Journal of Digital Imaging

VOL 9, NO 1 FEBRUARY 1996

Architectural Design and Tools to Support the Transparent Access to Hospital Information Systems, Radiology Information Systems, and Picture Archiving and Communication Systems

Ricky K. Taira, Claudine M. Breant, Hing M. Chan, Lujia Huang, and Daniel J. Valentino

The fragmentation of the electronic patient record among hospital information systems (HIS), radiology information systems (RIS), and picture archiving and communication systems (PACS) makes the viewing of the complete medical patient record inconvenient. The purpose of this report is to describe the system architecture, development tools, and implementation issues related to providing transparent access to HIS, RIS, and PACS information. A client-mediator-server architecture was implemented to facilitate the gathering and visualization of electronic medical records from these independent heterogeneous information systems. The architecture features intelligent data access agents, run-time determination of data access strategies, and an active patient cache. The development and management of the agents were facilitated by data integration CASE (computer-assisted software engineering) tools. HIS, RIS, and PACS data access and translation agents were successfully developed. All pathology, radiology, medical, laboratory, admissions, and radiology reports for a patient are available for review from a single integrated workstation interface. A data caching system provides fast access to active patient data. New network architectures are evolving that support the integration of heterogeneous software subsystems. Commercial tools are available to assist in the integration procedure.

KEY WORDS: data base integration, medical information systems, software mediation, computer assisted software engineering (CASE) tools.

MEDICAL practice now critically relies on data stored in hospital information systems (HIS), radiology information systems (RIS), and picture archiving and communication systems (PACS).[1] However, current implementations of HIS, RIS, and PACS provide only a first-order solution to the global problems associated with storing, searching, communicating, and presenting the complete electronic multimedia patient record. Each of these information systems has evolved independently and manages overlapping subsets of data associated with the patient. This has created a situation in which the electronic medical record is fragmented along artificially determined information boundaries, thus limiting the collective usefulness and utilization of these separate information systems for the purpose of supporting health care. As an example, at our institution three separate types of computer terminals from HIS, RIS, and PACS are distributed throughout the radiology department. A radiologist needing to review a patient's pathology reports, medical history, and radiology data must go to a PACS workstation to review the patient's radiology images, to an HIS terminal to view pathology and laboratory results, and finally to an RIS terminal to view radiology reports. Unnecessary burdens are placed onto the clinician to memorize cumbersome log in procedures, meaningless record identification numbers, keyboard mappings, and inefficient data base navigation methods. The time and energy spent having to perform these tasks often inhibit a radiologist from viewing all diagnostic information for a patient making them unaware of test results and/or observations from other current and/or historic diagnos-

From the Departments of Radiological Sciences and Surgery, University of California, Los Angeles.

Supported by National Institute of Health Grant No. USPHS CA39063, awarded by the National Cancer Institute.

Address reprint requests to Ricky K. Taira, PhD, Department of Radiological Sciences, University of California, Los Angeles, 10833 Le Conte Ave, Los Angeles, CA 90024-1721.

0897-1889/96/0901-0001$3.00/0

tic examinations. This partial knowledge may compromise patient care because results from nonradiology diagnostic procedures have shown to affect the perception and/or interpretation of radiographs.[2]

Although much theoretical work exists on methods for integrating distributed heterogeneous data base systems,[3-8] there are few integrated hospital systems that provide easy access to the complete electronic patient record. Standardization of protocols and data models are at the heart of the problem (differences in hardware platforms, communication protocols, data base management software, etc). In this report we describe our efforts to develop a practical solution to providing transparent user access to HIS, RIS, and PACS information. We introduce our client-mediator-server architecture that features data access agents managed by an intelligent mediation software layer. The main body of the report describes the use of computer-assisted software engineering (CASE) tools that were used to accelerate and formalize the design and implementation of integrated medical information systems.

Clinical Environment

The development of a campus-wide project to connect all medical services within our institution began in 1988. A fiber-optic token ring backbone within our medical center supports various communication protocols including SNA, IPX, and TCP/IP. Several subnetworks are connected to this backbone including high-speed circuits from PACS.[9] A brief description of our HIS, RIS, and PACS follows.

The HIS manages patient demographics, billing information, admissions-discharge-transfer reports, laboratory results, pathology results, and medical/surgical reports. The system, installed in the 1970s, has over 300 user terminals connected to a centralized IBM mainframe (IBM, Purchase, NY) running under the multiple virtual storage (MVS) operating system. It uses an IMS (Information Management System) hierarchical data base management system. Communication to terminals is through the IBM 3270 serial communication protocol. Data base access is restricted to in-house developed application programs that provide custom user specific views (clinicians, administrators, receptionists, technologist, etc).

The RIS manages radiology schedules, requisition information, study descriptions, and study reports. It was installed in 1990 and has about 125 user terminals connected to a centralized VAX/VMS 6200 minicomputer (Digital Equipment Corp, Maynard, MA). It uses the MUMPS-based Maxifile IV data base software (Dimensional Medicine Inc, Minnetonka, MN). User access is also restricted to user class specific (radiologists, receptionists, technologists, etc) application programs.

