The Practice of Informatics

JAMIA

Application of Information Technology ■

Internet as Clinical Information System: Application Development Using the World Wide Web

JAMES J. CIMINO, MD, SOCRATES A. SOCRATOUS, PAUL D. CLAYTON, PHD

Abstract Clinical computing application development at Columbia–Presbyterian Medical Center has been limited by the lack of a flexible programming environment that supports multiple client user platforms. The World Wide Web offers a potential solution, with its multifunction servers, multiplatform clients, and use of standard protocols for displaying information. The authors are now using the Web, coupled with their own local clinical data server and vocabulary server, to carry out rapid prototype development of clinical information systems. They have developed one such prototype system that can be run on most popular computing platforms from anywhere on the Internet. The Web paradigm allows easy integration of clinical information with other local and Internet-based information sources. The Web also simplifies many aspects of application design; for example, it includes facilities for the use of encryption to meet the authors' security and confidentiality requirements. The prototype currently runs on only the Web server in the Department of Medical Informatics at Columbia University, but it could be run on other Web servers that access the authors' clinical data and vocabulary servers. It could also be adapted to access clinical information from other systems with similar server capabilities. This approach may be adaptable for use in developing institution-independent standards for data and application sharing.

■ **JAMIA.** 1995;2:273–284.

System designers usually cite construction of user-friendly graphic interfaces, limitations of hardware platforms, proprietary desktop operating systems, access to information from other systems, and integration with other applications as impediments to the development of clinical workstations.[1] One of the greatest obstacles, however, is that those clinical systems that *are* developed are typically one-of-a-kind, institution-specific legacy systems. So, while many institutions can use the same bibliographic retrieval software, few are able to share applications for displaying laboratory results.

The World Wide Web, a system composed of local browser software and Internet-based hypertext servers, addresses these issues elegantly.[2] Web browsers are available for most popular hardware and operating system platforms and provide access to a variety of servers on the Internet. They employ a standard graphic user interface that can display data files,

Affiliation of the authors: Department of Medical Informatics, Columbia University, New York, NY.

Presented in part at the 1995 Spring Congress of the American Medical Informatics Association, Boston, Massachusetts.

Supported by a High Performance Computing and Communication contract from the National Library of Medicine, a grant from the IBM Corporation, and the Center for Advanced Technology program of New York State.

Correspondence and reprints: James J. Cimino, MD, Atchley Pavilion, Room 1310, Columbia–Presbyterian Medical Center, 161 Fort Washington Avenue, New York, NY 10032. e-mail: james.cimino@columbia.edu

Received for publication: 4/11/95; accepted for publication: 5/16/95.

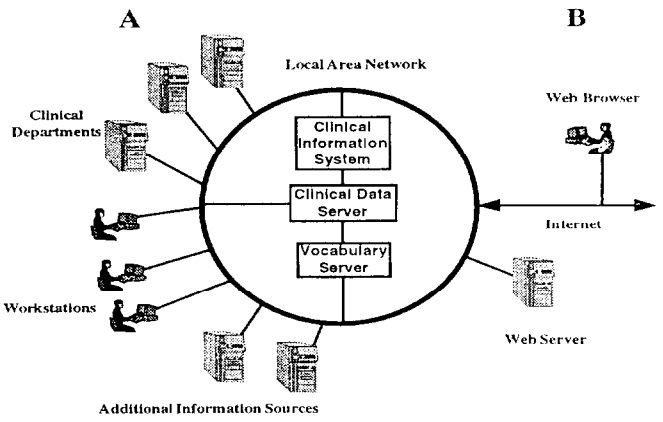

Figure 1 The clinical information environment at Columbia–Presbyterian Medical Center. (**A**) The current arrangement: clinical departmental systems exchange patient data with the central clinical information systems (CIS) via the clinical data server. When coded data are involved, they are translated between local vocabularies and the central Medical Entities Dictionary through the vocabulary server. Clinical users access patient data via terminal emulation to the CIS over the local area network. The network also provides access to other online information systems such as MEDLINE and a variety of text databases. (**B**) The components of the World Wide Web prototype clinical information browser. Web browsers on the Internet access the local Web server. Common Gateway Interface (CGI) programs on the Web server access the clinical data server and the vocabulary server to generate Hypertext Markup Language (HTML) documents that, on the browser, are the display screens for the application. HTML documents also provide links to other applications, both locally and elsewhere on the Internet (not shown).

graphics, images, video, and sound. They can also display hypertext documents that can provide pointers to other resources on the Internet. Sophisticated applications can be built using a collection of interlinked hypertext documents.

