows SQL queries to be stored as views in the database and accessed from XNAT's web application. Although these stored queries are less dynamic than the XML-based queries, they provide the ability to implement complex joins that are not currently supported in XNAT's query language and to generate queries on derived fields in the view.

Storing Image Files

Because the paths to image files are represented explicitly in XNAT's data model, no particular file organization scheme is required in

*http://www.w3c.org/TR/xpath.
†http://www.w3c.org/TR/xquery.
§http://www.w3c.org/TR/xml-infoset.

order for XNAT to access the images. However, it is recommended that sites implement and consistently follow some well-considered scheme. For example, a single archive root directory, containing a hierarchical series of subdirectories representing projects, subjects, visits, and/or experimental data allows subsets of the archive to be efficiently backed up, moved, and bundled. By adding a layer of subarchive directories between the root directory and the data directories, the archive could be distributed across an extendable set of disc partitions.

Security

XNAT was designed to manage a large number of scientific projects within a single archive deployment. The benefits of such "data mingling" include a reduction in a site's overall administrative costs, opportunities to mine and share data across projects, and improved coordination of subject participation between studies. However, it comes with the increased burden of needing to restrict individual user's access to the appropriate subset of records. XNAT enables this level of access control by implementing a security system that leverages XNAT's hybrid XML-relational database structure: security protocols are defined against the XML data model and these protocols are then enforced when data is retrieved from the relational database.

The XML-based security protocol, though independently developed, follows a model similar to that proposed by Ilioudis et al. (2001) and especially Carminati and Ferrari (2005). The protocol is defined by selecting one or more *security fields* in each data type included in a site's managed XSDs. Each security field is then assigned one or more *allowed values* in each user's account, indicating the values for the field that if present, grant the user access to a record of that data type. For instance, for a Clinical Assessment data type, a field called Investigator could be labeled as a security field. When setting up user Brittany Smith's account, the Investigator security field could be assigned allowed values of "John Anderson" and "Cayla Jones." Subsequently, Brittany Smith will be only granted access to Clinical Assessment records that have either "John Anderson" or "Cayla Jones" in the Investigator field. In addition, each allowed value entry includes the access privileges (create, read, update, and delete) the user has for records matching that value. Brittany Smith, for example, could be assigned read-only access to John Anderson's clinical assessments and full create/edit/delete privileges for Cayla Jones' assessments. By selecting appropriate security fields, access control can be established for complex groups of data. For example, the Investigator field is preconfigured as a security field for the data types in the core XNAT XSD. Data types that derive from these core schema elements automatically inherit the Investigator security field. Sites can add additional security fields as needed.

Actual enforcement of the security protocol is achieved when queries are submitted to the relational database. When a user requests records of some type from the archive, XNAT checks the security protocol for that user's allowed values and appends these as parameters to the query submitted to the relational database (i.e., a "WHERE" clause is added to the SQL query). In this regard, XNAT's security system resembles the row-level security found in Oracle* and other database systems. Providing such granular access control allows XNAT sites to comply with common privacy and regulatory requirements.

XNAT uses roles within its security system to grant access to more general actions within

*http://www.oracle.com/technology/deploy/security/db_security/pdf/twp_security_db_database_10gr2.pdf.

the archive. Typical users are assigned the SiteUser role, which allows them to use the archive website and command line tools. Users responsible for maintaining data can also be assigned the DataManager role, which allows them to access the XNAT workflow tools. High-level users can be assigned Administrator and Boss roles, which allow them to create and edit user accounts, create and assign additional roles to reflect a site's particular responsibilities, and to access a number of additional administrative tools.

The XNAT web application includes administrative tools for maintaining security protocols. Administrators can create and edit user accounts, assign roles, and allowed values for the core security fields, and activate or deactivate login privileges to the archive website. Additionally, these tools can be used to edit the data types available to web users, to monitor user activity, and to send e-mails to individual or multiple users. A current limitation of the security interface is that is does not allow additional security fields beyond the core fields to be dynamically defined and implemented through the web application—such protocols must be added by hand to XNAT's XML-based security specification file.