Our PACS began clinical operation in 1990 and currently has 10 viewing stations. It contains both textural and image data.[10] The text information includes radiology image acquisition descriptions, basic patient data, procedure descriptions, and image archival directory information. This information is managed by a relational data base system (Sybase, Emeryville, CA) configured in a client-server architecture. The server runs on a Sun 4 computer (Sun Microsystems, Mountain View, CA) under the UNIX operating system. Data base clients access the system using remote structured query language (SQL) procedure calls written in the C-programming language. PACS image data are stored as UNIX flat files because of their higher storage, communication, and processing requirements. The image data are distributed over several nodes on various magnetic disk banks and optical library units.[11]

MATERIALS AND METHODS

System Requirements

The team assigned to the project included: an operations manager (10% time), a computer science graduate student (25% time), an assistant professor in medical imaging (25% time), a PACS manager (10% time), an RIS manager (5% time), and one data base administrator from the Department of Surgery (25% time). Our goal was to provide data integration in a structured, evolving manner at a reasonable cost.

The clinical requirements for accessing the data from the HIS, RIS, and PACS included the following: (1) Simple interface—the user should be able to use the system without extensive instructions; (2) read-only access—users would not be allowed to update any HIS, RIS, or PACS information from the integrated interface; (3) fast response—thereby not impeding the radiologist's throughput; (4) reliable. The challenge for system development is further compounded by the following technical requirements that

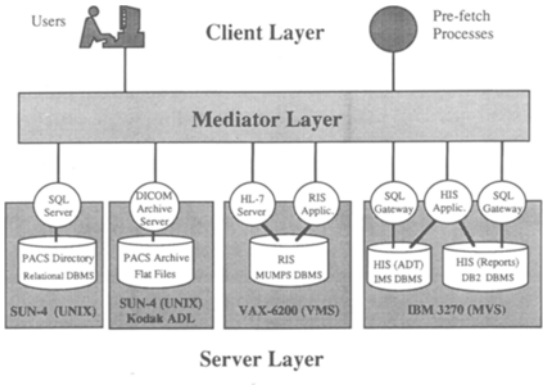

Fig 1. Client-Mediator-Server architecture. The mediator layer hides the lower-level data base access methods from the client.

we impose: (1) System integration efforts cannot affect the existing HIS, RIS, and PACS operations; (2) the system architecture should be extensible to include other future information systems; (3) the system must be easily adaptable to system and technology changes.

System Architecture

Since the inception of the HIS, RIS, PACS integration project, the system architecture design has been a central issue. The architecture must be flexible in accommodating diverse information processing tasks, scalable in its user base, extensible to various information sources and software subsystems and adaptable to system and technology changes.

We implemented a *client-mediator-server* architecture[12,13] to support the integration of our HIS, RIS, and PACS (Fig 1). In this architecture, *client* processes initiate patient information requests. The Client Layer includes for example, PACS workstation processes. The request is directed to a data access Mediator Layer (see Fig 1) and is presented as a precompiled form. The form defines what data items are to be retrieved as well as the expected presentation format for the data. There is no specification for where the data might be located or how it is structured. Thus, the query interface is in the paradigm of "query-by-form."[14] Conceptually, the request from the Client Layer to the Mediator Layer can be dialogued as "Please fill in all the necessary information on this form for this particular patient." Different types of clients (intensive care unit [ICU] PACS workstations, Thoracic radiology PACS workstations, Pediatric PACS workstations, etc) may have different forms specifying the particular information to be returned.

The Mediator Layer contains three important components for retrieving distributed patient information. First, it contains a collection of data access and filtering *agents.*[15] These agents are software robots that know how to access the hospital network, make session connections with various data base machines, communicate with remote data bases in the appropriate language, navigate through target data bases, and translate data from a source to target format. Agent development was facilitated using the data integration CASE tools described below. Second, the Mediator Layer contains a process that monitors the state of the resources required to access patient data. These resources include network circuits, storage devices, computer central processing units (CPUs), and various data base server processes. Third, the Mediator Layer contains a global information *cache.* This cache is a redundant storage subsystem for currently active patients. It provides a method for storing the patient records with highest probability of access onto a fast retrieval system.

The Server Layer of our architecture includes the software processes that have direct access to the HIS, RIS, and PACS data bases. Initially, these servers consisted of SQL and digital imaging and communications in medicine (DICOM) archive servers for PACS data, the Maxifile IV application program server to RIS, and the IMS application to HIS data. More recently, an HL-7 server (Digital Equipment Corp) has been developed for accessing RIS data, and an SQL gateway for more direct access to HIS data (see Fig 1). These servers require knowledge of the implementation of the target HIS, RIS, and PACS data bases. The Mediator Layer of our architecture encapsulates the data and access methods within these systems thus hiding the heterogeneous and distributed nature of the data from the Client Layer.

Development Environment

Figure 2 illustrates our development environment which features a set of CASE tools (Integration Works; Data Integration Solutions Corp, Gardena, CA)[16] running on a Sun Sparc 2 minicomputer (Sun Microsystems) for the creation, registration, and coordination of information processing agents. Agents created by these tools are executable scripts that are stored within a global resource repository that registers the input needs and processing capabilities of each agent. A Workflow Designer tool is used to link the requirements of one mediator agent with the capabilities of another. Thus, low-level processing agents can be linked to perform complex high-level tasks. A description of the data integration CASE tools follows.

