Title
openSourcePACS: An extensible infrastructure for medical image management

Journal
IEEE Transactions on Information Technology in Biomedicine, 11(1)

ISSN
1089-7771

Authors
Bui, AAT
Morioka, C
Dionisio, JDN
et al.

Publication Date
2007

Peer reviewed

openSourcePACS: An Extensible Infrastructure for Medical Image Management

Alex A. T. Bui, *Member, IEEE*, Craig Morioka, John David N. Dionisio, David B. Johnson, Usha Sinha, Siamak Ardekani, Ricky K. Taira, Denise R. Aberle, Suzie El-Saden, and Hooshang Kangarloo

***Abstract*—The development of comprehensive picture archive and communication systems (PACS) has mainly been limited to proprietary developments by vendors, though a number of freely available software projects have addressed specific image management tasks. The openSourcePACS project aims to provide an open source, common foundation upon which not only can a basic PACS be readily implemented, but to also support the evolution of new PACS functionality through the development of novel imaging applications and services. openSourcePACS consists of four main software modules: 1) image order entry, which enables the ordering and tracking of structured image requisitions; 2) an agent-based image server framework that coordinates distributed image services including routing, image processing, and querying beyond the present digital image and communications in medicine (DICOM) capabilities; 3) an image viewer, supporting standard display and image manipulation tools, DICOM presentation states, and structured reporting; and 4) reporting and result dissemination, supplying web-based widgets for creating integrated reports. All components are implemented using Java to encourage cross-platform deployment. To demonstrate the usage of openSourcePACS, a preliminary application supporting primary care/specialist communication was developed and is described herein. Ultimately, the goal of openSourcePACS is to promote the wide-scale development and usage of PACS and imaging applications within academic and research communities.**

***Index Terms*—Biomedical imaging, image communication, radiology information system (RIS)/picture archiving and communication system (PACS) fusion, software libraries.**

I. Introduction

THE increasing presence of medical imaging within clinical care is evident: as an objective source of documentation and as a means to improve communication, imaging serves as a tenet of evidence-based medical practice [1]. Moreover, images contain important biomarkers, providing *in vivo* snapshots of anatomical and physiological processes. Thus, imaging also plays an emergent role in research, expanding the understanding of normal and disease states. With the growing estimates from tera- to petabytes of imaging data acquired annually, the effective management of imaging data is now a paramount necessity. Unlike standard medical data, however, images pose additional, distinctive management challenges. These problems are only amplified in consideration of the rapid developments in medical imaging—ranging from core technical investigations into new imaging modalities, to novel applications of existing imaging—all of which are capable of marshaling an evolution in healthcare; but such changes must be integrated into the clinical environment and research in order to be practical. Facilitating this objective requires an infrastructure that can support both the common requirements of today's imaging applications, while further serving as a springboard for future explorations.

openSourcePACS leverages earlier work by the authors in teleradiology and medical imaging information systems to provide an open source architecture for a picture archiving and communication system (PACS), creating a foundation for integrated imaging applications. Three considerations permeate the design of openSourcePACS: 1) cross-platform development and deployment, leveraging both Java and web-based technologies; 2) standards compliance, focusing on present and upcoming digital image and communications in medicine (DICOM) protocols; and 3) extensibility, allowing additional types of functionality (e.g., distributed image processing, non-DICOM querying) and standards (e.g., web services) to be incorporated in a unified manner by abstracting image-handling processes. The openSourcePACS framework loosely considers the multiple aspects of image management in terms of workflow, starting from the point of ordering an image, to its interpretation and use in documentation, through to the final distribution of results. Collectively, the software under openSourcePACS aim to provide a common, comprehensive suite of tools that can be adapted by developers to readily implement PACS-based functionality.

The remainder of this paper is organized as follows. Section II briefly provides background on related work in the area of PACS and imaging informatics, focusing on image management issues. Section III provides an overview of the openSourcePACS architecture, followed by the design issues and implementation details of each major module making up the system. An example application using openSourcePACS to support rapid communication between primary care physicians and specialists is described in Section IV. Finally, we conclude with a discussion on openSourcePACS issues, and future directions for this project.

Manuscript received September 30, 2005; revised February 23, 2006. This work was supported in part by the National Institutes of Health (NIH) Program Project Grant (PPG) PO1-EB00216 and in part by NIH Grant RO1-EB000362.

A. A. T. Bui, C. Morioka, U. Sinha, R. K. Taira, D. R. Aberle, S. El-Saden, and H. Kangarloo are with the UCLA Medical Imaging Informatics Group, Los Angeles, CA 90024 USA (e-mail: buia@mii.ucla.edu; morioka@mii.ucla.edu; usinha@mii.ucla.edu; rtaira@mii.ucla.edu; daberle@mednet.ucla.edu; sels@mednet.ucla.edu; hkangarloo@mii.ucla.edu).

J. D. N. Dionisio was with the UCLA Medical Imaging Informatics Group, Los Angeles, CA 90024 USA. He is now with Loyola Marymount University, Los Angeles, CA 90045-2659 USA (e-mail: dondi@lmu.edu).

D. B. Johnson was with the UCLA Medical Imaging Informatics Group, Los Angeles, CA 90024 USA. He is now with E! Networks, Los Angeles, CA 90036 (e-mail: dbjohnson@eentertainment.com).

S. Ardekani was with the UCLA Medical Imaging Informatics Group. He is now with The Johns Hopkins University, Baltimore, MD 21218 (e-mail: sardekani@jhu.edu).

Digital Object Identifier 10.1109/TITB.2006.879595

II. Background and Related Work

Several long-standing projects exist, some freely downloadable and released as open source packages, which address the basic functionality involved in a PACS.

1) *PACS servers and DICOM tools*. Several different PACS servers are available [2]–[6], running under Microsoft Windows and Linux operating systems; all offer basic DICOM functionality, such as query/retrieve and moving image studies. With the exception of [6], the storage backend is a relational database (e.g., MySQL, PostgreSQL), and many now offer web-based access to images and administration of the PACS. More sophisticated features include some image-processing abilities, and implementation of the DICOM printing standard. Largely, these systems are meant for single-server setup and smaller installations, in that distributed or larger PACS are not considered or handled efficiently. Separately, utilities for performing DICOM image management functions arose outside of these full PACS implementations: DCMTK [7], the Central Test Node software (CTN) [8], and dcm4che [9] are three popular software packages used by the above PACS software to execute core DICOM functions.
2) *Image viewers*. In addition to the above PACS servers, which often embed viewing software, there is a wide selection of stand-alone applications for displaying medical images [10]–[16]. For the most part, these software programs are geared towards the (diagnostic) review of imaging studies; hence, the main operations for manipulating images (i.e., window/level settings, cine, affine transforms) are central to all these programs. Many of these viewers also support the export of DICOM images to other graphical formats (e.g., joint photographic experts group; JPEG), annotation overlays, and real-time two-dimensional/three-dimensional (2-D/3-D) image analysis tools.

The choice of implementation language varies between projects, and includes C/C++, Perl, Java, and more platform-specific solutions, such as Microsoft ActiveX. It should be noted that the above-mentioned software packages are not an exhaustive list; and that these programs are in addition to the many commercial solutions such as those offered by major PACS vendors. Individually, each of these projects excels in their targeted application area; but none have yet yielded a complete solution, instead remaining focused on a specific part of the imaging workflow. For example, limited attention has been placed in formalizing the requisition of imaging studies across a community of physicians (e.g., clinicians outside of a primary institution); the integration of additional clinical information from other medical and research databases; or the dissemination of imaging results to broader audiences—all key matters for imaging to progress beyond its current state. Notably, [17] details a systematic workflow to support many of the tasks involved in a radiology enterprise, implemented using Java and a common object request broker architecture (CORBA); while some of the described objectives and technologies are similar to openSourcePACS, [17] primarily describes the application of software engineering principles to image management, rather than an open-source solution. Thus, the goal of openSourcePACS was to provide an inclusive open-source set of programs and modules supporting the multifaceted needs of imaging applications, including but not limited to the requisite functionality of PACS. openSourcePACS provides a framework wherein other existent imaging projects can be effectively tied together.

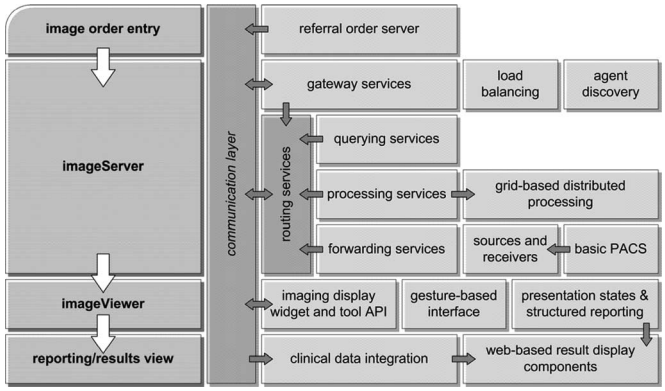


Fig. 1. Overall architecture for the openSourcePACS system, consisting of four different modules: 1) image order entry; 2) imageServer; 3) imageViewer; and 4) reporting/results viewing. A Java messaging system (JMS)-based communication layer ties together each of these parts. The submodules making up each module are depicted.

III. System Description and Architecture

openSourcePACS comprises four major modules that capture a generic imaging process model (Fig. 1), combining elements of conventional clinical workflow and documentation with support for research-oriented investigations; implementation details are provided in the subsequent sections. The first openSourcePACS component, *image order entry*, addresses the basic problem of requesting an image study for a given patient, thus initiating the process (e.g., referrals from a primary care physician; specialists ordering exams). Following image acquisition, the *imageServer* module performs routing, (ancillary) image processing, and archiving in a distributed network. *imageViewer* provides a standard image review interface for reading DICOM imaging studies, enabling the capture of key images and annotations through DICOM presentation states and structured reporting. Lastly, the flow of imaging is completed in the *reporting/result view module*, which facilitates further clinical data integration and communication of image findings. While many of the functions supported by openSourcePACS are derived from the DICOM standard, where possible the operations are abstracted to enable flexibility in image management. With the intent of supporting truly cross-platform development and deployment, all modules were built using Java and established frameworks.