The Internet is already supporting numerous medical applications,[3] and the Web is providing a rich environment for development of sophisticated information access.[4] Thus far, emphasis has been on "electronic library" capabilities and medical education,[5–7] but many authors predict that applications on the Internet and the Web will provide health care workers and patients alike with immediate, searchable access to medical information and even consultation.[8–10] Currently, many patient care applications are accessible on the Internet through Internet protocol terminal sessions (TELNET). Some applications have been developed on Gopher servers (which provide menu-based access to information sources) to link clinical information with other information resources.[11,12]

As part of our High Performance Computing and Communications project,[13] we are exploring the capabilities of the Web as a development environment for clinical information systems (CISs). The goal is to develop platform-independent browsing applications that provide health care workers with information about their patients, integrated with access to other medical information sources. Several features of the clinical information architecture at the Columbia–Presbyterian Medical Center (CPMC)[14] are available to support this goal, including a centralized patient database server, which services queries via the Internet, and a controlled vocabulary server, which provides translation capabilities for coded data. Using these features, we have been able to construct a working prototype CIS front end that allows us to explore issues of Internet-based patient care. The prototype makes use of software available commercially and in the public domain, several easily constructed programs for producing Hypertext Markup Language (HTML) documents, and the CPMC clinical data and vocabulary servers. The prototype provides access to CPMC information from virtually anywhere in the world on most common personal computers. With modest adaptation (primarily to customize data and vocabulary queries), it could also access other sites with similar clinical data server capabilities. The prototype therefore provides a model for portable, institution-independent clinical applications. This paper describes our approach and identifies the requirements of servers needed to support it.

Information Architecture at Columbia–Presbyterian Medical Center

Figure 1A shows the core elements of the CPMC clinical information architecture. Ancillary systems communicate with the central patient database via a clinical data server running on an RS/6000 (IBM Corp.), using HL7 messages,[15] to obtain patient information, such as demographic data, and to upload data, such as patient results. The information stored in the central CIS consists of text reports (such as radiology reports, operative notes, and discharge summaries) and coded data (such as laboratory results, medication orders, radiology procedures and findings, and discharge diagnoses). Health care workers access the CIS via workstations (usually IBM PCs using terminal emulation software). The workstations also provide access to other information sources through a variety of menus.[16]

Coded data are managed with the help of the CPMC Medical Entities Dictionary (MED),[17] a controlled vocabulary of over 42,000 coded terms used by CPMC

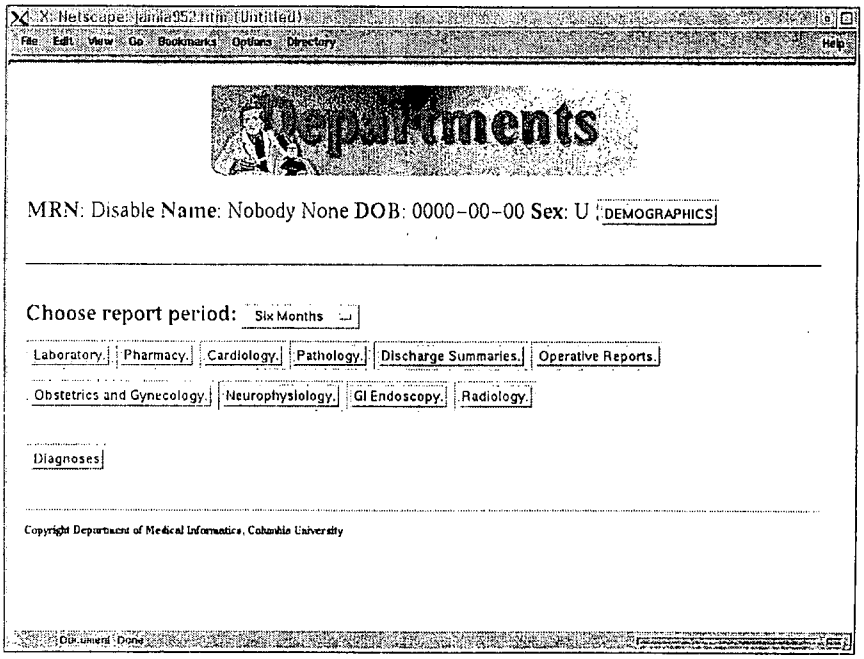

Figure 2 Screen view of the clinical data options available through the Columbia–Presbyterian Medical Center Web-based application. In this case, the user has already selected a patient (whose identifying information has been censored in this screen) and may now select one of the buttons for clinical data. If the user selects "Laboratory," a list of tests performed on the patient will be displayed. Subsequent figures show screens based on the user's selection of a Chem-7 from such a list.

clinical applications. Each term has a unique integer code (its MED code) and a frame-based representation that includes information about its name, synonyms, and ancillary system codes. The frames are organized into a semantic network that relates terms in meaningful ways. For example, the coded term "Chem-7 Glucose Test" is related to other coded terms, such as "Serum Specimen" (through a "has-specimen" link) and "Glucose" (through a "substance-measured" link). The MED also includes a classification structure that allows multiple hierarchies. For example, "Chem-7 Glucose Test" is in the class "Serum Glucose Test," which, in turn, is in two classes ("Glucose Test" and "Serum Test"). A vocabulary server, also running on an RS/6000, provides information that can be used to communicate coded data for upload (e.g., translating ancillary codes into MED codes) and download (e.g., translating MED codes into term names).