Global Schema Designer

Our first step toward system integration involved identifying the information objects to be shared and the global views required for each client application. We analyzed the

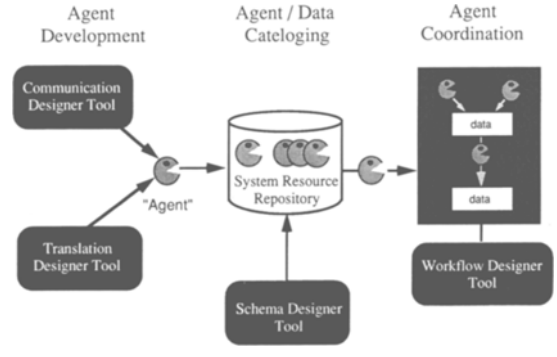

Fig 2. Development environment for the creation, registration, and linking of information processing agents.

data available from HIS, RIS, and PACS screens and solicited copies of the complete data schemata for each component data base system. The schemata were transferred to our development computer in the form of *data definition files*. These files described the exportable data entities from HIS, RIS, and PACS. For each data item, the attribute name, data type (eg, alphabetic, numeric, etc), hierarchical level of the data item (for composite attributes), storage length, sample value, default value if any, alias names, and narrative description were given and/or edited. These files were formatted as C and COBOL structures and included into the system's resource repository.

An extended entity relationship (ER) diagramming tool was then used to assist in graphically modeling the data (Fig 3). A graphical representation of the global schema allows administrators and programmers to more easily understand, locate, and access distributed information. A *Load View* feature of the tool was used to convert the data definition files into a tabular representation of the HIS, RIS, and PACS schemata. The tabular representations were then imported by the ER diagramming tool as graphical entities (Fig 3). Using the tool, views from different component data bases were merged by selecting an entity and dragging an arrow to conceptually interrelate it with another entity. Relationships including aggregation, generalization, and association and cardinalities (degree of relationship) were specified at this stage. Both forward and reverse relationships and cardinalities were specified for related entity pairs. To reduce the complexity of the global ER diagram, entities such as "Patient Reports" were logically clustered into a composite meta entity composed of medical reports, laboratory reports, pathology reports, and radiology reports. This meta entity could then be graphically collapsed and expanded during display.

Once the global schema was created graphically, relational data base table definitions were automatically generated by the ER-diagramming tool and stored as a SQL text command file. The command file was then used to create global relational tables to be used for generating output views for use by our system cache.

Data Base Communication and Navigation

Automating the collection of data from legacy information systems was the most difficult task we faced. Because of the stand-alone origins of our HIS and RIS, a screen capture method was used to access data from these systems.[17] The advantages of this approach are the following: (1) No new code (eg, data base language translators, communication, etc) needs to be developed or installed on the legacy data base systems. The integration process makes

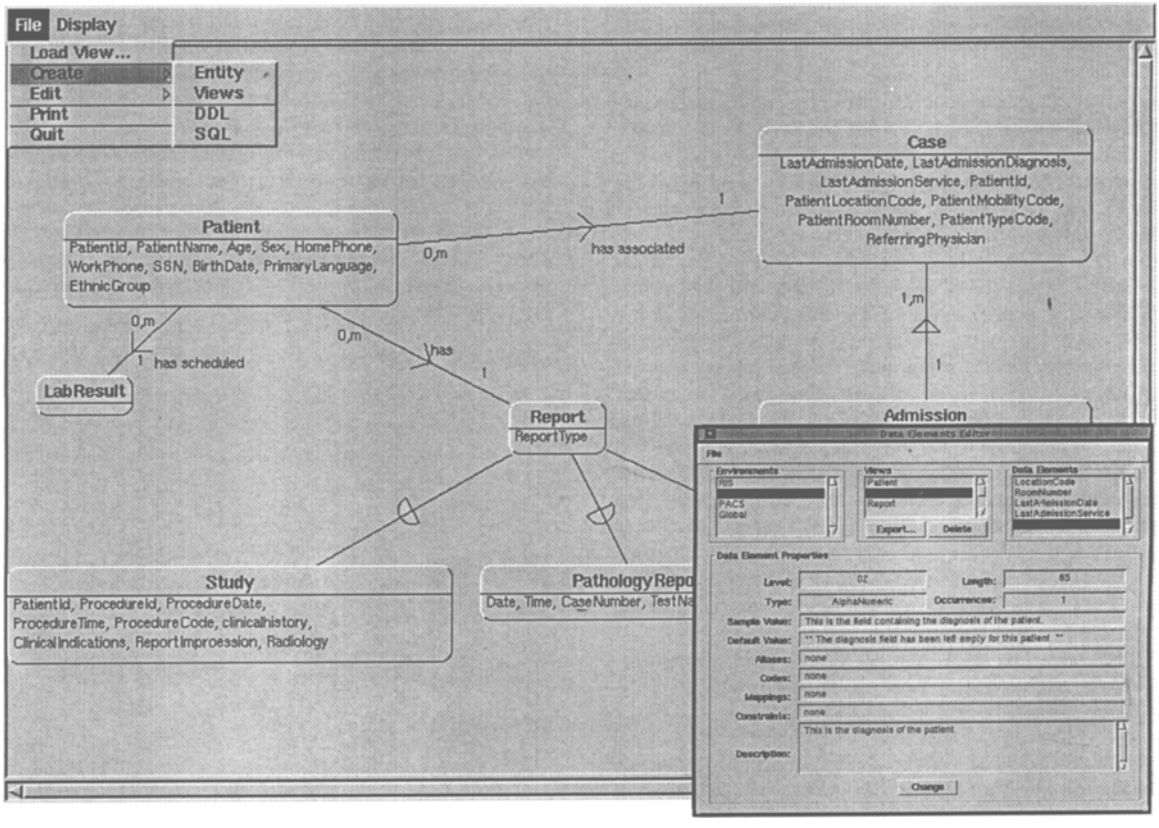

Fig 3. The Graphic Schema Designer tool allows developers to import views from component data bases, define interdata base entity relationships, and edit attribute information. Relational data base tables can be automatically generated from these high-level graphical specifications.