To coordinate the different modules, an open-source application server, JBoss, was used: the different openSourcePACS modules were connected together under JBoss using a Java messaging system (JMS), supporting both point-to-point and publish/subscribe models for communication; and enterprise Java bean (EJB) components. The JBoss engine also provides for scalability by allowing developers to configure server clusters. Security in openSourcePACS is handled twofold: 1) standard

security and encryption methods (e.g., secure socket layer, SSL; authentication via login/password) was employed throughout the system and 2) the Java authentication and authorization service (JAAS) were further used to enforce role-based access control, attaching security requirements to the EJB methods.

A. Image Order Entry

The process of image management can be seen to start with the ordering of a study by a physician: fundamental information detailing the rationale for the study and the patient's presentation are formulated in this early stage. The paper-based analog, the imaging requisition order, plays this role in many radiology environments, stating the reason for exam and particular paths of inquiry to confirm (or to refute) a differential diagnosis. From this description (and through investigation of past clinical history), a radiologist can guide his/her response. However, two issues present themselves: 1) the contents of requisition orders are largely unstructured—most of the information is free-text (e.g., chief complaint information stated in the patient's own language)—and thus defies consistent coding (e.g., to ICD-9, International Classification of Disease, 9th Revision) and 2) in an increasingly distributed healthcare enterprise, the ordering of an exam requires knowledge of the imaging capacity of a given imaging clinic (e.g., how busy the clinic is, whether a given modality and protocol are available), and some experience in selecting the most appropriate imaging exam for the patient's problem. It is these two problems that the first module in openSourcePACS tackles by providing order entry components to coordinate the requisition and referral process.

openSourcePACS' image order entry consists of two parts: 1) a referral order server (ROS) that maintains (per imaging site) a database of available procedures and site-specific information (e.g., address, hours of operation, contact information, etc.) and 2) Struts-based user interface panels that tie together application logic with the database's contents. In addition, the ROS provides a user database template that can be extended to maintain a history of user (physician) orders and access rights to the system. The ROS database thus handles the first difficulty of providing site-specific information. The web-based interface logic embeds requisite fields for completing an imaging requisition—specifically, the patient's chief complaint and at least one reason for examination (RFE). The RFEs reflect the ordering physician's hypotheses, and are stated in terms of rule-in/rule-out diagnoses. Both types of fields can be specified as free-text—as in the traditional requisition form. Based on the free-text entry of the RFEs, the ROS can suggest potential diagnostic codes (e.g., systematized nomenclature of medicine (SNOMED); ICD-9), which in turn helps drive the selection of the procedure type and associates codified entries with the exam. To generate these codes, a statistical analysis of past records was performed [18], determining likely mappings. The suggestion system is designed to provide high recall (100%) over precision (60%), thus ensuring that the appropriate code will be in the suggestion list.

The use of specific diagnostic codes, in theory, permits some structuring of the information that can be carried throughout the imaging workflow. Using the web interface, a physician can thus login to a selected imaging site and order a study by providing patient information and entering the chief complaint and an RFE; the system then prints a "prescription" for the imaging study. Ideally, scheduling of the imaging exam should occur with the order entry process by the referring physician. However, though support for (web-based) scheduling and calendar protocols are ever-more common functions of many workflow management systems, most radiology information systems (RIS) are still "closed" and thus a generic solution was not feasible during the initial openSourcePACS implementation. Further avenues for linking scheduling into the order entry process are being investigated, including the use of HL7 messages.

B. imageServer

The second openSourcePACS module, *imageServer*, provides a logical operations layer for image management, handling the functions typically attributed to a PACS image acquisition gateway and router [19]: receiving and forwarding images; executing image postprocessing; archiving images to storage; and responding to queries for images. Though DICOM services (e.g., C-MOVE, C-STORE, C-FIND) can provide much of this base functionality, it is in this component where the need to abstract DICOM processes becomes evident in order to support heterogeneous computing environments, newer communication frameworks, and additional features (e.g., non-DICOM image implementations, advanced querying). The imageServer implementation follows an agent-based architecture, though the design is influenced by concepts from web services and distributed computing paradigms to provide for both extensibility and scalability (Fig. 2). Agents exist on a given system (e.g., a server, an imaging workstation, etc.) and create a dynamic network of published services. The services provided by an agent are classified into five different categories, with an agent supplying one or more of these services: gateway, routing, querying, processing, and source/receiving. As with the DICOM application entity (AE) title, each agent is assigned a unique identifier (UID) in the network, assigning an IP address and port number for activity. There are no constraints on the implementation of an agent other than inter-agent communication must be through defined extensible markup language (XML) messages. For example, the supplied base agents in openSourcePACS are implemented as a Java servlet, capable of parsing the XML text passed as parameters; equally, a simple object access protocol (SOAP) implementation could be used.

Two assumptions were made in the design of the imageServer: 1) that network capacity is steadily increasing in terms of speed and, hence, the sending of (large) image studies and multiple messages in a distributed data environment is acceptable and 2) that sufficient disk space is available in the network to store images as required. We acknowledge that while these statements are usually true, they must be balanced by the rising quantity and size of imaging studies.

1) Gateway Services: The image gateway service is the starting point for initiating image management, being the portal for commands, queries, and imaging data into the system, and the means of managing all other agents in a distributed fashion.

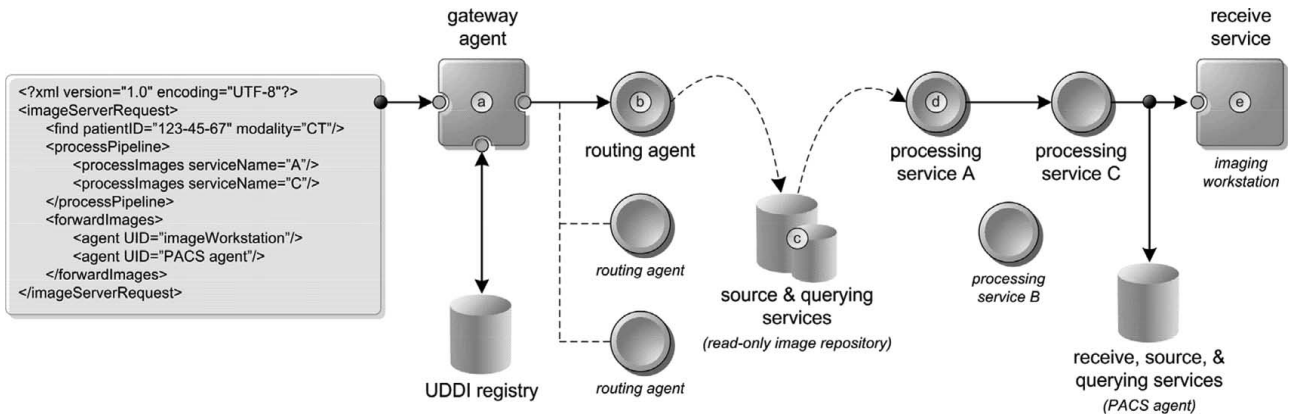


Fig. 2. Imageserver uses an agent-based framework to link together services for image management; a simple example is shown. (a) One agent acts as a gateway for receiving an XML document representing a sequence of commands; the gateway determines which agents with routing capabilities are available to handle the command based on current load estimates. (b) The routing agent receives and executes the commands, ascertaining what potential resources may be required, including accessing other services in the openSourcePACS network. (c) Agents specify their capabilities in terms of services, including querying, processing, source, and receiving. In the example XML request, the routing agent is directed to search for a given patient's past images with a specific modality criterion; more complex queries are enabled in openSourcePACS through a query-by-example syntax. (d) The discovered images are subsequently processed, accessing two additional agents (processing services A and C): the routing agent is responsible for establishing an imaging processing pipeline. (e) Finally, the images are forwarded to the requested receiving agents—in this case, an imaging workstation and a PACS.

All requests in openSourcePACS are sent to a gateway, which in turn redirects than to an available router for processing. The role of the gateway is summarized twofold.

1) *Agent discovery and status.* At the outset, a gateway is configured with an initial list of other agents providing services. The image gateway implements three methods for continually updating this set of agents: i) when new agents come online, they can directly inform the gateway of their services through a predefined XML message interchange; ii) a universal description, discovery, integration (UDDI) registry can be queried; and iii) a Rendezvous-based protocol can be used to actively find new agents within a subnet. The first approach provides an explicit means for agents to become part of an openSourcePACS network; the latter two web service-based and automated discovery protocols are more proactive techniques. A gateway service hence maintains knowledge of the state of the complete network, allowing it to appropriately assign requests to the available agents. The configuration of the network is passed on to other agents; when changes to the network occur, this information is immediately propagated. The time between searches can be altered, depending on the expected dynamic nature of the number of agents in the network. Notably, the methods that the gateway uses to search and continually update the status of the agents is configurable: depending on the setup, one or more of these protocols can be used and/or additional search methods can be used in the gateway service so long as the appropriate interface is implemented.
2) *Load balancing.* The image gateway service is additionally responsible for load balancing the requests across the set of routers. To make an informed decision, a periodic update of each known router's load is performed. When a new request is received by the gateway, a round-robin scheduling algorithm is used to select the next router that is below a given capacity threshold (in the situation where all routers are above the threshold, the next router in the list is used). By default, router load is a simple computation based on the number of outstanding requests that may be queued by the router, versus the router's configured capacity (the maximum queue size). More sophisticated metrics can be substituted on a per-agent basis; because the load calculation is reported by the agent itself, additional self-reporting statistics can be incorporated (e.g., average time to complete each request).

One agent serves as the gateway in an openSourcePACS environment; theoretically, while this "singleton" limitation can lead to a potential processing bottleneck, the actual request to the gateway itself is constrained in nature and thus high throughput is possible. Also, as the gateway service is implemented as a Java servlet, several instances can be running simultaneously (i.e., in different threads, but under the same virtual machine), thus enabling the gateway to scale accordingly. Basic fault tolerance can be established by creating a master gateway (e.g., through an election process among agents or outright configuration); when this service fails, another agent implementing gateway functionality can take over. Arguably, the image gateway service is comparable to a centralized job submission process used in grid computing (e.g., a submission node in Condor [20], a job manager in Globus [21]), with similar principles for load-balancing across the network; such frameworks are in fact currently being explored to replace the gateway service in openSourcePACS.