The database and vocabulary servers were developed to support communication between the ancillary and

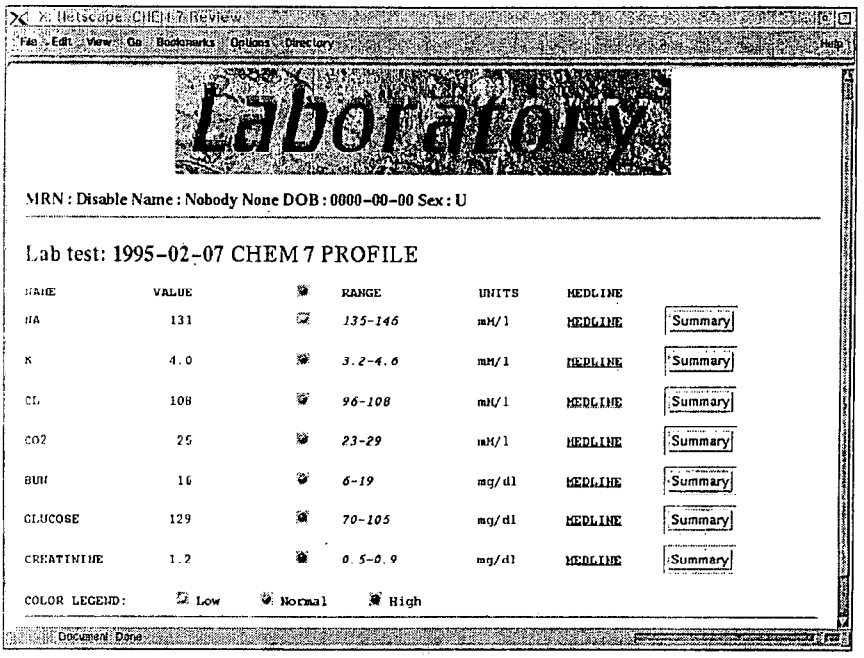

Figure 3 Screen view of laboratory data in the clinical information browser. This is the result of the Web browser's displaying the Hypertext Markup Language (HTML) document shown in Figure 4.

```
<title>CHEM 7 Review</title><center><img src="lab.gif"></center><B><br>

MRN : Disable Name : Nobody None DOB : 0000-00-00 Sex : U<HR>

<h3>Lab test: 1995-02-07 CHEM 7 PROFILE</h3>

NAME        VALUE        <IMG SRC="greenball.gif">    RANGE        UNITS        MEDLINE

<FORM METHOD="POST" ACTION="summary/35456!0000-00-00!">
NA          131          <IMG SRC="yellowball.gif">    <i>135-146    </i>       mM/l
<a href="dmedline/35456">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

<FORM METHOD="POST" ACTION="summary/35455!0000-00-00!">
K           4.0          <IMG SRC="greenball.gif">    <i>3.2-4.6    </i>       mM/l
<a href="dmedline/35455">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

<FORM METHOD="POST" ACTION="summary/35451!0000-00-00!">
CL          108          <IMG SRC="greenball.gif">    <i>96-108     </i>       mM/l
<a href="dmedline/35451">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

<FORM METHOD="POST" ACTION="summary/35452!0000-00-00!">
CO2          25          <IMG SRC="greenball.gif">    <i>23-29      </i>       mM/l
<a href="dmedline/35452">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

<FORM METHOD="POST" ACTION="summary/35450!0000-00-00!">
BUN          16          <IMG SRC="greenball.gif">    <i>6-19       </i>       mg/dl
<a href="dmedline/35450">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

<FORM METHOD="POST" ACTION="summary/35454!0000-00-00!">
GLUCOSE      129          <IMG SRC="redball.gif">    <i>70-105       </i>    mg/dl
<a href="dmedline/35454">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

<FORM METHOD="POST" ACTION="summary/35453!0000-00-00!">
CREATININE   1.2          <IMG SRC="redball.gif">    <i>0.5-0.9      </i>    mg/dl
<a href="dmedline/35453">MEDLINE</A>     <INPUT TYPE="submit" VALUE="Summary"></FORM>

COLOR LEGEND:  <IMG SRC="yellowball.gif"> Low       <IMG SRC="greenball.gif"> Normal
<IMG SRC="redball.gif"> High<HR>
```