efficient use of already existing communication and data base software that has been clinically tested. (2) Implementation details of the local data base management system are invisible to the integration developers. (3) Communication with the host and application software merely involves setting up a new user on the local host. (4) The application program of each legacy system acts as a data filter protecting the data base from corruption caused by faulty programs and/or unauthorized data access. The disadvantages of the screen access approach include the following: (1) Performance is not optimized because direct access using remote procedure calls, in the appropriate data base language, is not allowed or available. There is a large overhead in electronically navigating through possibly many screens to access desired data. (2) Only a limited view of the data is provided, as determined by the available application program running on the legacy data base management system. (3) Changes in the presentation of the data require changes in the integration software. (4) Fault recovery is more difficult because there is no formal message passing facilities. (5) Differences in keyboard mapping and terminal types present extra levels of heterogeneity that must be worked out.

The first step in developing a data access agent for a closed legacy system (eg, HIS) is to "teach" the agent how to access a single patient's records. The following text explains the creation of our HIS data access agent. Agent development for RIS follows a similar methodology. This initial teaching step was performed using a *session capture* utility. This utility consists of a terminal emulation window and monitoring software that transparently captures into a *session capture* log file, all user inputs and all HIS system responses. We used this utility to conduct an interactive session with the target HIS using native communication software (telnet with tn3270 emulation) and the HIS application program mode for radiologists. The interactive session involved logging into the HIS, viewing the patient's demographic information, admission and discharge records, viewing various procedure lists (laboratory, medical, and pathology) and viewing the full text reports for these procedures. The input keystrokes and system screen responses were automatically recorded by the session capture utility providing the system with the necessary basic knowledge for navigating through the HIS for accessing a single patient's data. From the perspective of the target data base, this step appears simply to be another data base session with a clinical user.

Once this initial learning step was completed, a *session replay* utility was used to replay the just completed interactive session (Fig 4). The replay is driven by the data in the session log file and hence does not require a network connection to the actual HIS. This replay feature was useful for reviewing the sequence of response screens, the menu options, and the location of interesting data items from HIS. The utility provided "rewind," "fast-forward," "speed control," and "pause" control options.

From the session replay utility, a first version of the communication code was automatically generated based on the data contained within the session log file. This initial communication program is coded in a high-level language known as *SDL* (System Description Language). SDL is

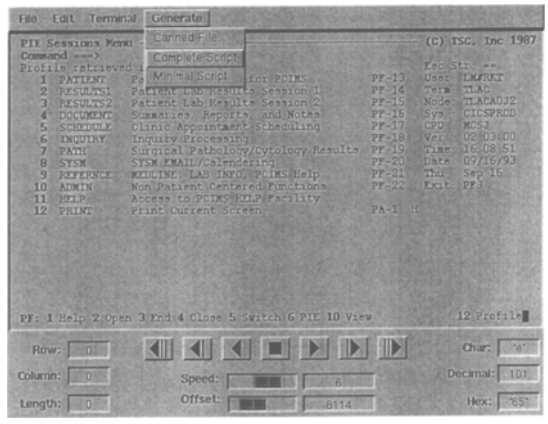

Fig 4. The Communication Designer session replay utility.

based on the 1988 Consultive Committee on International Telephony and Telegraphy standard for distributed communication systems and provides a description of the behavior of a distributed system in terms of a stimulus-response relationship. The SDL code models the interaction of the user and the target data base management system as a finite state machine. It describes the various HIS states (eg, log in screen, main menu screen, pathology report screen, etc) and the expectation conditions (ie, keystroke entries) for the valid transition between states (see Fig 5). The initial SDL code was then edited to refine the expectation conditions for a state transition, parameterize patient identification variables, add control loops to acquire multiple pages of reports, add data extraction statements for required data, provide more meaningful state and variable names, and improve overall program documentation. Performing these tasks did not require a sophisticated software programmer, as the SDL code is very high level and descriptive. The code to define a state and the transition conditions is typically about 10 to 25 statements in length.

After editing the SDL code, the session replay utility was used to execute a command to automatically generate a C++ program directly from the SDL code. The C++ code is then compiled and linked, to produce a run-time process that knows how to access data from the HIS system as encoded by the finite-state machine representation. This program was then installed into the system repository as a

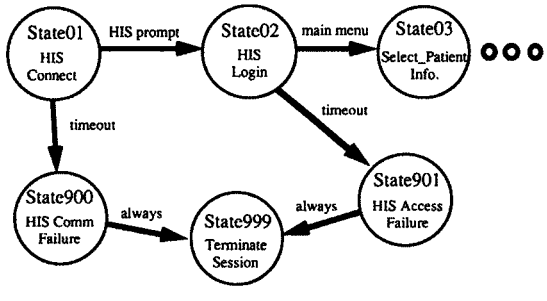

Fig 5. Data access process to various data bases is represented as a finite-state machine using SDL coding.

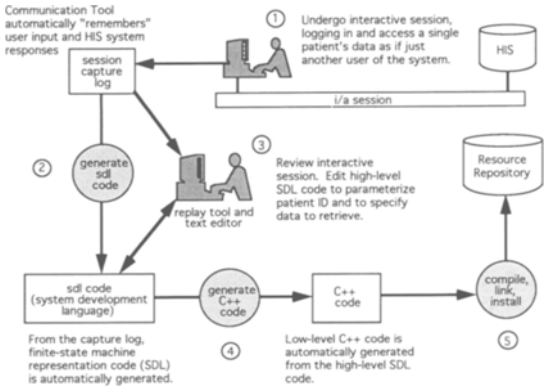

Fig 6. Development steps for generating a data access agent to HIS.