2) Receiving DICOM Images Via the Gateway: As described, the gateway service does not directly replace a classic PACS image acquisition gateway: in a DICOM environment, for example, scanners may push the images directly to a gateway without an intermediate command (or re-routing). openSourcePACS' gateway service implements a secondary process for handling DICOM C-STOREs. When the C-STORE

process is completed, the appropriate XML is automatically generated and sent to the servlet, which can then choose a router, as given earlier, to initiate processing.

3) Routing Services: Agents providing the routing services are the nexus of the imageServer, handling the requests passed on by the gateway. Received requests are automatically queued and processed individually in a first-in–first-out (FIFO) order. A request consists of one or more of the following commands in the following order, represented using XML (Fig. 2); the router parses the statements to determine the required resources and sequence of actions, returning the results to the request initiator. Specific services describing the processing of these requests are described in later sections.

1) *Image querying*. A key ability of a PACS is the retrieval of past image studies. Unfortunately, DICOM C-FIND and many PACS implementations provide very little flexibility in efficiently querying clinical archives: for example, clinical PACS queries are often driven by patient ID (e.g., prefetching) and as such, database indexing occurs by this field—but broader queries combining different potential information values (e.g., anatomy, modality, demographics, clinical outcome) cannot be readily answered. The escalating need for PACS to support such retrievals can be seen in the growing number of research-oriented image repositories that expand upon current clinical data models (e.g., Lung Image Database Consortium, LIDC [22], [23]; American College of Radiology Imaging Network, ACRIN [24], [25]). Indeed, as such "non-clinical" image databases become more common [26], it will not be possible to assume a given database schema for querying. Furthermore, the query language itself may not be "fixed"—while a relational database may be used to store DICOM study information, direct access (e.g., via the structured query language, SQL) may not be provided: concerns regarding clinical performance impact and security are common reasons. Yet eventually, more complex querying methods need to be supported. Hence, other (non-DICOM) querying methods may be available, and should be supported in a unified fashion. openSourcePACS therefore abstracts the querying process, allowing new query objects to be posited to a router and returning a set of image references.
2) *Image processing*. A request may require that a given imaging study be processed; for example, quantitative magnetic resonance (qMR) values may be extracted in postprocessing, or a 3-D reconstruction may be desired. The router attempts to match the requested processing algorithm with an available processing service (see later), resulting either in altered images (i.e., the raw image data is replaced with the result), or additional images being generated. Available processing services are published in an UDDI registry, declaring the expected inputs and outputs. Processing requests can consist of a chain of operations, establishing a pipeline for image manipulation such that the output of one operation serves as the input to a new image-processing step.
3) *Image forwarding*. The transmission of images to a specific location is the usual purpose for a router, ensuring that a study is sent to the appropriate archive and/or review workstations. The images sent to the openSourcePACS gateway and/or the results from querying and processing can be sent to any agent implementing receiving services. For example, a user querying for certain types of DICOM images and/or processing may have the results sent to a secondary server. Requests processed by a router can explicitly state to which computer(s) a set of images are sent. Alternatively, for traditional PACS routing in which acquired images are sent by default from the acquisition device to the gateway, a rule-based approach using DICOM header fields is used to select where the (clinical) images are dispatched.
4) *Image archiving*. Archiving is considered a special case of image forwarding requests, in that the series/studies are specifically sent for storage to a PACS; in openSourcePACS, archival services are represented by any agent providing querying, source, and retrieval services. Both generated images and series sent to the openSourcePACS gateway can be archived into a PACS.

In addition to these service definitions, XML messages to the router may specify "pass-through" information that is not parsed, but sent to the underlying service; this ability permits customization of a given service to use supplementary data.

Though an assumption of openSourcePACS is that communication networks are becoming faster and storage space is becoming cheaper, routing agents and the overall architecture attempts to minimize the amount of data transfer and replication; three techniques are employed to this end. 1) All imaging is assigned a uniform resource locator (URL) that is used in a planning phase by the router to minimize the number of "data hops" that must be made to execute a request. 2) Data can be selectively moved to an intermediate shared storage point. 3) A standard lossless data compression algorithm (e.g., gzip) is used to internally pass images between agents when necessary. The use of a URL to represent the imaging data allows the router to create a plan that can attempt to localize the usage of resources and agents: as each agent provides information on its location also in terms of a URL, similar IP addresses and subnets are preferentially selected over similarly available services available elsewhere in the openSourcePACS system (on the assumption that like IPs provide for some sense of proximity). For example, if two agents both implement the same image-processing algorithm, the agent that is "closer" to the data source will be used. Furthermore, by representing the data via a URL, data copying and the need for redundant storage may be reduced (at the cost of increased network traffic). Following this same idea, the routing agent can also select to temporarily copy the imaging data to shared network storage between agents, thereby facilitating dissemination and intermediary processing. Planning thus takes into account the sequence of actions represented by a given request (i.e., querying to retrieve images, ensuing processing, forwarding/archiving), determining which available agents and services can best accomplish the given task(s); semaphores are

implemented on each agent to provide resource locking. Collectively, these approaches attempt to strike a balance between limiting the amount of network traffic, reducing storage (via shared resources), and overall performance through localization. Given the complexity of this scheme, an important consideration is guaranteeing service; the base routing agent in openSourcePACS provides checks for service availability and transaction management (akin to standard database constructs). All errors in transmission or processing are automatically retried for a configured number of attempts before automatically logging an error and reporting a problem to the initial requestor.

4) Querying Services: Agents implementing querying services take as input a set of constraints that describe the target image series, and output a list of one or more imaging studies in terms of URLs; beyond this condition, the execution of a query is largely left to the agent. A uniform XML querying syntax is defined in openSourcePACS, following a query-by-example (QBE) paradigm that subsumes a standard DICOM C-FIND; each agent is accountable for parsing this XML into its own underlying query language. For example, openSourcePACS provides a default archive implementation with querying services that uses a relational database (PostgreSQL) to keep track of images in a file system. The XML query is transformed automatically into SQL statements that can be executed directly against this repository via JDBC (Java database connector). Equivalently, a web service querying implementation can be used to provide a mapping to a DICOM-compliant PACS. The free-form nature of the XML query allows new fields to be added to any given queryable agent: the agent is responsible for determining how best to map these fields to its own data source. On receipt of a query, the routing agent determines whether the request involves a query to specific image repositories, or is more general in nature. A query can opt to specify one or more specific agents to query by their UID; otherwise, a general query occurs, and all known queryable agents receive the request (determined by the router through the gateway's working configuration). The routing agent sends the request to each determined agent, and awaits receipt of a result set that is passed on to the next stage in the routing process. The authors' experience with teleradiology infrastructures and other PACS applications has highlighted the fact that a certain subset of queries involves discovery and retrieval of recently processed imaging studies for purposes of (internal) image management. To expedite these searches in openSourcePACS, the imageServer provides additional indexing of all image studies handled by the gateway and routers: a recent log of all forwarded imaging series (including those that are archived) are maintained centrally in a database; requests for these images are first searched for within this data store before being broadcast to all queryable agents.

5) Processing Services: Processing services present a "pipeline" for analytic procedures involving image data. For example, voxel-based analysis of diffusion tensor imaging (DTI) data between two population groups requires accurate warping to a common frame of reference—a nonstandard task that may be used for both clinical and research purposes. Accurate warping of DTI data from different subjects can involve multiple steps, including image de-noising, skull stripping, geometric distortion correction, and registration to create an atlas that converges to the centroid of the population being studied [27]. This progression of steps represents distinct processing stages from which individual images (and ancillary processing data) may be passed between a chain of algorithms. Fundamentally, an image pipeline may be abstracted to a directed acyclic graph (DAG) that captures a more complex processing architecture with multiple end points representing generation of several new images. Given the increased computational complexity associated with some image-processing methods, coupled with the higher resolution and number of image slices within a series, image processing can be overly time consuming. By way of illustration, creation of an atlas of ten subjects (image volumes at 1 mm^3 resolution with a matrix size of $256 \times 256 \times 128$ and two channels of data) can take approximately 12 h running on a standard Pentium-IV computer. openSourcePACS therefore takes advantage of the available grid computing systems to distribute the processing load when images can be handled individually (i.e., computations within a slice can be made independent of other images). Specifically, openSourcePACS allows Java-based algorithms to be spawned on a Condor-based grid, with the images and required code automatically copied locally to the participating grid clients. These services may also invoke non-Java algorithms through a wrapper, such as those provided in the insight toolkit (ITK) [28], which offers a growing library of C/C++ methods for advanced image processing; and the visualization toolkit (VTK) for 3-D image-processing [29]. Similar distributed image-processing approaches, described in [30]–[33], also detail pipeline and parallel imaging processing.

6) Sources and Receivers: The last set of services that an imageServer agent can declare represents sources and receivers of imaging data. Source services indicate that an agent is capable of transmitting an image to a specified location; receiving services indicate that an agent can handle the converse operation and store a transmitted image. With the generalization of imaging beyond DICOM standards, a problem arises in consideration of how images can be uniformly sent and received between different agents, as potential mismatches can occur (e.g., what if a source agent knows how to transmit via DICOM, but the target receiving agent only knows an HTTP-based protocol?). openSourcePACS solves this issue by providing a "lowest common denominator" protocol of DICOM C-MOVE/C-STORE as part of the base agent implementation, allowing all agents to communicate using this method, should a mismatch happen. However, by default, the routing agent will use the preferred communication protocol specified by source and receiving agents involved in a request.

7) Basic PACS Server Implementation: With the preceding concepts, in openSourcePACS a conventional images archive is therefore defined as an agent providing both sources and receiver services, coupled with querying. A rudimentary PACS agent is implemented under openSourcePACS, supporting both direct DICOM communications and XML messages via the imageServer framework. All the images received by this agent are stored using the underlying file system, organized in a hierarchical scheme by DICOM study/series UIDs to provide ready lookup. The agent uses a PostgreSQL database to save

image file locations and DICOM header information, making it searchable across all fields via SQL; by default, the tables are indexed by patient ID, modality, and anatomy. Interestingly, because of the abstracted image operations, this agent can support multiple different DICOM implementations via mapping: for example, [9], [34], and the authors' own DICOM implementations can be "plugged in" to this agent and dynamically instantiated via Java reflection methods. Correspondingly, the XML messages for executing DICOM C-MOVE, C-STORE, and C-FIND commands are mapped to an underlying DICOM implementation.