Figure 4 The Hypertext Markup Language (HTML) document corresponding to the screen displayed in Figure 3. HTML format marks are enclosed in ⟨⟩. For example, all text between "⟨i⟩" and "⟨/i⟩" will be displayed in italics. The first line of the document defines the title of the document and specifies the graphic to be displayed at the top of the screen (the "Laboratory" logo). The next line contains the patient information to be displayed at the top of the document. The ⟨HR⟩ tag specifies that a horizontal line should be drawn across the screen. Next is the date and name of the test being displayed and the column headers. The next seven sets of lines correspond to the seven lines on the screen showing the results for the components of the Chem-7. Each line is constructed as a "form," containing the text for display, a graphic color ball (green for normal results, red for high, and yellow for low), a MEDLINE link, and a"Summary" button. The last two lines describe the information needed to display the color legend at the bottom of the document. Additional details about the format of the forms, links, and buttons can be found in the text.

central systems. However, we are also using them in the client-server architecture being employed in the design of new user applications.[18] These new applications are being developed to run on mid-range computers and interact with users via character-based (VT100) and graphic (MS-Windows and X-Windows) front ends. Most of the development work has not been on the application software itself, but on addressing front-end display characteristics, communication between client and server applications, and integration between the user application and other remote information sources.

Architecture of the World Wide Web

The World Wide Web includes a set of specifications for display of information, communication between systems, and retrieval of remote information. Software includes a variety of Web browsers (such as Mosaic and Netscape) that obtain information from a variety of servers [including File Transfer Protocol (FTP), Gopher, and Hypertext Transfer Protocol (HTTP) servers] and display the information in a standard fashion. Browsers and servers have been written for hardware platforms running UNIX, Windows, OS/2, and Macintosh operating systems. These browsers and servers interact well without any need to match software or hardware platforms.

Information displayed may be text, graphic, video, or sound. The information may be passive, such as a file obtained from an FTP server, or it may include active links. Gopher servers typically provide information as menus, in which each menu item acts as a link to another menu, to a file, or to an application run by remote access using an Internet terminal emulation protocol. HTTP servers provide information as HTML files (HTML documents), which may have attractively formatted text, graphics, and complex features such as scrollable lists. HTML documents can also include links and buttons. When a user selects a link, the browser retrieves an associated file using an address called a Uniform Resource Locator

(URL) associated with the link.* The file might be on the same server as the file containing the link, or it might be on some other server, somewhere else on the Internet. When a user selects a button, the associated URL points to a Common Gateway Interface (CGI) program that resides on some Web server (again, it may or may not be on the same server). The CGI returns an HTML document that might be predetermined by the application developer, selected from a set of preconstructed files, or generated dynamically. The browser then displays the document to the user. The user perceives this as the browsing of a hyperdocument, while the actual process might be a complex sequence of CGI calls and document generations on a mixture of FTP, Gopher, and HTTP servers around the world. The browsers also provide functions such as printing and encryption. Further technical information can be found in references 2 and 19 and also in online documents.†

The CPMC Web-based Clinical Application Design

Figure 1B shows the architecture of our prototype clinical application, which makes use of both the CPMC and the Web architectures. The heart of the application is a set of CGIs residing on a single Netscape HTTP server (Netscape Communications Corp., Mountain View, CA). The user's workstation, running a Web browser (Netscape Navigator), accesses the "home page" of the application, which offers a number of options for patient lookup. Once the user has identified the desired patient, a set of options are displayed for clinical data lookup (Fig. 2). The user interacts with the application by selecting options, which trigger CGIs. The CGIs on the Web server interact with the clinical data server to obtain patient information and with the vocabulary server to translate coded data. Once the required data are obtained and translated, the CGI constructs on HTML document that is returned to the browser and serves as the next screen for the user interaction. We take advantage of the built-in RSA[20] encryption facilities of the Netscape server and browsers to provide secure communications for authentication, authorization, and privacy.

Figures 3 through 8 show how the application moves from one document (a display of a Chem-7 laboratory panel) to another (a summary of glucose tests). Figure 3 shows how an HTML document appears when viewed with the Web browser. The document generated is the result of a particular sequence of events consisting of button presses on HTML screens and CGI calls: 1) the user selected a patient of interest, 2) a CGI returned a general menu of options for patient information (as shown in Fig. 2), 3) the user selected the "Laboratory" option, 4) a CGI returned a list of laboratory procedures performed on the patient, 5) the user selected a particular "Chem 7 Profile" from the list, and 6) a CGI returned the results for the seven components of the selected Chem-7. The CGI also placed options labeled "Medline" and "Summary" into the HTML document. Figure 4 shows this HTML document. We consider the 'Summary" option in greater detail to show how the Web application responds if the user selects it; the "Medline" option is discussed below.