Deploying XNAT

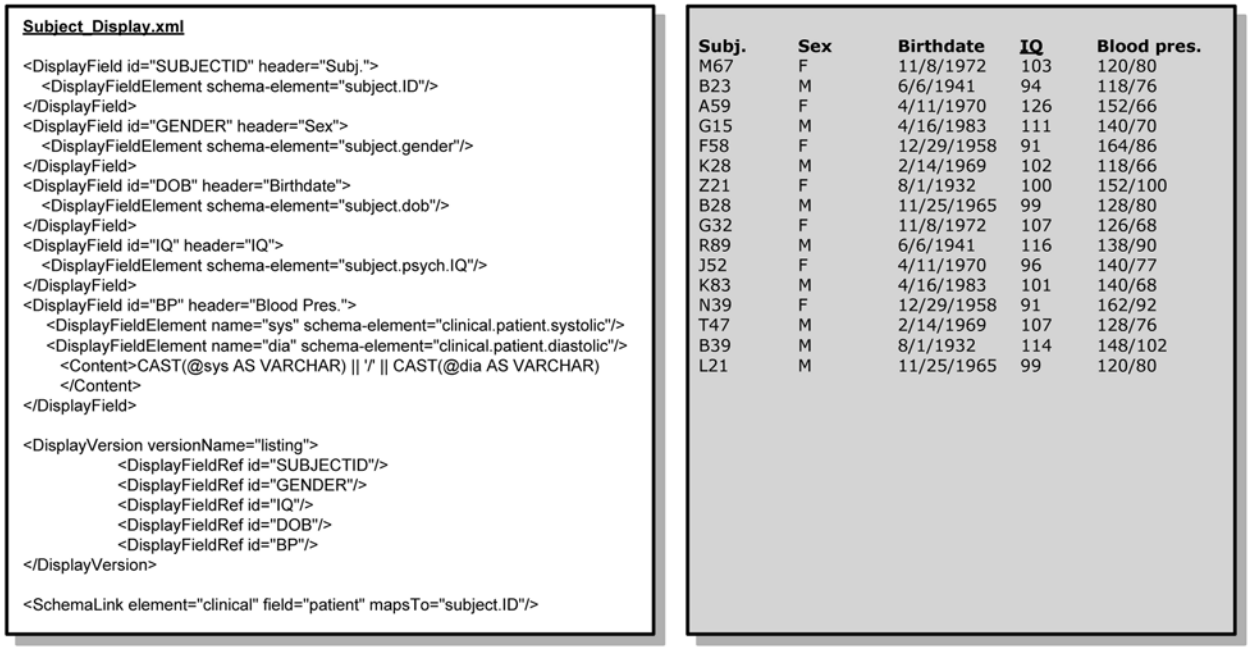
XNAT is an open source application freely available to the community under a BSD-style license with copyright held by Harvard University/HHMI. XNAT's extensible data model, modular design, and customizable user interface constitute a flexible platform that can be adapted to suit a wide range of research environments. A number of resources, including technical documentation, source code, an installation package, a mailing list, and a setup guide are available at www.xnat.org to support sites considering using XNAT.

Sites must complete the following steps to deploy an XNAT archive:

1. Read and accept the open source license;
2. Configure and install XNAT and its prerequisites;
3. Model site data in XML;
4. Modify local scripts and spreadsheets to generate XML;
5. Run XNAT's setup script;
6. Customize XNAT's user interface and feature set.

For data types not already in XNAT's catalog, the XML modeling and customization steps potentially require substantial commitment to implement. However, a number of XNAT features are available to simplify these tasks. The abstract types in XNAT's core XSD, for example, highly simplify modeling and integrating local data types. XNAT also includes generic template scripts for converting local spreadsheets and other documents to XML. The user interface, in particular, was designed to allow new data types to be added with minimal effort to its listings and reports. A listing for a new data type is implemented by creating a configuration document that describes which fields should be included in the listing and its associated search box, and how they should be formatted (Fig. 9). A report for a new data type is automatically generated by XNAT at setup time (Fig. 10). The generated report can be edited in an HTML editor to alter the layout and insert additional functionality. In addition, the backend code for generating the content of a report can be completely customized (Fig. 10).