C. imageViewer

The third openSourcePACS module, *imageViewer*, provides a graphical user interface (GUI) for image display. Data transfer between imageServer and imageViewer is supported in three ways: 1) via standard DICOM C-MOVE and C-STORE processes, thus copying images to the local file system for viewing; 2) via network file shares; and 3) via Java-specific (compressed) data streams that bypass DICOM and high-level protocols (e.g., HTTP) to provide low-level data transfer at more optimal speeds, copying the data to storage local to the imageViewer application. When possible, the latter of the methods is used to provide faster on-demand access to images.

The principles behind the imageViewer were threefold: 1) the development of an extensible application programmer's interface (API), providing standard manipulation and layout tools; 2) the creation of new methods for manipulating images using gesture-based interactions; and 3) the support for image documentation processes used in many common clinical, research, and educational imaging-based applications. imageViewer integrates querying and receiver agents from the imageServer to enable access to imaging studies.

1) Application Programmer's Interface: The imageViewer uses the Java advanced imaging (JAI) package [35] for optimized low-level manipulation on raw (DICOM) imaging data and memory management through image caching; in addition, fast file input/output (I/O) is available. The choice of using JAI was primarily to enable cross-platform display beyond that of the chief graphic algorithms in Java (e.g., Java2D), with the added benefit of: 1) platform-specific native libraries (Microsoft Windows, Sun Solaris) to enhance operational speed; 2) access to commonly employed data structures for image processing [e.g., histograms, regions of interest (ROI)]; and 3) the ability to import and export images to a range of popular graphical formats.

The present imageViewer class hierarchy roughly corresponds to a model-view-controller (MVC) paradigm. At the heart of the model components are Java classes that capture the (raw) imaging data and ancillary information: a DICOM-specific image reading class was implemented via JAI, allowing for on-demand image reading. Following the standard DICOM data model, groups of images are collected into a single series, which in turn aggregated to make up a study. imageViewer view classes render the DICOM images into image panels (extending the Java Swing JPanel class). Panels can be clustered together into a group, representing a single series; and grouped instances can be recursively spatially (i.e., graphically) clustered, capturing a study or other arbitrary type of image collection that is visually manipulated as a unit. More complicated renderings of a sequence of images can also be handled using this hierarchy. For example, an image panel can be associated with a study, using Java3D to render a 3-D texture-based visualization of the dataset. [36] presents a similar method for volume rendering.

Associated with each view is a rendering pipeline, borrowing from the JAI paradigm. The base rendering pipeline consists of the usual affine transformations and color maps that may be applied to a DICOM image (window/level, rotations/flips, scaling, translations); however, it is configurable such that new operations can be inserted (or substituted) into the sequence by referencing new Java classes. To illustrate, a published computed radiography (CR) edge-sharpening algorithm [37] was easily implemented and inserted at the beginning of the pipeline; likewise, algorithms for optimal window/level settings have been developed to replace presets [38]. Benefits in the pipeline architecture include JAI chains and delayed processing: changes to the parameters of an operation within the sequence only require re-rendering of the image in subsequent steps—a complete re-rendering of the entire image is only necessary if the first operation in the pipeline is affected. Moreover, the rendering steps of the pipeline only occur if the image is presently being viewed (e.g., if an image in a series is not displayed, the image processing is delayed until needed), thereby providing faster response time. Lastly, the controller follows mouse events at each success view level, propagating user input/responses to the appropriate view/model; each view grouping can be separately controlled and/or selectively joined/separated to perform an operation.

Atop these core components, the imageViewer provides for image tools, configurable user interface components, and layout mechanisms for images, completing the set of functions needed to construct a basic imaging workstation (Fig. 3). The rendering pipeline illustrates several of the elementary operations available as tools to the user: manual window-level manipulation (or, invocation of presets, automated window/level selection) and inverse color mapping; free-form rotations and horizontal/vertical flips; magnification, including a "magic lens" function; and translation (i.e., image panning). Additional tools include cine, annotations, and undo/redo functions. All of these actions can be associated with user-defined key mappings, graphical toolbars, and menus. With the complexity of image grouping/clustering supplied by the underlying view and model, imageViewer also supports dynamic changes in layout that can combine multiple panel sizes/scaling behaviors; new layouts can be defined using XML.

Additional support for reading non-DICOM medical images, such as the increasingly popular Analyze data format and raw-data images are also a part of the imageViewer package. Other conventional image types (e.g., Tagged Image File Format, TIFF) are supported directly through JAI.

2) Gesture-Based Interaction: Our observation of PACS imaging workstations in the past several years is that most employ standard user interface paradigms for interaction, including menu bars, tool palettes, and keyboard strokes. However, efficiency and functionality are sometimes contradictory objectives

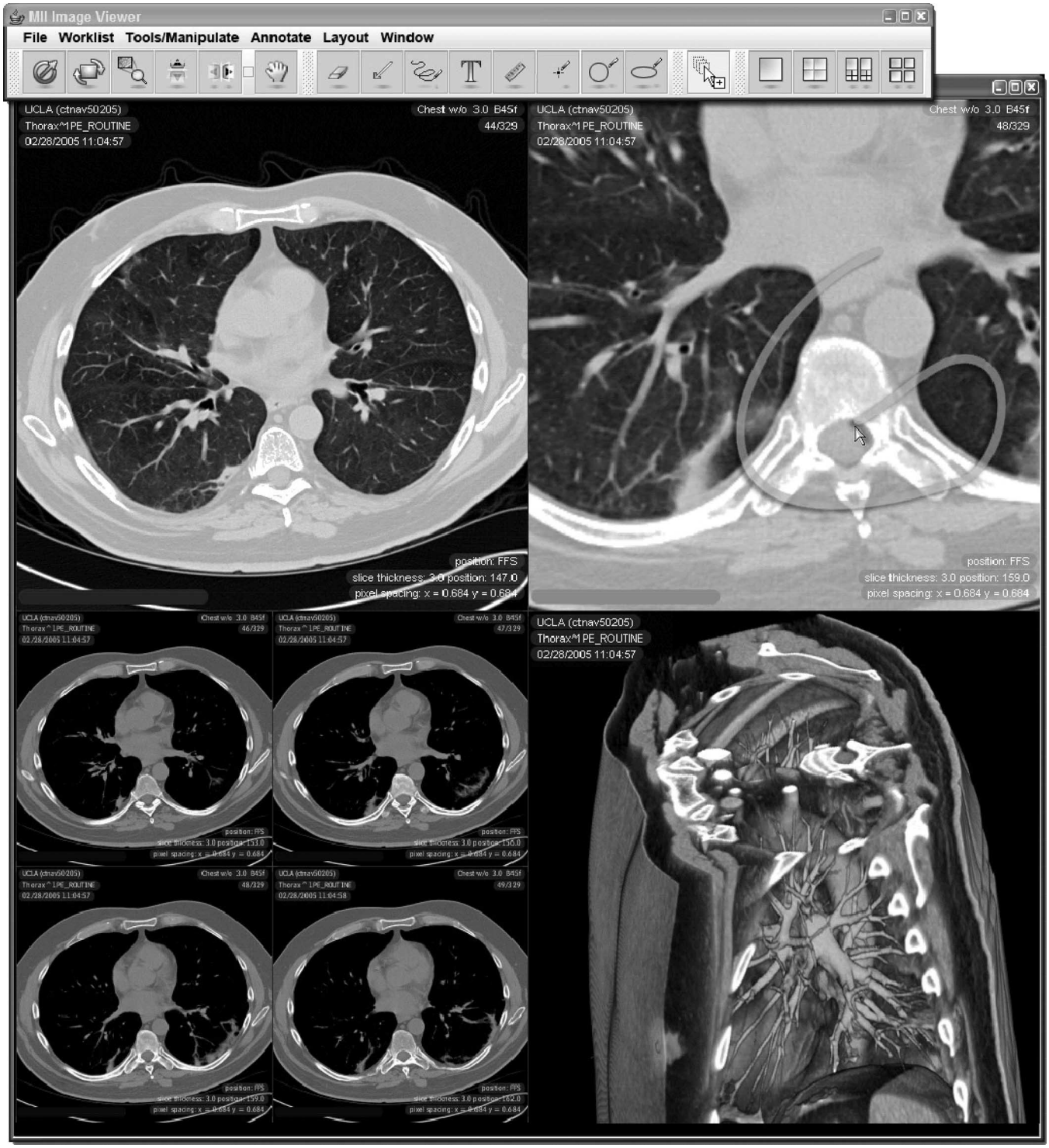


Fig. 3. Basic imageViewer application, allowing for the display of DICOM images using Java advanced imaging (JAI). Standard image workstation tools, including annotations, are available through the imageViewer. In addition to the traditional menu- and toolbar-based user interface widgets, the upper-right panel shows the use of the mouse gesture interface to select a command; in this case, the user has drawn a spiral-shape to invoke the DICOM presentation selection mode in order to capture a key image. The bottom-right panel in the display shows a texture-based 3-D volume rendering of the image series. Patient identifying information in this figure has been masked.

for a well-designed image-viewing workstation. Thus, a goal of the imageViewer was to create simplified user interfaces focusing on imaging. In addition to providing standard GUI widgets in imageViewer, an alterative method of user interaction, gesture recognition, is implemented, with the goal of providing smoother, faster manipulation of images in a manner that is not as disruptive to the primary task of reading/interpreting a study [39]. The user can draw a predefined shape (i.e., gesture) anywhere on the screen to invoke a command: using a mouse (or other input device), a graphical representation of the gesture