Each test result line is actually an HTML structure called a *FORM*. At the beginning of the form is the *METHOD*, which indicates that button values are to be "posted" to a CGI, and an *ACTION*, which has the name of the CGI ("summary") and its parameters. The form includes the data and color graphic to be displayed on the screen and ends with an *INPUT* to the form. The INPUT is a button (of type "submit") whose *VALUE* ("Summary") is displayed on the screen. Thus, if the user selects the button next to the "Glucose" result, the value "Summary" is "submitted" to the form to be "posted" to the "summary" CGI, along with the parameter list "35454!0000-00-00!," where the 35454 is the MED code for the glucose tests and the 0000-00-00 is the medical record number of the patient.

The "summary" CGI is written in C; some of the source code is shown in Figure 5. This CGI takes the

*A URL consists of four parts: a server protocol, an Internet address, a file name, and a file label. Server protocols include FTP, Gopher, and HTTP. The Internet address identifies the machine on which the server resides. The file name includes the directory where the file is found on the server. The file label is optional and is used to specify a particular location within a file. For example, the URL for our home pages is http://www.cpmc.columbia.edu/homepages/index.html, which points to a file called index.html in the homepages directory of the HTTP server on the CPMC Web server.

†Documentation about all aspects of HTML, HTTP, and other WEB protocols is available from several online sources. Interested readers who have Web browsers can obtain information from HTTP servers, while those who have only basic Internet access can obtain information from FTP servers. Included on FTP servers are software files for Web browsers. The European Laboratory for Particle Physics (CERN) provides general Web information (ftp://info.cern.ch and http://info.cern.ch), the National Center for Supercomputing Applications (NCSA) provides information about its Mosaic browsers and servers (ftp://ftp.ncsa.uiuc.edu and http://www.ncsa.uiuc.ed), and Netscape Communications provides information about Netscape products (ftp://ftp.netscape.com and http://www.netscape.com). For example, the URL http://www.ncsa.uicu.edu/General/Internet/WWW/HTMLPrimer.html points to a very nice primer on HTML.

MED code of a particular test and queries the vocabulary server in order to determine the class (parent) of the test. The program then generates an HL7 query that requests laboratory data for all test results in that class. The results are returned by the data server as an HL7 message (Fig. 6). The CGI parses this message to create an HTML document (Fig. 7) for display by the Web browser (Fig. 8). In addition to the initial query to the vocabulary server for the class and the query to the data server for the results, the CGI also makes several queries to the vocabulary server for each test result in order to obtain information needed for display (such as test name and units).

The process for moving from the screen shown in Figure 3 to that shown in Figure 8 is fairly typical of the operations carried out by the application. As implied by the options shown in Figure 2, these two screens are only a sample of the clinical information displayed by the prototype.

Use of Internet-based Resources

Because the prototype application is running on the World Wide Web, it is able to access the myriad resources available on the Internet. Since it contains patient information, the application is an ideal environment for exploring methods for obtaining medical knowledge relevant to particular patient problems. As mentioned above, the screen in Figure 3 shows that each test result is associated with a "Medline" button,[21] which is linked to a CGI called "dmedline" (as shown in Fig. 4). The "dmedline" generates

```
main() {

/* The medical record number and test MED code are obtained from the HTML form.      */

    pathenv=getenv("PATH_INFO");
    startmedcode=atoi(getinfo(pathenv,1));
    strcpy(mrn,getinfo(pathenv,2));

/* Calls to the vocabulary server require establishment of a "client handle".  Once   */
/* handle is obtained, the vocabulary server is queried for the parent class of the   */
/* original test.  (If there is more than one parent class, the first is chosen       */
/* arbitrarily.)                                                                      */

    clienthandle=init_qmedRPC();
    return_mcode_parents(&parentlist,startmedcode,clienthandle);
    classcode=parentlist->val;

/* An HL7 message is created, using the medical record number (mrn) and the parent    */
/* class.  This message is an instruction to the vocabulary server to use a           */
/* particular query (PDQRES2) to return test results.  The "C" parameter near the     */
/* end of the message indicates that the results should be for all tests in a class,  */
/* rather than all tests associated with a particular test code.  The function        */
/* hl7sap sends the query to the data server and returns the result.                  */

    sprintf(tempstr,"MSH!^~\\&!socratesqry!wash!resquery!cicsu9!19941125165542!!qry!
    19941125165590!!2.1!!\rQRD!19941125165542!R!I!0113142726!!!99!%s!res!%d!!\rQRF!*!
    19901101000000!19941230170100!PDQRES2~*~OPSTA!95~95~95~PF~~~UR~C~*~*~*~*\rDSC!\r",
    mrn,classcode);
    k=hl7sap("",tempstr,outmsg);

/* In the remainder of the program (not shown), the resulting HL7 message (shown in  */
/* Figure 6) is parsed to create the HTML document shown in Figure 7.  In general     */
/* the process is an iteration through all of the observation records (OBRs) in the   */
/* message.  The MED code for the test, the result of the test and the normal ranges  */
/* of the test are extracted from each OBR.  The vocabulary server provides the test  */
/* names ("slot 12", in the MED) and units ("slot 17" in the MED) for each of the     */
/* test codes, through the function calls:                                            */

    list_val_slots_for_mcode(&testname,12,testmedcode,clienthandle);
    list_val_slots_for_mcode(&units,17,testmedcode,clienthandle);

/* Each test result is added to the HTML document as a line in an HTML table.        */
/* Each record includes the date, testname, value, range and units.  Ranges are       */
/* formatted as bold and italic (enclosed between the HTML marks "<b><i>" and         */
/* "</i></b>") and the units are formatted as italic (enclosed between the HTML       */
/* marks "<i>" and "</i>").  The print commands to create a record are:               */

        printf("%s",date);
        printf("%s",testname->name);
        printf("%s",value);
        printf("<b><i>%s</i></b>",range);
        printf("<b>%s</b>",units->name);
}
```