XNAT also includes a programming interface (API) that provides developers with tools to implement more sophisticated custom functionality. The API includes both generic classes for accessing archive data and tailored classes generated specifically for the site's data types. Of the generic classes, *Search*, which implements complex searches of the archive, and *File*, which provides basic file information (e.g., location, size, and name), are the most useful.



Subject_Display.xml

```
<DisplayField id="SUBJECTID" header="Subj.">
   <DisplayFieldElement schema-element="subject.ID"/>
</DisplayField>
<DisplayField id="GENDER" header="Sex">
   <DisplayFieldElement schema-element="subject.gender"/>
</DisplayField>
<DisplayField id="DOB" header="Birthdate">
   <DisplayFieldElement schema-element="subject.dob"/>
</DisplayField>
<DisplayField id="IQ" header="IQ">
   <DisplayFieldElement schema-element="subject.psych.IQ"/>
</DisplayField>
<DisplayField id="BP" header="Blood Pres.">
    <DisplayFieldElement name="sys" schema-element="clinical.patient.systolic"/>
    <DisplayFieldElement name="dia" schema-element="clinical.patient.diastolic"/>
      <Content>CAST(@sys AS VARCHAR) || '/' || CAST(@dia AS VARCHAR)
      </Content>
</DisplayField>

<DisplayVersion versionName="listing">
            <DisplayFieldRef id="SUBJECTID"/>
            <DisplayFieldRef id="GENDER"/>
            <DisplayFieldRef id="IQ"/>
            <DisplayFieldRef id="DOB"/>
            <DisplayFieldRef id="BP"/>
</DisplayVersion>

<SchemaLink element="clinical" field="patient" mapsTo="subject.ID"/>
```

Subj.	Sex	Birthdate	IQ	Blood pres.
M67	F	11/8/1972	103	120/80
B23	M	6/6/1941	94	118/76
A59	F	4/11/1970	126	152/66
G15	M	4/16/1983	111	140/70
F58	F	12/29/1958	91	164/86
K28	M	2/14/1969	102	118/66
Z21	F	8/1/1932	100	152/100
B28	M	11/25/1965	99	128/80
G32	F	11/8/1972	107	126/68
R89	M	6/6/1941	116	138/90
J52	F	4/11/1970	96	140/77
K83	M	4/16/1983	101	140/68
N39	F	12/29/1958	91	162/92
T47	M	2/14/1969	107	128/76
B39	M	8/1/1932	114	148/102
L21	M	11/25/1965	99	120/80

Fig. 9. Creating custom listings requires appending to XNAT's configuration file (left, simplified for clarity). DisplayField elements are used to describe the contents of a particular field. The contents may correspond to a single field from the XML Schema (DOB) or it may be a derived from multiple fields (BP). The DisplayVersion element allows the fields and their sequence to be specified for a particular listing. Multiple DisplayVersion elements can be included to be used in different display contexts. The SchemaLink element allows related fields within or between schemas to be joined. Here, the "patient" field in the clinical element from example schema 2 to be joined to the subject element in schema 1. The resulting listing is schematized on the right.