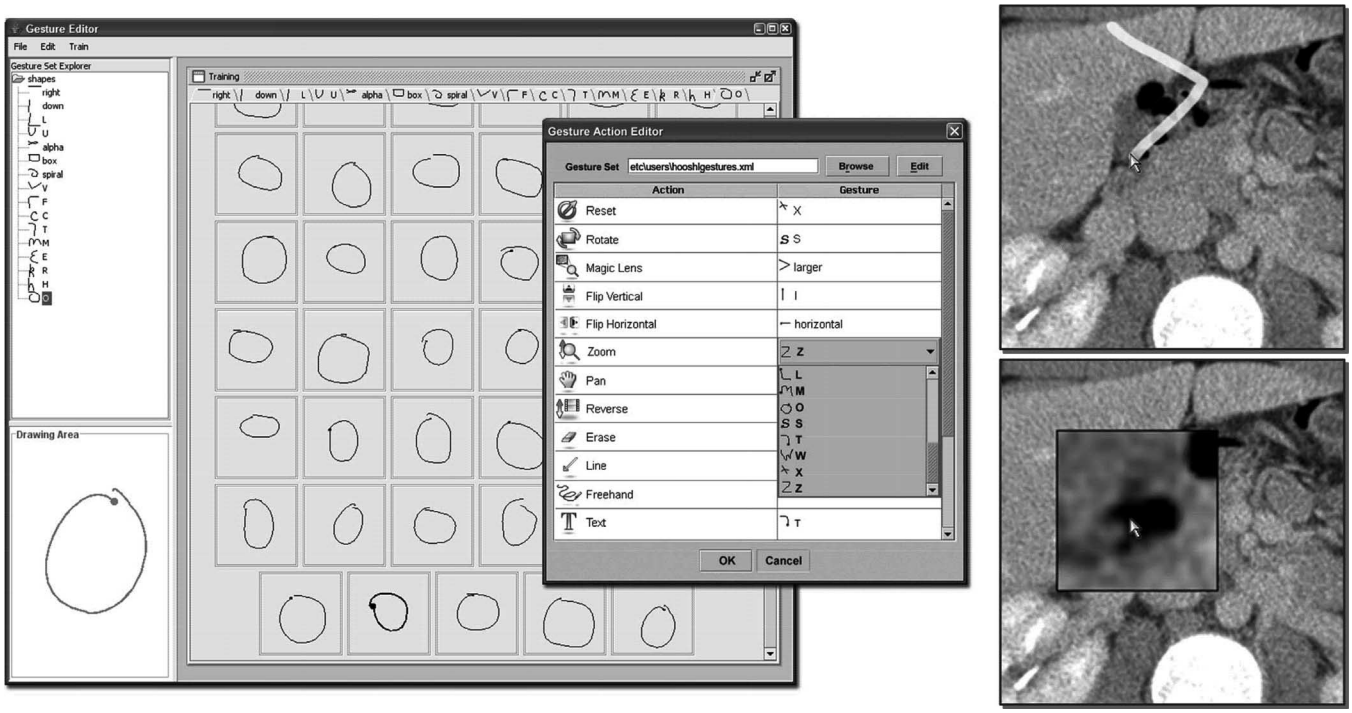


Fig. 4. Left panel shows the Java application that is used to store demonstrative shapes that the classifier is trained on (in this example, an "O"). The trained classifier learns to discriminate amongst a given set of shapes. In the imageViewer, the user can then associate a given action with a gesture (*middle panel*) as part of the configuration process. Drawing a given gesture (a ">") on the screen (*right panel, top*) invokes the corresponding command (*right panel, bottom;* in this example, the magic lens function).

is drawn on-screen in real-time, overlaying the current display (Fig. 3, top right). This functionality allows for "uncluttering" of the display, removing GUI widgets and preserving space for the key focal point of any image review workstation—the images. Gesture recognition itself has been explored in other application contexts, including: handwriting recognition, personal device assistants (PDAs), web browser manipulation, computer drawing programs, and computer games [40]–[44]. The set of gestures is configurable such that a gesture from a training set can be dynamically assigned to one of a set of common image manipulation functions (e.g., window-level setting, rotation, image layout settings, measurement, annotation), allowing the user to tailor the display and the functionality of the application to his/her preferences.

imageViewer's gesture-recognition engine combines two basic algorithms to classify 2-D "strokes" drawn by a user: a linear interpolation algorithm and adaptation of [45] are used. The first algorithm is a straight-line interpolation that derives a simple signature for linear-type shapes (e.g., a straight line, "L" shapes, "V" shapes), analyzing the predominant direction of sequential line segments to formulate a pattern based on eight directions. Based on this decomposition, the directional signature is compared against known patterns to attempt to classify the gesture, associating it with an imageViewer action. Failing this first classification method, a linear classifier algorithm is employed, using different shape features to characterize a best-fit (e.g., bounding box angle, end cosine, end length, time to draw, etc.) [45]. The gesture engine is trainable, allowing a user to select a (sub)set of the features to use in training; and to create/supply a set of exemplar gestures. Fig. 4 shows an example of how gestures are trained, linked to a given command in the imageViewer, and executed. For example, a circular mouse motion may represent invocation of a rotation tool. Likewise, a box-like shape may indicate a region for magnification.

A pilot study involving 15 individuals (radiologists, physicians, novice users) examining the efficacy of this gesture-based interface was completed. This preliminary testing showed an improvement of 20%–60% in the total time required to select and invoke an image-manipulation tool using the gesture-based interface (as compared to a toolbar and menu-based interface), even with only 90% accuracy in initial gesture classification by the engine; further details on this gesture-based approach and more formal evaluation will be reported in future work.

3) Supporting Documentation: An important role for the imageViewer is to facilitate documentation processes. In considering the role of the radiologist reviewing images (or any other image expert), often specific images and regions are used as the basis for documenting disease state—distilling a large imaging dataset into a few sentinel slices. Capturing how the expert views these images and provides interpretation is a critical part of the communication process, as this information is often relayed to other clinicians to guide treatment [1]. Directly supporting these tasks, DICOM has come to include presentation states (PS) [46] and structured reporting (SR) [47]; imageViewer implements both of these standards. A suite of tools are implemented in the imageViewer, allowing a user to annotate an image with drawing tools (lines, arrows, boxes, circles, polygons, freehand) and text. In addition, pixel and ROI value readouts can be imprinted on

the image. These annotations are overlaid on the base DICOM image, and can be edited and/or deleted. With the current image state (i.e., affine transforms, window/level), the annotations can be explicitly saved as a DICOM PS instance associated with the original study; review of the images by another party can thus access the viewing state and annotations. Additional free-text comments not shown inline with the image (e.g., a label or short description) can also be added to an image as part of a DICOM SR object; particularly, information on the findings present on the image can be elaborated upon. Dependent on the application using the imageViewer API, the DICOM PS and SR objects can be stored into a PACS with the archived image study.

D. Reporting/Result Viewing

The last openSourcePACS module handles the tasks of integrating and reporting results to a broader audience than imageViewer users. Note that here, "reporting" does not refer to the process in clinical image review (e.g., a radiologist dictating image findings to generate a document)—but rather, to the dissemination of image findings. Two issues are considered in openSourcePACS.

1) *Access to other (non-imaging) clinical data sources.* In order for image findings to be properly understood, the appropriate clinical context must be provided. Stand-alone PACS, while providing some insights for a given patient through past imaging studies, does not give a complete understanding of a patient's state—further objective information, such as from pathology, laboratory, and other data sources are often needed for proper assessment. In fact, the integration of PACS with clinical repositories is a goal of the Integrated Healthcare Enterprise (IHE) [48], which demonstrates workflow management and data sharing between clinical systems (e.g., hospital information systems, HIS; RIS; etc.); in a similar vein is the National Health Information Infrastructure initiative in the United States, which encourages data interchange among the array of clinical systems [49]. To this end, openSourcePACS leverages the authors' *DataServer* system [50], which enables real-time querying and retrieval of information from clinical and research databases. DataServer facilitates access by creating a single, uniform XML representation, abstracting underlying data sources and access mechanisms; like openSourcePACS, its objective is to enable generalized access to data, irrespective of the querying protocol and providing a mapping between disparate data representations. Given that each potential data source may implement different access protocols (e.g., HL7 versus a relational database or SOAP object), DataServer provides a library of methods for tailoring connections, transforming an XML-based query into a specified format (e.g., SQL) and in turn, translating the results into a given (XML) data representation. Querying DataServer itself occurs via a servlet, much like the querying component of the imageServer. Supplementary Java classes are included in openSourcePACS to execute DataServer queries, allowing for the logical grouping of information around specified clinical attributes (e.g., data source, report types, etc.).
2) *Integrated web-based visualization of results.* Given access to the DICOM presentations states, structured reporting, and other clinical information, the final aspect of openSourcePACS is to construct a comprehensive display that puts the image findings into perspective. Specifically, the goal of this interface is to facilitate the distribution of results, such as to a clinician (e.g., the referring physician), students (e.g., medical students, residents), or researchers. Construction of such GUIs and the integrative data model unifying such data is chiefly application dependent; as such, openSourcePACS' strategy is to provide a set of web-based widgets that can be re-used in multiple future applications. Two implementation approaches have been used. First, an asynchronous JavaScript and XML (AJAX) framework was used to create dynamic HTML components that can be adapted using stylesheets. Customized graphics [e.g., exporting the DICOM images and associated presentation states to a static portable network graphics (PNG) format] are generated server-side and made accessible via a secure web server. However, the multitude of different web browsers and nuances in rendering implementations makes identical cross-browser rendering challenging (e.g., Microsoft Internet Explorer version 6, for instance, does not properly support PNG transparency). To this end, a second set of GUI components that can integrate the clinical data, imaging, and DICOM objects was developed using the multimedia scripting language, Laszlo, which can create Macromedia Flash presentations [51]. Laszlo itself provides basic GUI components, much akin to those found in Java's Swing package (e.g., windows, menus, scrollable views, tabbed panes, buttons, text/graphic canvases). Using these core components, openSourcePACS provides demographic, DICOM PS/SR, document (e.g., clinical reports), and laboratory widgets.

Though the primary intent of this last module is to support the creation of web-based interfaces, it also serves to make available the results of the review process (via DICOM PS/SR) in other nongraphical formats, such as XML [52], and thus accessible in turn through DataServer.