Figure 5 Some of the source code for the "summary" Common Gateway Interface (CGI), referred to in Figure 4. The code has been edited extensively to show only the portions relevant to server queries and Hypertext Markup Language (HTML) document generation.

```
MSH!^~\&!resquery!cicsu9!socratesqry!wash!19950314151110307!!ORF!19950314151110307!!2.1
MSA!AA!19941125165590!RESULT LIST COMPLETED.!!
QRD!19941125165542!R!I!0113142726!!!99!1644144!res!32309!!
QRF!*!19901101000000!19941230170100!PDQRES2~*~OPSTA!95~95~95~PF~~ ~UR~C~*~*~*~*

OBR!!!M136903542294808^0001!35422^^L!!!19940808071900000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!35456^^L!1^0!138$135-146]mM/l!!!!!!!

OBR!!!M140333542394808^0001!35423^^L!!!19940808071000000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!35456^^L!1^0!136$135-146]mM/l!!!!!!!

OBR!!!X263  3542294807^0001!35422^^L!!!19940807065200000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!35456^^L!1^0!140$135-146]mM/l!!!!!!!

OBR!!!CC86940806600289^0001!33799^^L!!!19940807055500000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!33802^^L!1^0!135!!!!!!!

OBR!!!CC87940806600034^0001!33800^^L!!!19940806070300000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!33802^^L!1^0!147!!!!!!!

OBR!!!CC87940110100550^0001!33800^^L!!!19940110153500000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!33802^^L!1^0!142!!!!!!!

OBR!!!CC02930316220396^0001!1724^^L!!!19930316112200000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!1612^^L!1^0!145!!!!!!!

OBR!!!CC02930120230484^0001!1724^^L!!!19930120124000000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!1612^^L!1^0!142!!!!!!!

OBR!!!CC02920601210491^0001!1724^^L!!!19920601123300000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!1612^^L!1^0!142!!!!!!!

OBR!!!CC02910911230401^0001!1724^^L!!!19910911115000000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!1612^^L!1^0!142!!!!!!!

OBR!!!CC02910225210484^0001!1724^^L!!!19910225123500000000!!!!!!!!!!!!!!!!F!!!!!!!!!!
OBX!!TX!1612^^L!1^0!139!!!!!!!
```

Figure 6 The HL7 message that was returned to the Common Gateway Interface (CGI) in Figure 5 when the user selected the "Summary" button for the glucose test shown in Figure 3. The first four records (MSH, MSA, QRD, and QRF) represent header information about the query. Following the header records are Observation Records (OBRs), each of which has a single Observation Segment (OBX). The OBR provides information about the procedure (such as date and time) with which tests are associated. The OBX has the specific test information, including the Medical Entities Dictionary (MED) code for the test, the result, and the normal ranges. For clarity, only some of the OBRs are shown and several trailing fields of the OBR have been omitted. Note that the query has retrieved results associated with many different procedures (MED codes) and tests (MED codes 33802, 35456, and 1612). This is due to the fact that several procedures (Chem-7, Chem-20, etc.) have glucose tests and each of these tests has a different MED code. Note that the query has retrieved terms from the new laboratory system (1994) and the old laboratory system (1991–1993).

a list of appropriate questions about the laboratory test and returns the list as an HTML document in which each question is a link to another CGI. If a question is selected, its associated CGI initiates a MEDLINE search. We have created similar links to other online resources such as a database on the local Web server containing HTML documents about drug–diet interactions, the *Physician's Desk Reference* on the Columbia Health Sciences Web server, and DXplain on the Massachusetts General Hospital Gopher server.[22]

In each case, the vocabulary server provides the associations between the clinical data being displayed and the medical concepts that are the entries to other information sources. For example, if a user is looking at a Serum Digoxin test result, the MED contains the knowledge that this test measures the substance Digoxin, which is, in turn, a Pharmacologic Substance. This information causes the CGI to include the MEDLINE pharmacology question: "What are the side effects of digoxin?" Similarly, if a user is looking at Chem-7 Glucose Test, the MED contains the knowledge that this test measures Glucose and that Glucose is linked (based on knowledge available from the Unified Medical Language System[23]) to the disease Hypoglycemia. The CGI can then generate the DXplain question: "What are the symptoms of hypoglycemia?"