HIS data access agent. We never had to edit directly the C++ code. Figure 6 summarizes the overall procedure.

This process of session capture, session replay, SDL code editing, SDL code translation to C++, and C++ code compiling, linking, and installing was also applied to the RIS. Several iterations of these steps were required to develop a reliably working agent. In the case of PACS, a stored SQL procedure routine was developed to serve as the PACS data access agent.

Data Translation and Formatting

Translators are needed in response to problems that arise because of name, format, and structural differences for the same entity attribute.[18] The definition of the mapping between input views from the global schema and output custom client views was defined by our global system administrator and the code was generated using a *Translation Designer* tool (Fig 7). The Translation Designer uses an outline model paradigm to divide the information into input and output views. Input views were imported from the global schema stored in the system's resource repository. These views include descriptions of entities such as patient demographics, procedures, schedules, case descriptions,

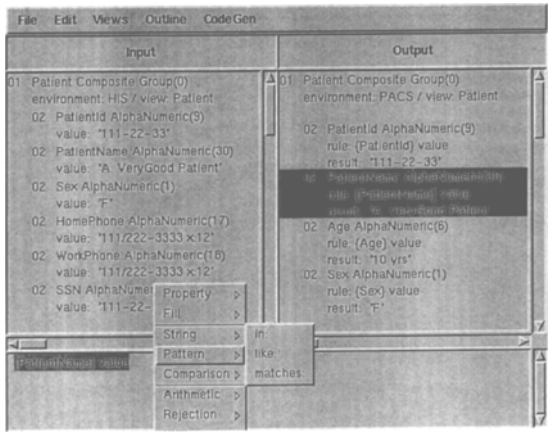

Fig 7. Translation Designer tool for mapping input views to output user views.

and reports. Output views (ie, workstation views) are loaded into the system's resource repository (also a dictionary/directory) using a COBOL record layout format. This output view is then imported into the Translator Designer outline viewer (Fig 7).

In the Translation Designer environment, mappings between elements of input views and output views are defined using conversion and data integrity rules. These translation rules were applied using pull-down paste functions available from the Translation Designer utility, (eg, centering, formatting, arithmetic operations, string operations). Changes in patient hospital identification numbers and date formats were the most common transformations required. Once the translation rules were specified, a "generate code" command was issued from the utility that automatically creates C++ code to transform between input and output views. The resulting executable code inputs a flat file containing the data collected by the data access agents and generates the required output view also as a flat file.

Workflow Design

The *Workflow Designer* tool (Fig 8) was used to construct graphical descriptions of how communication agents (scripts), data files, translation filters, and applications programs were to be assembled, ordered, and interfaced to complete the entire data integration strategy. The graphical representation allowed for intuitive visualization of the flow of data and control for the distributed set of tasks that were to be executed. The Workflow Designer was also used to generate configuration files to control the run-time environment of the data integration scripts. Figure 8 shows the graphical Workflow Design diagram for accessing RIS information.

User Interface

Perhaps one of the most time-consuming tasks was designing the presentation screens for the integrated HIS, RIS, PACS data. How to develop mechanisms to allow a user to easily visualize and access volumous record-oriented data on a single 12" × 14" computer screen has not been well studied thus far. A user may be interested in several data items, some of which should be displayed simultaneously with related items, whereas others are fairly disjointed and can be reviewed separately. It is crucial that the interface be usable with relatively little training. We required that a radiologist can approach the interface and use it instantly to view clinical data.

Our initial user interface design was based on interviews with a number of radiologists from our institution and from work published by Nygren et al[19] at the Center for Human-Computer Studies at the University of Uppsala. Paper models of the integrated HIS, RIS, PACS interface were developed and modified accordingly until a generally acceptable presentation screen was developed. The radiologists interviewed were asked to make recommendations on which data items and options were to be displayed, which should be summarized, and which should be hidden under, for example, pull down menus. The interface was built in Motif using the X-Designer CASE tools (V.I. Corp, Northampton, MA).

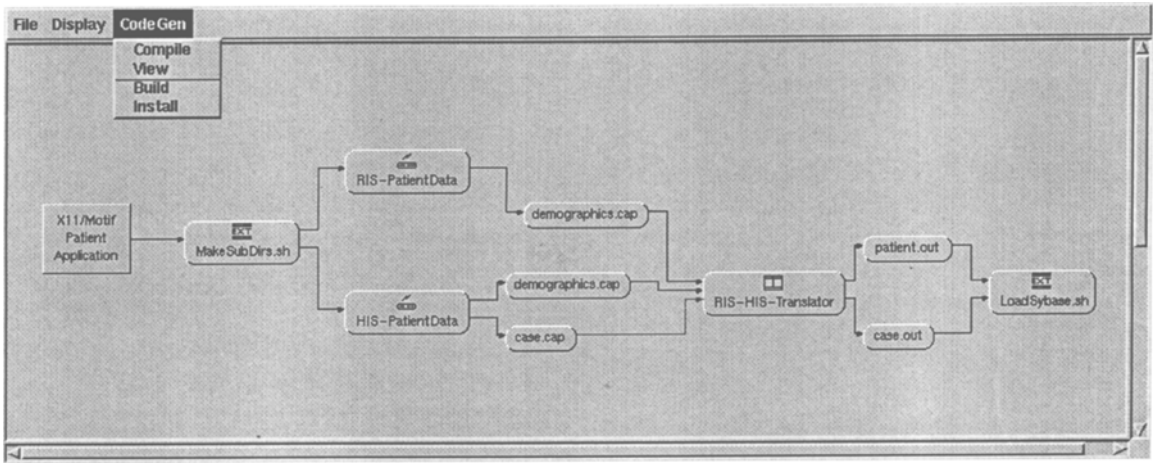

Fig 8. The workflows designer allows users to put together the various communication scripts, intermediate data files, and translation scripts together to view how the entire integration strategy is performed.