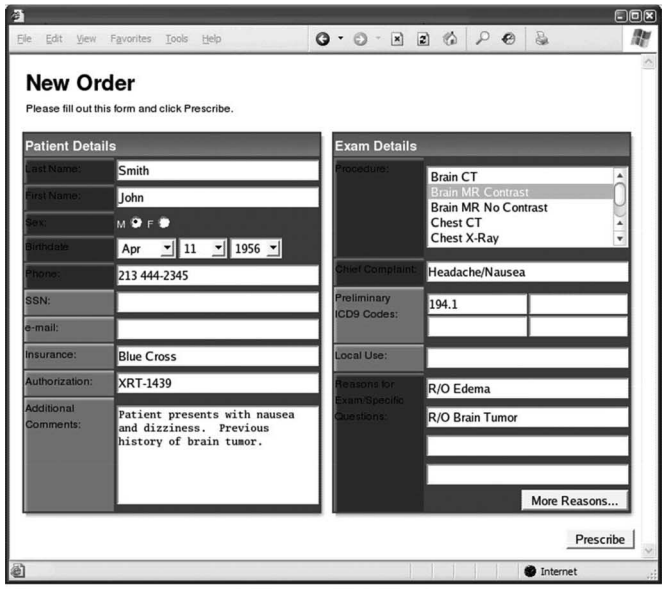
IV. Example Application: Rapid Primary Care/Specialist Communication

The functionality encompassed by openSourcePACS lends itself to a variety of different applications, including the development of imaging-based teaching files, image-analysis workstations, and web-based referrals/teleradiology. To demonstrate its utility, an application supporting primary care/specialist communication was prototyped along the lines of a "wet-read." In many hospital environments, radiologists are often asked to perform review of imaging studies that require immediate interpretation for the sake of ruling in/out a given differential diagnosis; this process is known as a preliminary read or "wet-read" (in film-based environments, radiologists were asked to interpret urgent cases on developing films before drying—hence

the term "wet-read"). The communication between the requesting physician and the radiologist in a preliminary read is often terse, with a reason for examination and targeted "yes/no questions" that need to be answered (e.g., patient has dyspnea; does he have pneumonia?). It is understood that the radiologist's response in this situation is only an initial impression. This model is readily extended to enable community-based physicians (e.g., primary care physicians) to request preliminary reads by experts (e.g., subspecialist radiologists at an academic medical center) with rapid turn-around response as soon as a patient completes a study at an imaging center. This problem has been considered in the context of urgent care [53]. Other related work in this area includes web-based teaching files, which provide the tools to capture images with annotations to effectively create presentation states for instructional purposes [54]–[56]; however, order entry and result views, as in openSourcePACS, are typically not handled in these applications.

All four modules of the openSourcePACS implementation were used in this project, building a web-based referral, review, and reporting system.

1) *Image order entry module*. Minimal customization was required to adapt the image order entry system to this application: only the list of available imaging procedures needed to be configured, with information on the target imaging center to produce an image requisition. The user database template was customized to handle referring physicians. Additionally, the ROS was augmented to maintain a list of completed and outstanding requests per user, linking to study results when finished. Web pages supporting the login process and for showing a history of requested studies were created (Fig. 5). The existing implementation does not attempt to solve the problem of scheduling the imaging exam, leaving this step as a manual process.
2) *imageServer.* Given the loose coupling of order entry to image acquisition and in the absence of a requisition ID, it is necessary to match the studies to the initial request. The DICOM image-receiving process that communicates with the gateway agent was combined with a reconciliation module. A matching mechanism based on the data fields for first name, last name, gender, date of birth, and referring physician are pulled from the DICOM header and compared against the ROS entries. Simple string matching is initially performed; however, failing such analysis, more heuristic approaches are taken, as suggested by [57]: name variants (e.g., nicknames) are referenced, and variations in the date (e.g., due to typos, for example) can be suggestive of potential matches. The study date is also compared to the ROS order date (the study should be completed after the order). Finally, further matching is done by comparing states of anatomy and contrast in the DICOM header descriptions versus those of the ROS procedure name. An aggregate score, based on the number of exact matches, potential (variant) matches, and nonreconcilable fields, is then computed to indicate the overall likelihood of a match. A Swing-based GUI was created to allow manual inspection and to perform any correction. Once a match is validated, both sets of information are forwarded via the gateway to a specific imaging workstation designated for preliminary reads.
3) *imageViewer*. The imageViewer API was used to create a rudimentary review workstation. A worklist was attached to the main display, indicating all available patient data on the workstation with any outstanding requests. A response module was also added to facilitate the preliminary read process: radiologists reviewing the study were required to explicitly answer each of the differential diagnosis questions posited by the referring physician (as



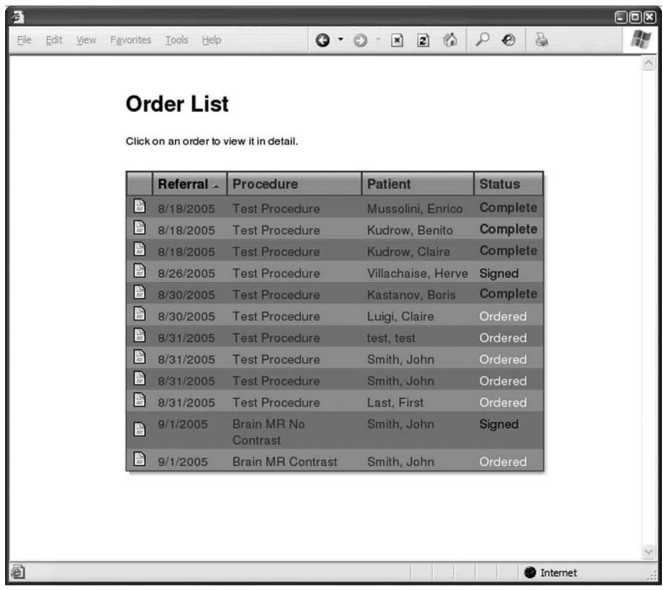


Fig. 5. Web pages were developed to customize the image order entry component of openSourcePACS for the primary care/specialist loop application. The "New order" page shows the basic openSourcePACS ROS, allowing an authorized physician to request a new imaging study; required fields include the reason for exam and the patient's chief complaint, in order to create specific questions for the radiologist. The "Order list" page is a new page that shows a physician his/her history of image orders and the status of the preliminary read; clicking on an item brings up the corresponding information and when completed, the results (see Fig. 6).

Fig. 6. Web-based results view display created for the openSourcePACS primary care/specialist communication loop application (patient information is fictitious). When a radiologist completes the preliminary review, the system generates a website that pulls together the DICOM presentation states (PS) and comments in the associated DICOM structured reporting object. Patient demographics and study information are presented at the top of the interface. The reasons for exam and the radiologist's answers are given in the left-side panel with icons (in this case, a brain tumor was found, thus indicated by a check mark; edema was not found, indicated by "X"; and additional findings are given). Selecting a reason for exam on the left, calls up the image slices chosen by the radiologist. The DICOM PS is shown in the middle of the display. Additionally, related image slices selected by the radiologist can be seen by choosing from the thumbnail views on the right.

given in the original order entry). Evidence in response to each question uses the DICOM presentation state, capturing the specific image view and annotations. A DICOM SR object was instantiated for each question, consisting of a single response (positive finding, negative finding, uncertain), and one or more presentation states (i.e., annotated images). Incidental findings and additional free-text comments can be also be added, and are stored as part of the SR instance. When all questions are answered, the workstation marks the study as complete, and forwards the completed dataset to the reporting module.

4) *Reporting*. Finally, on receipt of a completed preliminary read, the reporting module was used to generate a result view that integrated the original request for examination, differential diagnosis, and image findings from the radiologist (Fig. 6). The website and its graphical components are automatically published. Further code was added to inform the ROS of the completed result status (and thus the referring physician), adding a link to directly access the web page, and thus completing the primary care/specialist communication loop.

This application directly facilitates the practice of evidence-based radiology [58]. Presently, this system is being adapted at the authors' institution to provide an integrated oncology consultation (e.g., lung cancer), incorporating digital pathology slides and findings, laboratory, and other clinical documents.

Also, this openSourcePACS application is being evaluated by a private company at a site in Melbourne, FL, in support of communication between a primary care center, a dedicated imaging facility, and local community physicians [59]. The imaging facility digitally captures CT, MR, and CR imaging. Given the relatively low volume of imaging with an average of

45 studies/day (relative, at least, to an academic medical center), the entire system was implemented on a single server, running both the Apache Server and Postgres database (dual 2.4-GHz Intel Xeon processor system, 1-GB RAM, Redhat Linux 9, 34-GB hard drive), and integrated with an existent local PACS (NeuroStar). At the time of writing, this system has been piloted for over three months, and now references in excess of 2500 new DICOM imaging studies with associated (annotated) reports. Initial feedback from users is being incorporated to refine the system before rollout.

V. Discussion

openSourcePACS aims to provide a solid foundation upon which imaging- and PACS-based applications can be built. The majority of the code, including the primary care/specialist communication loop application, has already been released to the open source community under the Gnu Lesser General Public License (LGPL) [60]; the remainder of the code base is presently being prepared for general release. Releasing source code is a necessary but not a sufficient condition for successful dissemination of software: potential users must have a means to receive additional assistance and guidance [61], [62]. Moreover, a motivation behind open source is to sustain development and interest in a project so that it grows beyond the original vision. To this end, a full and active website with mailing lists, API and documentation (e.g., Javadocs), source code control, and forums (e.g., a wiki) were implemented for public developers and users.[1] Complete binary and source code distributions can be freely downloaded. Since a formal presentation at the Radiological Society of North America (RSNA) in November 2005, a conservative estimate of approximately 50–60 downloads have occurred per month[2] and a small group of users ($\sim$20) outside of the initial development group has joined the mailing lists. We anticipate that this group will grow in size as more components of openSourcePACS are released and further development occurs.

More significantly, support for such a project and its clinical deployment requires a large degree of technical maintenance that may be beyond academic boundaries and what is found in open source forums. For example, problems arising from software bugs may require immediate attention in a "24/7/365" operation. In this light, service-oriented commercial entities that are early adopters of openSourcePACS may become important in providing consulting and operational services, while still contributing back to the open source community as a whole. Indeed, a sharp contrast exists in comparing the level of support offered by commercial PACS vendors versus openSourcePACS. Moreover, many commercial PACS provide aspects of the same functionality described in openSourcePACS. However, a tradeoff exists between the proprietary (if not typically closed) systems of the former, against the more modular premise and functional extensibility of the latter.

[1] http://www.mii.ucla.edu/openSourcePACS

[2] Compiled statistics for the openSourcePACS website actually indicate that on average, over 90 downloads have occurred per month since the RSNA presentation; however, given the nature of web statistics, it is likely that some of these downloads may be from automated web crawlers; thus, a more conservative number is provided.