Discussion

The current information system environment at Columbia is rich in terms of clinical data and additional online information resources.[16] Until now, however, the ability to integrate existing applications and develop new ones has been hindered by the hardware and operating system characteristics (DOS-based workstations and terminal emulation to the mainframe CIS).[24] Application development in the Web environment frees us from worrying about back-end applications and front-end user platforms, and allows

us to concentrate on experimentation with innovative ways to bring applications together.

This freedom, coupled with the availability of powerful data and vocabulary resources, has allowed extremely rapid prototype development. For example, during a recent inspection, the Joint Commission on Accreditation of Healthcare Organizations (JCAHO) indicated that drug–diet interaction information should be made available at nursing stations. We initially considered responding by developing an automated way to display such information on the clinical workstations, based on the patient's medication orders from the pharmacy system. However, the development of such an application on the central CIS would have required several man-months of programming. At the time, we chose to post the information as a separate online resource, so that a user would access the information by noting the patient's medications on the CIS, logging off the CIS, starting the drug–diet application, and manually looking up information about the patient's medications. In the Web environment, however, a more integrated approach is feasible. The MED contains the drug–diet interaction codes provided by the pharmacy system. When a user views a list of patient medications and selects a drug, the application queries the vocabulary server for the interaction code. The drug–diet information is available on the Web server as a set of HTML documents where the name of each document is the interaction code. The MED provides the information needed to create the URL linking the drug term with the interaction information. The addition of this feature required the creation of the HTML documents (approximately 1 man-hour) and the modification of a CGI to include the MED query and link (less than 30 man-minutes).

Security on the Web

One concern with all Internet-based applications is security. Any application that makes clinical information available on the Internet must be prepared to address three issues: authentication (the user is who he or she says he or she is), authorization (the user is allowed to do what he or she is asking to do),

```
<title>Summary Results for test GLUCOSE</title>
<b><h2><center>Summary Results for test GLUCOSE</center></h2><hr>

<pre>DATE                    NAME            VALUE       RANGE       UNITS
<IMG SRC="greenball.gif">

1995-02-07 - 14:03:00      GLUCOSE          129         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=85 HEIGHT=10>

1994-06-01 - 16:25:00      GLUC             137         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=75 HEIGHT=10>

1994-02-01 - 14:02:00      GLUC             142         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=76 HEIGHT=10>

1993-07-19 - 16:38:00      GLUC             107         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=81 HEIGHT=10>

1993-07-19 - 16:37:00      GLUC             100         <i>70-105     </i> mg/dl
<IMG SRC="greenbar.gif" WIDTH=32 HEIGHT=10>

1993-06-15 - 16:44:00      GLUC             104         <i>70-105     </i> mg/dl
<IMG SRC="greenbar.gif" WIDTH=40 HEIGHT=10>

1993-03-02 - 16:28:00      GLUC             101         <i>70-105     </i> mg/dl
<IMG SRC="greenbar.gif" WIDTH=31 HEIGHT=10>

1993-02-03 - 15:31:00      GLUC             113         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=82 HEIGHT=10>

1992-12-29 - 14:12:00      GLUC             130         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=85 HEIGHT=10>

1992-05-26 - 15:19:00      GLUCOSE          142         <i>70-105     </i> mg/dl
<IMG SRC="redbar.gif" WIDTH=76 HEIGHT=10>
<hr>
<FORM METHOD="POST" ACTION="graphsum/10!142.000000!0!/graph21107164.tbl">
<INPUT TYPE="submit" VALUE="BAR GRAPH RESULTS"></FORM>
<FORM METHOD="POST" ACTION="graphsum/10!142.000000!1!/graph21107164.tbl"><INPUT
TYPE="submit" VALUE="LINE GRAPH RESULTS"></FORM>
<FORM METHOD="POST" ACTION="graphsum/10!142.000000!2!/graph21107164.tbl"><INPUT
TYPE="submit" VALUE="BOTH GRAPH RESULTS"></FORM>
<hr></pre>
```

Figure 7 The Hypertext Markup Language (HTML) document that was generated by the Common Gateway Interface (CGI) shown in Figure 5, based on the HL7 result shown in Figure 6. Each result line in the summary includes the date, time, test name (this is not always the same), normal ranges, and units. Also included for each result is a graphic line, which is red, green, or yellow depending on whether the result is above, in, or below the normal range, and which has a length corresponding to the value of the result. The choice of color and length was determined by the CGI code (not shown in Fig. 5). At the bottom of the file are three forms corresponding to buttons that link to graphics functions.