RESULTS

Execution Environment

The sequence of actions performed by the Mediator Layer in response to an invocation by a client is as follows (see Fig 9): (1) a front-end query processor accepts the request form from the client. To review, this form contains the patient's hospital identification number, a data range of interest, and specifications of the data items and presentation format for the data needed. It corresponds exactly to the user interface screen shown in Fig 10. The receiving front-end query processor is implemented using the Sybase OPENLIB software. (2) The front-end process first assigns the request a unique identifier and checks the global patient data cache to quickly locate and return any requested records that might have been prefetched for this patient.[20] (3) If not all the desired information is available from the cache, the front-end process develops a run-time plan to retrieve the required information based on the status of the software (eg, server processes) and hardware resources (eg, CPUs and network connections) required to access the data, the data items requested, the partial information that may already be available in the cache, and the capabilities of the registered information processing agents. (4) The front-end query processor then communicates the plan to an Agent Dispatcher process that spawns and monitors the appropriate software agents. The data access agents return patient data in formatted flat files into a temporary workspace where they are then translated, assembled, and inserted into the global data cache. The global cache, a relational data base management system, provides the necessary user views to match the views specified by the initial request form.

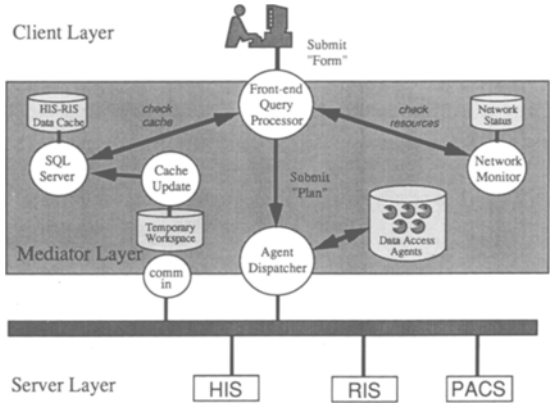

Fig 9. Process and data flow diagram for retrieval of global patient information.

Figure 10 shows our user interface for displaying HIS, RIS, and PACS medical records. The left half of the user screen includes patient demographic information, a selectable list summarizing radiology procedures performed on the given patient, and an area where radiology reports for the selected radiology procedure can be viewed in full. On the right half of the interface, are selectable icons allowing the user to specify the type of HIS or PACS information to be displayed in the central listing area. These