Several additions to openSourcePACS are presently being pursued. A complete implementation of the Web Access to DICOM Persistent Objects (WADO) standard is being completed within the imageServer as an additional method for requesting and transmitting images and associated data from archives to the imageViewer. By extension, the web-based viewing components (e.g., from openLaszlo) will be able to more readily access and process images in native web browser formats. Notably, many other software packages and projects already provide some WADO support (e.g., dcm4che [9]), which can be leveraged herein. Integration of openSourcePACS with other existent and well-established open source projects (e.g., dcm4che, OsiriX [10]) is being pursued as a means to further the overall goals of a comprehensive open source solution. Additionally, the imageServer is being extended to handle non-DICOM images, as may be generated through more complex image analyses or nonstandard modalities.

It is important to acknowledge that openSourcePACS does not yet provide a full clinical picture archiving and communication system: while several key components have been completed, more robust services are needed. For example, quality of service (QoS) and true fault tolerance—a vital aspect of any clinical environment—are not available. Similarly, full archive capabilities (e.g., automatically transitioning and retrieving older images from tertiary storage) and backup strategies are not tackled. However, recognizing that alternative (hardware) solutions are available (e.g., redundant array of inexpensive disks, RAID), we believe that the given architecture for openSourcePACS can be extended to address these issues using existing techniques. The overall performance of this architecture will also need to be further tested under high loads and varying conditions (i.e., stress-tested). For example, load balancing performance by the gateway is dependent on choosing appropriate metrics considering the capabilities of a given node versus overall (global) system performance, and perhaps even the urgency of a given request (e.g., clinical retrieval over research processing). Load simulations and real-world deployments of this framework will be reported in future work.

Also, for openSourcePACS to be deployable in clinical environments, proper certification of the software will be required—as is conducted by commercial vendors. In the United States, approval would be required from the Federal Drug Administration (FDA); in Europe and other regions, corresponding certification would also be needed. Such regulatory approvals are being investigated by the authors.

Notably, openSourcePACS can work with existing PACS infrastructures, and can evolve to handle new capabilities—as illustrated by the many future technical directions for the openSourcePACS project.

1) *Peer-to-peer searching*. The routing services in the imageServer broadcast queries to known sources (PACS) in order to find images. In a small (confined) network of agents, this technique is relatively efficient, as network traffic is limited and minimal processing need occur at the point of origin (i.e., the routing agent). Still, as image repositories continue to grow and are linked together for research and clinical purposes (e.g., Biomedical Informatics Research

Network [63], National Digital Mammography Archive [64], efficiently searching for a given image becomes more difficult and different approaches are required to scale in a large, highly distributed network environment. The need for new search methods beyond classic distributed database mechanisms is highlighted in [65] the analysis of which has shown to only effectively scale to $\sim$1000 nodes. One possibility under investigation is to adopt peer-to-peer (P2P) searching techniques, whereby agents propagate searches only to other "local" agents—the query "spreads" over a network; newer techniques have been shown to guarantee the discovery of targeted data in such networks with a controlled number of data hops [66], alleviating the disadvantages associated with earlier P2P networks.

2) *Cohort searching*. Presupposing the creation of large imaging networks and efficient search methods, the openSourcePACS concept of open querying can empower researchers to automatically create large imaging-based patient cohorts in a retrospective manner; specific data warehouses have been proposed for this purpose [67]. Given proper authorization to use imaging data for research, agent querying services in imageServer can be used to find patients' images that meet clinical and/or image-specific criteria. An automated de-identification package (e.g., removing patient identifiers from image headers and other clinical data), already in DataServer [68], is being generalized for use with openSourcePACS.
3) *Content-based image retrieval*. Medical content-based image retrieval (CBIR) has been an ongoing pursuit of multiple research groups [69], with good results in highly focused domain areas. In the long term, the ability to search for like images will become a powerful tool with several potential applications, including medical decision support (e.g., find patients with similar tumors in the same anatomical brain location); as the amount of imaging accumulates, the value of CBIR will clearly increase.
4) *Integrated multimedia patient records*. The allure of the electronic medical record (EMR) is perhaps best given by the longitudinal, virtual patient record [70], seamlessly accessing and integrating imaging and all other modes of communication (text, graphical, video, audio) all into a comprehensive display. The juxtaposition of openSourcePACS and DataServer is a step in this direction, though the complexity of re-organizing and filtering the wealth of clinical information into a single interface is an ongoing challenge and topic of research [18]. Indeed, as new imaging modalities become commonly available, novel techniques to visualize this data must be contemplated.

Past works in imaging and open source projects have remained fragmented, only offering niche solutions. Thus, developers are often left with the task of re-inventing or integrating dissimilar software components; ultimately, it is hoped that openSourcePACS, as an umbrella framework for all these efforts, can serve as starting point to foster new developments in PACS and the use of imaging in support of evidence-based medical practice, research, and education.

ACKNOWLEDGMENT

The authors would like to thank G. Weinger and S. Barretta for their work on the implementation of openSourcePACS; and W. Hsu for his review of the literature.

REFERENCES

[1] B. Kaplan and N. T. Shaw, "People, organizational, and social issues: Evaluation as an exemplar," in *IMIA Yearbook of Medical Informatics*, Stuttgart, Germany: Schattauer Verlag, 2002, pp. 91–102.
[2] M. van Herck and L. Zjip. (2005, Sep.). Conquest DICOM software website. [Online]. Available: http://www.xs4all.nl/~ingenium/dicom.html
[3] P. Sau (2005, Mar.). CDMEDIC PACS website, version 6.2. [Online]. Available: http://sourceforge.net/projects/cdmedicpacsweb/
[4] RainbowFish Software (2005, Sep.). Introduction, PACSOne Server website. [Online]. Available: http://www.pacsone.net/index.htm
[5] MiniWebPACS main website. (2005, Sep.). [Online]. Available: http://miniwebpacs.sourceforge.net/
[6] T. Sakusabe (2005, Jan.). DIOWave Visual Storage main website [Online]. Available: http://diowave-vs.sourceforge.net/
[7] OFFIS Computer Science Institute (2005, Mar.). DCMTK—DICOM Toolkit main website. [Online]. Available: http://dicom.offis.de/dcmtk
[8] S. M. Moore, S. A. Hoffman, and D. E. Beecher, "DICOM shareware: A public implementation of the DICOM Standard," in *Proc. SPIE, Medical Imaging 1994-PACS: Design and Evaluation*, vol. 2165, pp. 772–781.
[9] G. Zeilinger (2005, Feb.). dcm4che, A DICOM implementation in JAVA, website. [Online]. Available: http://sourceforge.net/projects/dcm4che/
[10] A. Rosset, L. Spadola, and O. Ratib, "OsiriX: An open-source software for navigating in multidimensional DICOM images," *J. Digit. Imag.*, vol. 17, no. 3, pp. 205–216, Sep. 2004.
[11] P. Puech and L. Boussel (2005, Sep.). DICOM Works main website. [Online]. Available: http://dicom.online.fr/
[12] M. Kanellopoulos (2005, Aug.). Sante viewer main website. [Online]. Available: http://users.forthnet.gr/ath/mkanell/viewer/viewer.html
[13] A. M. Loening and S. S. Gambhir, "AMIDE: A free software tool for multimodality medical image analysis," *Mol. Imag.*, vol. 2, no. 3, pp. 131–137, 2003.
[14] C. Rorden (2005, Sep.). ezDICOM software, website. [Online]. Available: http://www.psychology.nottingham.ac.uk/staff/cr1/ezdicom.html#users
[15] K. Muto, Y. Emoto, T. Katohji, H. Nagashima, A. Iwata, and S. Koga, "PC-based web-oriented DICOM server," (InfoRad Presentation) Radiology(P), 2000.
[16] National Institutes of Health (2005, Sep.). imageJ: Image processing and analysis in Java, website. [Online]. Available: http://rsb.info.nih.gov/ij/
[17] S. T. C. Wong, D. Tjandra, H. Wang, and W. Shen, "Workflow-enabled distributed component-based information architecture for digital medical imaging enterprises," *IEEE Trans. Inform. Technol. Biomed.*, vol. 7, no. 3, pp. 171–183, Sep. 2003.
[18] A. A. Bui, R. K. Taira, S. El-Saden, A. Dordoni, and D. R. Aberle, "Automated medical problem list generation: A practical method to create a patient TimeLine," in *Proc. MedInfo 2004*,, pp. 587–591.
[19] H. K. Huang, *PACS: Basic Principles and Applications*. New York: Wiley-LISS, 1999.
[20] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: The Condor experience," *Concurr. Comput. Pract. Exp.*, vol. 17, no. 2–4, pp. 323–356, Feb.–Apr. 2005.
[21] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *Int. J. Supercomput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.
[22] L. P. Clarke, B. Y. Croft, E. Staab, H. Baker, and D. C. Sullivan, "National Cancer Institute Initiative: Lung image database resource for imaging research," *Acad. Radiol.*, vol. 8, no. 5, pp. 447–450, May 2001.
[23] S. G. Armato, III, G. McLennan, M. F. McNitt-Gray, C. R. Meyer, D. Yankelevitz, D. R. Aberle, C. I. Henschke, E. A. Hoffman, E. A. Kazerooni, H. MacMahon, A. P. Reeves, B. Y. Croft, and L. P. Clarke, and Lung Image Database Consortium Research Group, "Lung Image Database Consortium: Developing a resource for the medical imaging research community," *Radiology*, vol. 232, pp. 739–748, 2004.
[24] B. J. Hillman, "The American College of Radiology Imaging Network (ACRIN): Research educational opportunities for academic radiology," *Acad. Radiol.*, vol. 9, pp. 561–562, May 2002.