XNATScreen_report_subject.java

```
public class XDATScreen_report_subject
(1)  extends SecureReport {

  public void finalProcessing(RunData data,
Context context) {

      //user & item are loaded
      //into context by parent class

(2)  Plot plot = createSubjectPlot();
     context.put(plot);
  }
}
```

(3)

XNATScreen_report_subject.vm

```
$page.setTitle("Subject Report")
$page.setLinkColor($ui.alink)
$page.setVlinkColor($ui.vlink)
<table width="100%" align="left">
 <tr><th><b>Subject Details</b></th></tr>
 <tr>
     <td>ID:</td>
     <td>$!item.getProperty("subject.ID")</td>
 </tr>
 <tr>
     <td>Gender:</td>
     <td>$!item.getProperty("subject.gender")
  </td>
 </tr>
 <tr>
(4)  <td>BP:</td>
     <td>$!item.getProperty("subject.BP")</td>
 </tr>
 </table>
(5)  <img src=$!plot.getUrl()>
```

(6)

Subject details

ID: b23
Gender: Male
BP: 118/76

Fig. 10. Creating custom reports requires developing java (left) and Velocity screen components (middle). (1) After data are retrieved from the database, they are placed in a Javabean-like XNAT object called *Item*. (2) Additional information can optionally be placed into Velocity's Context object for subsequent display. In this

The classes generated for each data type defined in the managed schemas include methods to get and set the field value for a particular instance of the data. The API in general is useful for implementing data type-specific actions. For example, a developer could use *Search* to find the records of some data type for subjects within a specified age range and then use the class generated for that data type to retrieve the values and generate a statistic for one of the fields.

Long-Term Archive Maintenance

Errors and discrepancies in data archives are inevitable. The goal of XNAT's quality-control tools is to minimize their occurrence, to identify them when they do occur, and to gracefully handle their correction. The XNAT workflow itself is largely a quality-control measure as it standardizes and enforces laboratory best practices for capturing, validating, and processing data. XNAT also maintains full data histories. Whenever a record is modified, XNAT creates a shadow record, indicating what data changed, when it was changed, who changed it, and why it was changed. This mechanism allows user's to determine what value a field had at any point in its history and to roll back changes that are found to be in error. In real-world use, data archives are seldom static. Projects, investigators, users, and hardware all come and go. XNAT's extensible XML data model is ideally suited to handle these dynamics. New data types can be incorporated into the archive by adding them to the managed schema and executing XNAT's *Update* tool. This tool generates the required database, middleware, and user interface code for the new types. XNAT's web-based administration tools can be used to deactivate users and obsolete records that are no longer in service. These items remain in the database (and can be reinstated if necessary) but are removed from archive activity.

The most vexing issue associated with maintaining an XNAT archiveand data repositories in general—is alterations to the data model. What happens when a field changes type? Or a field changes from a single measure to an unbounded set? Such changes are of particular concern in XNAT because it maintains separate XML and relational data models, which could get out of synch if not carefully managed. Simple changes such as the addition of a new field, can be handled in a straightforward manner by modifying the schema and running a corresponding "alter table" SQL command on the database. For more complex changes, we recommend adding a new version of the data type to the schema (using some nomenclature in the element name to reflect the version) and running XNAT's *Update* tool. Data associated with this new type will be stored in separate tables from the previous version, ensuring that the data types remain properly synchronized. When appropriate, the older data can be merged into the new version by downloading its XML, transforming it with an XSLT script to the new version, and storing the new XML. The original data can then be made obsolete in the archive.

Fig. 10. *(Continued)* example, a plot of subject data is created. (3) Turbine routes processing from the java class to the corresponding Velocity page. (4) The Velocity page lays out the particular data that will be displayed. Here, a number of subject fields are retrieved from the Item object and placed into an HTML table, including the blood pressure field defined in the subject display document (see Fig. 5). (5) The custom plot is displayed by retrieving the object that was placed into the Velocity context. (6) The Velocity engine parses the page to plain HTML and Turbine sends the document to the client. The resulting report is schematized on the right.