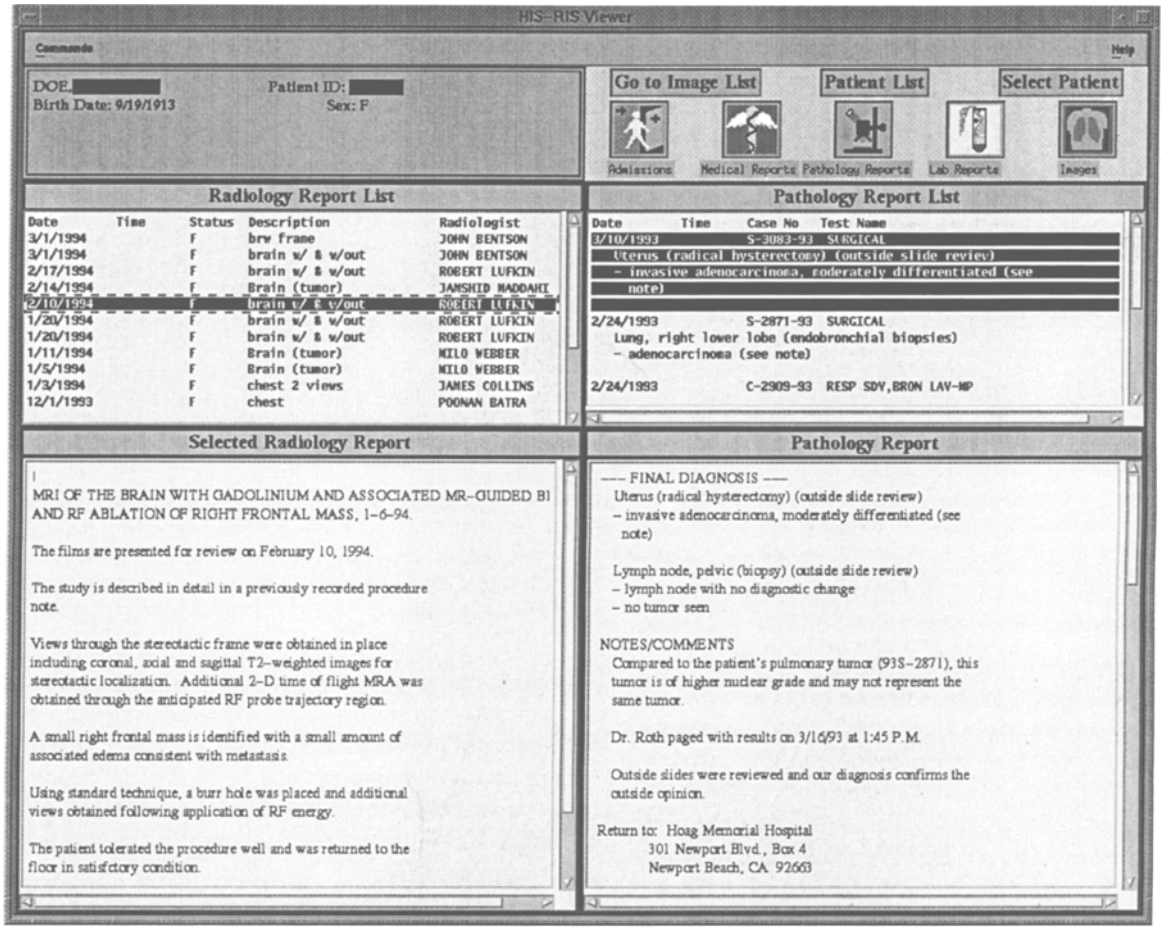

Fig 10. HIS, RIS, PACS interface. The right side includes demographics and radiology reports. The left side provides fast easy viewing of admission records, image lists, and medical, pathology, and laboratory reports.

icons correspond to admissions/discharge information, medical reports, laboratory reports, pathology reports, and PACS images. From the summary listing, the user can select a particular record and the corresponding full-text report can be viewed below. If a PACS image is selected, it can be viewed directly on a selected monitor. The interface has been successfully integrated with a PACS ICU workstation, a commercial Siemens PACS workstation (Siemens Medical Systems Inc, Iselin, NJ), and a research concept-based retrieval workstation.[21]

Performance Profiles

Table 1 lists the code profile and development time for the various subcomponents of our system. The tabulated times include the time spent deciphering keyboard mapping differences between various systems, learning to navigate through the HIS and RIS application programs, and obtaining RIS and HIS schema information. The times do not include the learning time for the CASE tools (about 4 weeks).

The HIS acquisition code was the most difficult piece of software we developed consisting of 1876 lines of SDL code (documentation

Table 1. Code Profile and Development Time for Various Software Components

Software Module Name	Lines of Code (Language)	Development Time (man-mos)
HIS access agent	1876 (SDL)	4.2
RIS access agent	965 (SDL)	2.5
PACS access agent	455 (C)	3.7
Front-end processor	2968 (C)	2.0
Agent dispatcher	Commercial package	—
Cache update process	3269	2.0

Table 2. Number and Size of Various Medical Documents for a Given Patient

Report Type	Avg. No. (Std)	Avg. Size (Std)
Laboratory (HIS)	3.7 (8.3)	1.3 KBytes (2.2 KBytes)
Pathology (HIS)	2.0 (3.3)	3.5 KBytes (4.5 Kbytes)
Medical (HIS)	2.1 (4.2)	2.9 KBytes (3.2 Kbytes)
Radiology (RIS)	1.8 (11.5)	1.1 KBytes (0.8 Kbytes)

inclusive), employing 81 finite states. There were 15 connecting states, 50 data gathering states, 11 error handling states, and 5 exiting states. There are only four control loops within the SDL code. The average amount of data retrieved from HIS per patient was 17.8 KBytes requiring 110 seconds (±167 seconds). The average amount of data retrieved from RIS per patient was 20.0 KBytes requiring 45 seconds. Table 2 summarizes the average number and size of various documents per patient for data retrieved from our HIS and RIS systems. The slow HIS and RIS retrieval performance is because of the screen-capture method used. PACS text data and cached HIS and RIS records can be retrieved in less than 3 seconds because of availability of an SQL server on our network.

DISCUSSION

Current infrastructure research in medical informatics is dominated by the development of communication networks (local area networks, teleradiology, asynchronous transfer mode networks, etc), multimedia display workstations, and hierarchical image storage architectures. However, limited work has been performed on developing flexible, expandable, and intelligent information processing architectures for the vast decentralized image and text data repositories prevalent in healthcare environments. Front-end software programs that can facilitate the retrieval of information from distributed heterogeneous information systems will greatly accelerate the wide-spread approval and use of electronic medical information management systems. The development of "middleware" solutions such as presented here have proven to be flexible, modular, and generalizable. An example from the networking world is the popular Mosaic front-end to the World-Wide Web. Before such facilitating programs, computer-users mainly relied on access to distributed files using primitive programs such as "telnet" and "ftp." These programs were clumsy for non-computer users because such knowledge as internet addresses, usernames, passwords, directories, and filenames of the data were needed. In today's electronic medical record environment, we should similarly not impose on medical personnel to memorize low-level networking and management details. Front-end programs such as Mosaic—together with its underlying architecture—allowed distributed information to be more easily accessed. This is shown by the fact that since 1991, the number of users on the information highway has doubled every 4 months and the amount of resources people are making available is increasing geometrically.[22] We expect that such front-end programs will be increasingly common in medicine, allowing the clinician to concentrate more on diagnosing and treating the patient rather than spending unnecessary time and energy doing laborious information searching and organizing tasks.