[25] M. S. Brown, S. K. Shah, R. C. Pais, Y. Z. Lee, M. F. McNitt-Gray, J. G. Goldin, A. F. Cardenas, and D. R. Aberle, "Database design and implementation for quantitative image analysis research," *IEEE Trans. Inform. Technol. Biomed.*, vol. 9, no. 1, pp. 99–108, Mar. 2005.
[26] M. W. Vannier and R. M. Summers, "Sharing images," *Radiology*, vol. 228, pp. 23–25, 2003.
[27] S. Ardekani and U. Sinha, "Geometric distortion correction of high-resolution 3T diffusion tensor brain images," *Magn. Reson. Med.*, vol. 54, no. 5, pp. 1163–1171, 2005.
[28] Insight Segmentation and Registration Toolkit website (2005, Sep.). [Online]. Available: http://www.itk.org
[29] The Visualization Toolkit website (2005, Sep.). [Online]. Available: http://public.kitware.com/VTK/index.php
[30] K. Liakos, A. Burger, and R. Baldock, "A scalable mediator approach to process large biomedical 3-D images," *IEEE Trans. Inform. Technol. Biomed.*, vol. 8, no. 3, pp. 354–359, Sep. 2004.
[31] Y. Kawaski, F. Ino, Y. Mizutani, N. Fujimoto, T. Sasama, Y. Sato, N. Sugano, S. Tamura, and K. Hagihara, "High-performance computing service over the Internet for intraoperative image processing," *IEEE Trans. Inform. Technol. Biomed.*, vol. 8, no. 1, pp. 36–46, Mar. 2004.
[32] S. Hastings, S. Oster, S. Langella, T. M. Kurc, T. Pan, U. V. Catalyurek, and J. H. Saltz, "A grid-based image archival and analysis system," *J. Amer. Med. Inform. Assoc.*, vol. 12, no. 3, pp. 286–295, May.–Jun. 2005.
[33] J. Montagnat, V. Breton, and I. E. Magnin, "Using grid technologies to face medical image analysis challenges," in *Biogrid '03, Proc. IEEE CCGrid03*, Tokyo, Japan, May. 12–15, 2003, pp. 588–593.
[34] Trispark Corp (formerly SoftLink) (2005, Sep.). Java DICOM Toolkit (JDT) website. [Online]. Available: http://www.trispark.com/jdt/features.htm
[35] Sun Java Advanced Imaging (JAI) main website (2005, Sep.). [Online]. Available http://java.sun.com/products/java-media/jai/index.jsp
[36] S. Saladi, P. Pinnamaneni, and J. Meyer, "Texture-based 3-D brain imaging," in *Proc. 2nd IEEE Int. Symp. Bioinformatics and Bioengineering (BIBE '01)*, Mar. 2001, p. 136.
[37] P. Vuylsteke, E. Schoeters, N. V. Agfa-Gevaert, and B. Mortsel, "Image processing in computed radiography," in *Proc. Comput. Tomography and Image Processing*, 1999, pp. 87–101.
[38] R. E. Wendt, III, "Automatic adjustment of contrast and brightness of magnetic resonance images," *J. Digit. Imag.*, vol. 7, no. 2, pp. 95–97, May 1994.
[39] J. D. N. Dionisio, A. A. Bui, R. Ying, C. Morioka, and H. Kangarloo, "A gesture-driven user interface for medical image viewing," in *Proc. Radiology*, 2003, p. 807.
[40] T. R. Henry, S. E. Hudson, and G. L. Newell, "Integrating gesture and snapping into a user interface toolkit," in *Proc. 3rd ACM Symp. User Interface Software and Technology (UIST)*, 1990, pp. 112–122.
[41] S. Keates, R. Potter, C. Perricos, and P. Robinson, "Gesture recognition—Research and clinical perspectives," in *Proc. RESNA' 97,* Pittsburgh, PA, 1997, pp. 333–335.
[42] J. I. Hong and J. A. Landay, "SATIN: A toolkit for informal ink-based applications," in *Proc. 13th ACM Symp. User Interface Software and Technology (UIST)*, 2000, pp. 63–72.
[43] A. C. Long, Jr., J. A. Landay, L. A. Rowe, and J. Michiels, "Visual similarity of pen gestures," in *Proc. CHI 2000, ACM Conf. Human Factors in Computing Systems*, 2000, vol. 2, no. 1, pp. 360–367.
[44] J. A. Landay and B. A. Myers, "Sketching interfaces: Toward more human interface design," *IEEE Comput.*, vol. 34, no. 3, pp. 56–64, Mar. 2001.
[45] D. Rubine, "Specifying gestures by example," in *Proc. 18th Annu. Conf. Computer Graphics and Interactive Techniques*, 1991, vol. 25, no. 4, pp. 329–337.
[46] *Digital Imaging and Communications in Medicine Standards Committee. DICOM Supplement 23: Structured Reporting Storage SOP Classes*, National Electrical Manufacturers Association, Rosslyn, VA, 2000.
[47] *Digital Imaging and Communications in Medicine Standards Committee, Working Group 11. DICOM Supplement 33: Grayscale Softcopy Presentation State Storage*, National Electrical Manufacturers Association, Rosslyn, VA, Sep. 1999.
[48] E. L. Siegel and D. S. Channin, "Integrating the healthcare enterprise: A primer," *Radiographics*, vol. 21, pp. 1339–1341, 2001.
[49] W. W. Stead, B. J. Kelly, and R. M. Kolodner, "Achievable steps towards building a National Health Information Infrastructure in the United States," *J. Amer. Med. Inform. Assoc.*, vol. 12, no. 2, pp. 113–120, Mar.–Apr. 2005.
[50] A. A. Bui, G. S. Weinger, S. J. Barretta, J. D. N. Dionisio, and H. Kangarloo, "DataServer: An XML gateway for medical research applications," *Ann. NY Acad. Sci.*, vol. 980, pp. 236–246, 2002.
[51] OpenLaszlo main website (2005, Sep.). [Online]. Available: http://www.openlaszlo.org
[52] A. Tirado-Ramos, J. Hu, and K. P. Lee, "Information object-definition-based unified modeling representation of DICOM structured reporting: A case study of transcoding DICOM to XML," *J. Amer. Med. Inform. Assoc.*, vol. 9, no. 1, pp. 63–72, 2002.
[53] W. M. Tellis and K. P. Andriole, "Integrating multiple clinical information systems using the Java message service framework," *J. Digit. Imag.*, vol. 17, no. 2, pp. 80–86, Jun. 2004.
[54] B. Henderson, S. Camorlinga, and J. C. Degagne, "A cost-effective web-based teaching file system," *J. Digit. Imag.*, vol. 17, no. 2, pp. 87–91, Jun. 2004.
[55] E. Weinberger, R. Jakobovits, and M. Halsted, "MyPACS.net: A web-based teaching file authoring tool," *Amer. J. Roentgenol.*, vol. 183, no. 3, pp. 679–582, Sep. 2002.
[56] A. Rosset, O. Ratib, A. Geissbuhler, and J. P. Valle, "Integration of a multimedia teaching and reference database in a PACS environment," *Radiographics*, vol. 22, no. 6, pp. 1567–1577, 2002.
[57] G. B. Bell and A. Sethi, "Matching records in a national medical patient index," *Commun. ACM*, vol. 44, no. 9, pp. 83–88, 2001.
[58] A. A. Bui, R. K. Taira, J. D. N. Dionisio, D. R. Aberle, S. El-Saden, and H. Kangarloo, "Evidence-based radiology: Requirements for electronic access," *Acad. Radiol.*, vol. 9, pp. 662–669, Jun. 2002.
[59] A. A. Bui, R. K. Taira, D. Goldman, J. D. N. Dionisio, D. R. Aberle, S. El-Saden, J. Sayre, T. Rice, and H. Kangarloo, "Effect of an imaging-based streamlined electronic health care process on quality and costs," *Acad. Radiol.*, vol. 11, no. 1, pp. 13–20, Jan. 2004.
[60] Free Software Foundation (1999, Feb.). Gnu Lesser General Public License. [Online]. Available: http://www.gnu.org/∼copyleft/lesser.html
[61] E. S. Raymond (2000, Sep.). The Cathedral and the Bazaar. [Online]. Available: http://www.catb.org/∼esr/writings/cathedral-bazaar-/cathedral-bazaar
[62] K. Johnson (2001, Jun.). A descriptive process model for open source software development, M.Sc. thesis Dep. Comput. Sci. Univ. Calgary. [Online]. Available: http://sern.ucalgary.ca/students/theses-/KimJohnson/toc.htm
[63] S. Santini and A. Gupta, "The role of Internet imaging in the Biomedical Informatics Research Network," in *Proc SPIE*, 2003, vol. 5018, Internet Imaging IV, pp. 99–110.
[64] B. G. Beckerman and M. D. Schnall, "Digital information management: A progress report on the National Digital Mammography Archive," *Proc SPIE,* 2002, vol. 4615, Biomedical Diagnostic, Guidance, and Surgical-Assist Systems IV, pp. 98–108,
[65] M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu, "Mariposa: A wide-area distributed database system," *VLDB J.*, vol. 5, pp. 48–63, 1996.
[66] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury, "Percolation search in power law networks: Making unstructured peer-to-peer networks scalable," in *Proc IEEE 4th Int. Conf. Peer-to-Peer Computing (P2P 2004)*, pp. 2–9.
[67] S. T. C. Wong, K. Soo Hoo, R. C. Knowlton, K. D. Laxer, X. Cao, R. A. Hawkins, W. A. Dillon, and R. L. Arenson, "Design and application of a multimodality image data warehouse framework," *J. Amer. Med. Inform. Assoc.*, vol. 9, no. 3, pp. 239–254, 2002.
[68] R. K. Taira, A. A. Bui, and H. Kangarloo, "Identification of patient name references within medical documents using semantic selectional restrictions," in *Proc. AMIA Fall Symp.*, 2002, pp. 757–761.
[69] H. Müller, N. Michoux, D. Bandon, and A. Geissbuhler, "A review of content-based image retrieval systems in medicine: Clinical benefits and future directions," *Int. J. Med. Inform.*, vol. 73, pp. 1–23, 2004.
[70] E. H. Shortliffe, "The evolution of electronic medical records," *Acad. Med.*, vol. 74, no. 4, pp. 414–419, 1999.

Alex A. T. Bui (M'03), photograph and biography not available at the time of publication.

Craig Morioka, photograph and biography not available at the time of publication.

John David N. Dionisio, photograph and biography not available at the time of publication.

Ricky K. Taira, photograph and biography not available at the time of publication.

David B. Johnson, photograph and biography not available at the time of publication.

Denise R. Aberle, photograph and biography not available at the time of publication.

Usha Sinha, photograph and biography not available at the time of publication.

Suzie El-Saden, photograph and biography not available at the time of publication.

Siamak Ardekani, photograph and biography not available at the time of publication.

Hooshang Kangarloo, photograph and biography not available at the time of publication.