Implementation Details

XNAT's web application pages are written in Velocity,* an HTML templating language that allows server-side data to be integrated into the pages. A default CSS stylesheet† is included to enable a consistent look and feel for the overall site. It can be tailored to suit individual project aesthetics. The application itself runs within Jakarta Turbine§, a Java Servlet-based framework for streamlining web application development. User-submitted requests from the website (e.g., a search) flow through the web application as follows (Fig. 11). Turbine initially routes each request to a Java "Action" class that is specifically paired with the originating web page. XNAT's SearchAction class, for instance, handles submissions from search pages. The Action class is responsible for implementing the business logical to fulfill the submitted request. As the action completes, Turbine routes to an appropriate "Screen" class, which implements preparatory work before routing to the final web page. XNAT's SearchResultsScreen class, for example, sorts the data returned by SearchAction. XNAT provides a number of core Action and Screen classes to implement basic application features like searches, form validation, and download requests. Sites can add new functionality to the web application by developing custom Actions and Screens using XNAT's API.

XNAT relies on several additional externally developed applications that must be independently installed. All are open source and widely available. The web application has been deployed with Apache Tomcat¶ in our testing, but it should be deployable in any Java servlet-based application server. Installation and deployment of XNAT use Jakarta Maven**. XNAT uses PostgreSQL†† for its database backend. All of XNAT's command-line tools and other scripts are written in Perl with the inclusion of a number of widely available modules.

Discussion

An important component of XNAT's development is that it has been deployed for several years in an active production environment at Washington University. The archive currently stores more than 2,000 imaging sessions and 15,000 scans obtained from 1,500 subjects. Data include structural, functional, and diffusion tensor imaging sessions in raw form and anatomically aligned to each other and between subjects (Buckner et al., 2004). Typically, six to eight new imaging data sets are added weekly with eight or more users, accessing data daily. The core XNAT schema sufficiently captures the required demographic and MR acquisition variables. The schema was extended to capture a number of additional data types, including clinical assessments (Morris, 1993), neuropsychological tests (Rubin et al., 1998), white matter disease assessments (Burns et al., 2005), derived anatomic measures including whole brain (Fotenos et al., 2005) and subcortical region volumes (Fischl et al., 2004), and cerebrospinal fluid antibody assays (Fagan et al., 2000). An image viewer plug-in was written to display subcortical segmentation images overlaid on coregistered averaged T1-weighted images. Over the course of the archive's existence, additional data types have been added periodically by simply appending new types to the schema.

*http://jakarta.apache.org/velocity/.
†http://www.w3.org/Style/CSS/.
§http://jakarta.apache.org/turbine/.
¶http://jakarta.apache.org/tomcat/.
**http://maven.apache.org/.
††http://www.postgresql.org/.