CONCLUSION

The CASE tools described herein allow the developer to integrate heterogeneous distributed data base systems in a rapid and evolutionary manner. The creation of modular programmable agents that can be linked to create higher level superagents forms the basis of our work. Our client-mediator-server architecture encapsulates the data and access methods to HIS, RIS, and PACS. Thus, it addresses the issues of hiding the heterogeneous and distributed nature of the data from the user. It also addresses the issues of scalability (the mediator layer of one cluster of data bases can be a client to the mediator layer of another cluster of data bases) and extensibility (mediators agents are modular, composable, programmable, and can be easily upgraded). The architecture allows agents to retrieve data from these systems without the necessity for modifying or adding any foreign software on the host data base machines. Thus, our approach is noninvasive. This is an important feature because most data base administrators are not willing to allow outside developers to jeopardized the autonomy of their systems. Potentially any networked medical data base can be integrated using our described methodology. The CASE tools also greatly facilitate the

documentation, generation, and maintenance of the data integration system software. The majority of the code for the HIS and RIS data access agents were automatically created using the graphical specification tools. This resulted in simpler and higher-level programs that are more easily modified than say C++ code. These high level tools allow complex and large integrated data base systems to be maintained by a modest sized technical staff.

The architecture does suffer from some disadvantages as well. Our "middleware" approach introduces an additional processing layer, thus performance is not optimal using this architecture. Also, it may be somewhat vulnerable to reliability and security problems. Currently, direct access to the data is slow because of the screen capture strategy used. A global data cache has been developed to improve system response performance. Currently, we are developing more active update strategies to the global data cache by using triggers from RIS and PACS events (patient arrival and image arrival, respectively). Also, gateways that provide more direct access to HIS and RIS data are being constructed.

ACKNOWLEDGMENT

We thank Drs Denise Aberle, Theodore Hall, Vikas Bhushan, Zoran Barbaric, and Barbara Kadell for their contributions to the user interface design. We thank Anthony Materna, Frank Stepczyk, and Boyd Hays from Data Integration Solutions Corporation for their assistance on the CASE tools, and Profs Alfonso F. Cardenas and Wesley W. Chu of the UCLA Computer Science Department for their recommendations on the system architecture design. Also, thanks to Hwa Kho and Gregory Tashima from our RIS team for their technical support.

REFERENCES

1. Bakker AR: HIS, RIS, and PACS. Comput Med Imaging Graph 15:157-160, 1991

2. Berbaum K, Franken EA Jr, Dorfman DD, et al: Influence of clinical history on perception of abnormalities in pediatric radiographs. Acad Radiol 1:217-223, 1994

3. Cardenas AF, Pirahesh MH: Data base communication in a heterogeneous data base management system network. Information Systems 5:55-79, 1980

4. Thomas G, Thompson GR, Chung C-W, et al: Heterogeneous distributed database systems for production use. ACM Computing Surveys 22:237-265, 1990

5. Sheth AP, Larson JA: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys 22:183-236, 1990

6. Hurson AR, Bright MW: Multidatabase systems: An advanced concept in handling distributed data, in Yovits, MC (ed): Advances in Computers, vol 32. San Diego, CA, Academic, 1991, pp 149-200

7. Bright MW, Hurson AR, Pakzad SH: A taxonomy and current issues in multidatabase systems. IEEE Computer 25:50-59, 1992

8. Attalur GK, Bradshaw DP, Coburn N: The CORDS multidatabase project. IBM Systems J 34:39-62, 1995

9. Stewart BK, Honeyman JC, Dwyer SJ III: Picture archiving and communication system (PACS) networking: Three implementation strategies. Comput Med Imaging Graph 15:161-169, 1991

10. Taira RK, Stewart BK, Sinha U: PACS database architecture and design. Comput Med Imaging Graph 15:171-176, 1991

11. Huang H, Taira RK: Infrastructure design of a picture archiving and communication system. Am J Roentgenol 158:743-749, 1992

12. Wiederhold G: Mediators in the architecture of future information systems. Computer 25:38-49, 1992

13. van Mulligen EM, Timmers T, van Bemmel JH: A new architecture for integration of heterogeneous software components. Meth Inf Med 32:292-301, 1993

14. Zloof MM: Query by example: A database language. IBM Systems J 16:324-343, 1977

15. Waldrop MM: Software agents prepare to sift the riches of cyberspace. Science 265:882-883, 1994

16. Data Integration Solutions Corporation: Integration Works User Guide, Gardena, CA 1994

17. Lee W, Rowberg AH, O'Leary YM, et al: Integrating a radiology information system with a picture archiving and communication system. Proc SPIE 1234:661-669, 1990

18. Kim W, Seo J: Classifying schematic and data heterogeneity in multidatabase systems. Computer 24:12-18, 1991

19. Nygren E, Johnson M, Henriksson P: Reading the medical record II. Design of a human-computer interface for basic reading of computerized medical records. Comput Methods Programs Biomed 39:1-12, 1992

20. Kohane IS, McCallie DP Jr: A two-tiered database cache mechanism for a clinician's workstation. Proceedings of the Fourteenth Symposium on Computer Applications in Medical Care, Washington, DC, IEEE Computer Society Press, pp 364-368, 1990

21. Taira RK, Rivera M, Bhushan V, et al: Design of a Graphical User Interface for an Intelligent Multimedia Information System for Radiology Research. Proc SPIE: Medical Imaging PACS Design and Evaluation: Engineering and Clinical Issues, 2435:11-23, 1995

22. Berners-Lee T, Cailliau R, Luotonen A, et al: The World-Wide Web. Comm ACM 37:76-82, 1994