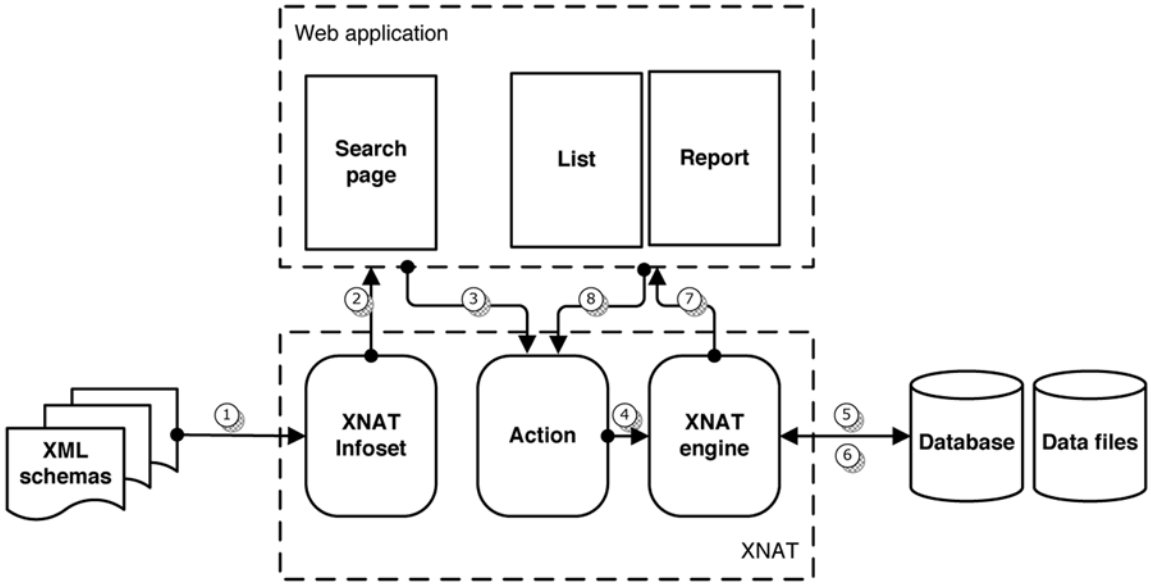


Fig. 11. Process flow of a web-based XNAT search. (1) When the web server is started, XNAT loads all of the specified XML Schema into an internal XML Infoset-like* representation. (2) Search fields based on the schema elements are incorporated into the search page. For the Subject element in Schema 1, fields for ID, lab, gender, and dob would be incorporated. (3) When the user submits a search, Turbine routes the request data to a generic XNAT Action class that handles all search submissions. (4) The action passes the search criteria into the XNAT engine, where it is parsed into an SQL query. (5) The query is run on the database. (6) A result set is passed back to the XNAT engine, where it is formatted for display. (7) If a single row is returned, it is presented as a report. Multiple rows are presented as a list. (8) The user can refine the results by resubmitting a search from a form embedded in the list.

The archive serves a number of roles for its user base. Most importantly, it acts as a central repository for the data. The archive's security, quality-control tools, and backup process ensure a reliable and robust archive. Second, the archive serves as a sharing mechanism for local and remote collaborators. XNAT's search tools facilitate the process of selecting appropriate subsets of the data from the larger resource, its security system allows the data that are accessible to these users to be appropriately restricted, and its download tools enable immediate distribution of the data. Third, the archive organizes previously diffuse data within a single database, providing a cleansed starting point for data mining operations. XNAT's Super Search tool was developed specifically to facilitate these operations. It is important to note that the archive is actively managed by domain-knowledgeable staff. Scripts were written to convert data sets from existing formats into XML, and large portions of legacy databases were migrated into the XNAT archive. An ongoing challenge is handling changes to the data model, including adding, removing, and redefining fields. Although regular communication with scientific personnel and clear annotation of the schemas are essential, further development of a strong versioning system in XNAT will be of great value. A final challenge worth

*http://www.w3.org/TR/xml-infoset/.

noting is that the XNAT workflow requires some effort to adopt. The probability of successfully deploying XNAT likely depends on a laboratory culture that buys into the long-term and broad value of its data.

To date, the goal of XNAT development has been to support within-institution archives. As more laboratories opt into using data management systems (XNAT or otherwise), developing mechanisms to enable communication and exchange between systems will be critical. The potential scenarios in which systems will need to interoperate include collaborating within a closed network of investigators, sharing resources within a set of federated database, and contributing data to a centralized warehouse (Martone et al., 2004; Van Horn et al., 2004; Toga, 2002). The likelihood of developing an interoperable infrastructure within the neuroimaging community to support these scenarios will depend largely on the technical solutions that are adopted. XNAT's focus on XML and XML Schema was deliberately chosen to simplify the transition from single-site archives to a multisite network. XML documents can be easily transported, validated, and deidentified. XNAT's base schema, for example, was designed to allow data providers to selectively include the experimental data that is relevant to the particular collaboration and to withhold sensitive data for a particular subject or acquisition. These benefits can be leveraged immediately by networks of close collaborators who adopt the XNAT schema (with or without the management infrastructure) as their exchange medium.

Acknowledgments

This work was supported by NIH-BIRN (U24 RR021382), the McDonnell Center for Higher Brain Function, and the Howard Hughes Medical Institute. We are grateful to Tracy Wang and Anna Cook-Linsenman for help in manuscript preparation and to anonymous reviewers for their informed and helpful suggestions.

References

Aarsten, A., Brugali, D., and Menga, G. (1996) Patterns for three-tier client/server applications. In Pattern Languages of Programs (PloP), Monticello, IL.

Amer-Yahia, S., Du, F., and Freire, J. (2004) A comprehensive solution to the XML-to-relational mapping problem. WIDM 4, 31–38.

Arenas, M. and Libkin, L. (2004) A normal form for XML documents. ACM Trans Database Syst. 29, 195–232.

Baru, C., Moore, R., Rajasekar, A., and Wan, M. (1998) The SDS Storage Resource Broker. Proceedings of CASCON'98 Conference, Toronto, Canada.

Buckner, R. L., Head, D., Parker, J., et al. (2004) A unified approach for morphometric and functional data analysis in young, old, and demented adults using automated atlas-based head size normalization: reliability and validation against manual measurement of total intracranial volume. Neuroimage 23, 724–738.

Bui, A. A., Weinger, G. S., Barretta, S. J., Dionisio, J. D., and Kangarloo, H. (2002) An XML gateway to patient data for medical research applications. Ann. NY Acad. Sci. 980, 236–246.

Burns, J. M., Church, J. A., Johnson, D. K., et al. (2005) White matter lesions are prevalent but differentially related with cognition in aging and early Alzheimer Disease. Arch. Neurol. 62, 1870–1876.

Carminati, B. and Ferrari, E. (2005) AC-XML Documents: Improving the performance of a web access control module. Proceedings of the tenth ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, pp. 67–76.

Chaudhuri, S., Chen, Z., Shim, K., and Wu, Y. (2005) Storing XML (with XSD) in SQL databases: Interplay of logical and physical designs. IEEE Trans. Knowledge Data Eng. 17, 1595–1609.

Dolin, R. H., Alschuler, L., Beebe, C., et al. (2001) The HL7 Clinical Document Architecture. J. Am. Med. Inform. Assoc. 8, 552–569.

Du, F., Amer-Yahia, S., and Freire, J. (2004) ShreX: Managing XML documents in relational databases. Proceedings of the 30th VLDB Conference, Toronto, Canada, pp. 1297–1300.

Fagan, A. M., Younkin, L. H., Morris, J. C., et al. (2000) Differences in the Abeta40/Abeta42 ratio associated with cerebrospinal fluid lipoproteins as a function of apolipoprotein E genotype. Ann. Neurol. 48, 201–210.

Fernandez, M., Kadiyska, Y., Suciu, D., Morishima, A., and Tan, W. (2002) SilkRoute: A framework for publishing relational data in XML. ACM Trans. Database Syst. 27, 438–493.

Fischl, B., Salat, D. H., van der Kouwe, A. J., et al. (2004) Sequence-independent segmentation of magnetic resonance images. Neuroimage 23, S69–S84.

Fotenos, A. F., Snyder, A. Z., Girton, L. E., Morris, J. C., and Buckner, R. L. (2005) Normative estimates of cross-sectional and longitudinal brain volume decline in aging and AD. Neurology 64, 1032–1039.

Gardner, D., Toga, A. W., and Ascoli, G. A. (2003) Towards effective and rewarding data sharing. Neuroinformatics 1(3), 289–295.

Hartmann, S. and Link, S. (2003) More Functional Dependencies for XML. Lect. Notes Comput. Sci. 2798, 355–369.

Hastings, S., Oster, S., Langella, S., et al. (2005) A grid-based image archival and analysis system. J. Am. Med. Inform. Assoc. 12, 286–295.

Ilioudis, C., Pangalos, G., and Vakali, A. (2001) Security model for XML data. Proceedings of the International Conference on Internet Computing, Las Vegas, NV, pp. 400–406.

Keator, D. B., Gadde, S., Grethe, J. S., Taylor, D. V., and Potkin, S. G., FIRST BIRN. (2006) A general XML schema and SPM toolbox for storage of neuro-imaging results and anatomical labels. Neuroinformatics 2, 199–212.

Krishnamurthy, R., Kaushik, R., and Naughton, J. F. (2003) XML-to-SQL Query Translation Literature: The State of the Art and Open Problems. Proceedings of the 1st International XML Database Symposium, Berlin, Germany, pp. 1–18.

Lee, M., Ling, T., and Low, W. (2002) Designing functional dependencies for XML, in EDBT 2002, Lecture Notes in Computer Science 2287, pp. 124–141.

Leuf, B. and Cunningham, W. (2001) The Wiki Way: Quick Collaboration on the Web. Addison-Wesley, Boston, MA.

Marenco, L., Tosches, T., Crasto, C., Shepherd, G., Miller, P. L., and Nadkarni, P. M. (2003) Achieving evolvable web-database bioscience applications using the EAV/CR framework: recent advances. J. Am. Med. Inform. Assoc. 10, 444–453.

Martone, M. E., Gupta, A., and Ellisman, M. H. (2004) E-neuroscience: challenges and triumphs in integrating distributed data from molecules to brains. Nat. Neurosci. 7, 467–472.

Morris, J. C. (1993) The Clinical Dementia Rating (CDR): current version and scoring rules. Neurology 43, 2412b–2414b.

Nadkarni, P. M., Marenco, L., Chen, R., Skoufos, E., Shephard, G., and Miller, P. (1999) Organization of heterogeneous scientific data using the EAV/CR representation. J. Am. Med. Inform. Assoc. 6, 478–493.

Nielsen, J. (2000) Designing Web Usability: The Practice of Simplicity. New Riders Publishing, Indianapolis.

Noy, N. F., Crubezy, M., Fergerson, R. W., et al. (2003) Protege-2000: an open-source ontology-development and knowledge-acquisition environment. AMIA Annu. Symp. Proc. 953.

Ozyurt, B. I., Wei, D., Keator, D. B., Potkin, S. G., Brown, G. G., and Grethe, J. S., (2004) Web-accessible clinical data management within an extensible neuroimaging database. Program Number 921.10. 2005 Abstract Viewer and Itinerary Planner. Society for Neuroscience, Washington, DC, 2005 Online.

Rubin, E. H., Storandt, M., Miller, J. P., et al. (1998) A prospective study of cognitive function and onset of dementia in cognitively healthy elders. Arch. Neurol. 55, 395–401.

Scheltens, P., Barkhof, F., Leys, D., et al. (1993) A semiquantative rating scale for the assessment of signal hyperintensities on magnetic resonance imaging. J. Neurol. Sci. 114, 7–12.

Shanmugasundaram, J., Gang, H., Tufte, K., Zhang, C., DeWitt, D. J., and Naughton, J. (1999) Relational databases for querying XML documents: limitations and opportunities. Proceedings of VLDB, Edinburgh, Scotland, pp. 302–314.

Toga, A. W. (2002) Neuroimage databases: the good, the bad and the ugly. Nat. Rev. Neurosci. 3, 302–309.

Van Horn, J. D., Grafton, S. T., Rockmore, D., and Gazzaniga, M. S. (2004) Sharing neuroimaging studies of human cognition. Nat. Neurosci. 7, 473.

Varlamis, I. and Vazirgiannis, M. (2001) Bridging XML-schema and relational databases: a system for generating and manipulating relational databases using valid XML documents. Proceedings of ACM Symposium on Document Engineering Atlanta, GA, pp. 105–114.

Wang, L., Riethoven, J. J., and Robinson, A. (2002) XEMBL: distributing EMBL data in XML format. Bioinformatics 18, 1147–1148.

Westbrook, J., Ito, N., Nakamura, H., Henrick, K., and Berman, H. M. (2005) PDBML: the representation of archival macromolecular structure data in XML. Bioinformatics 21, 988–992.

Zhao, L., Lee, C. P., and Hu, J. (2005) Generating XML schemas for DICOM structured reporting templates. J. Am. Med. Inform. Assoc. 12, 72–83.