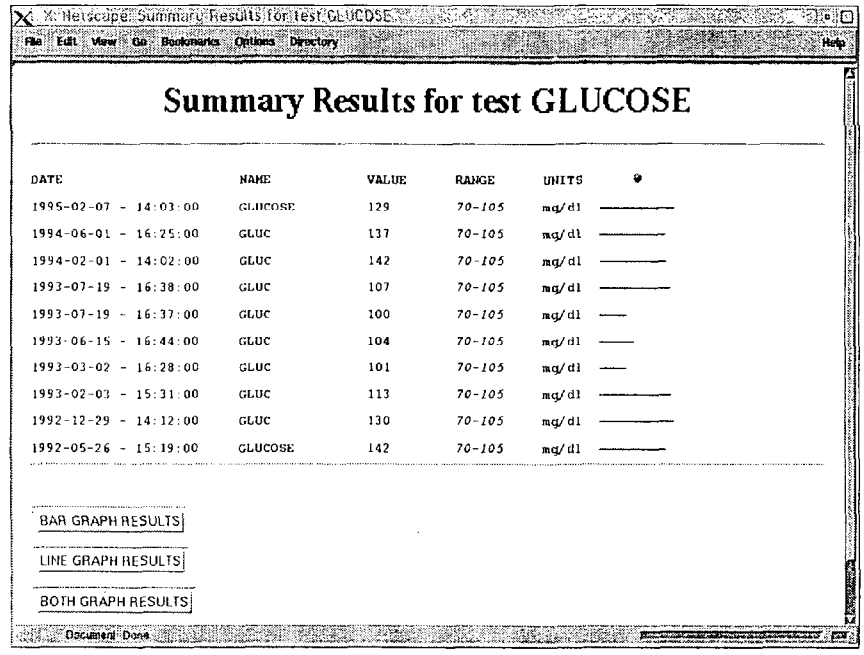

Figure 8 Screen view of laboratory summary data in the clinical information browser. This is the result of the Web browser's displaying the Hypertext Markup Language (HTML) document shown in Figure 7.

and confidentiality (the requested data are given only to the authenticated, authorized user). We take advantage of several features of the Netscape HTTP server to address these three security concerns. The first level of security is through domain restriction: the server application can be defined in such a way that only users operating from a relatively secure set of Internet addresses (e.g., the cpmc.columbia.edu domain at the medical center) are allowed access to the files and CGIs. The second level of security is a typical logon-ID-and-password protection (part of our application, not Netscape), which is used for authentication and authorization. The next level is the use of Netscape's built-in RSA encryption scheme,[20] which encodes all data from the browser to the server with the server's public key (which can be decoded only with the server's private key) and encodes all data from the server to the browser with the server's private key and a session key known only to the browser. Besides the obvious advantage of preventing unencoded patient data from being intercepted, it also prevents interception of the user's unencoded ID and password. Finally, the application establishes a session key (in addition to the RSA session key) that is specific to a particular logon from a particular Internet address. The session key expires when the user selects a "logout" button in the application or during a period of inactivity (currently 5 minutes). None of these security features will absolutely prevent unauthorized access to patient data; however, taken together they provide a formidable obstacle. For example, if someone managed to intercept a screenful of patient data encrypted with the 40-bit RSA key currently in use, and had exclusive access to a computer capable of 64 MIPS, he or she could decode the message successfully in, on average, one year.

Limitations of the Web Paradigm

The Web paradigm for information access currently has several features that may be disadvantageous for clinical application development. One problem is the "connectionless" aspect of the relationship between the browser and the server. Each time the browser executes a CGI on the Web server, a new communication session must be established with the vocabulary and data servers. This minimizes the overhead for the servers associated with maintaining connection for many simultaneous users. However, it may add considerable overhead to the retrieval process for information-intensive applications. Presently, one machine provides all Web and clinical database services for CPMC, while a second machine provides all vocabulary services. With the current arrangement, the main limitations on response time are due to unrelated applications running on the server machines and competing for system resources. If the application is to move beyond the prototype phase, we will need to provide a dedicated machine for Web, data, and vocabulary services and perhaps even distribute these services across different machines. We may also need to develop more powerful queries on the servers. For example, rather than returning HL7 messages with MED codes, the data server might simplify the tasks of the CGIs by interacting with the

vocabulary server directly to perform the translations.

Another limitation of the Web paradigm is the reliance on the HTML format. The limited formatting capabilities in HTML are reminiscent of the first simple word processors compared with the desktop publishing applications of today. However, we believe the platform-independent quality of the HTML standard far outweighs this limitation. Furthermore, if complex displays are needed, they can be created as graphic images, although their transmission is slower than that of ASCII HTML document files. In any event, HTML capabilities are expanding, with the introduction of new format features in the Version 3.0 specifications. Our experimentation has not yet been limited by HTML.

The biggest potential problem with the Web is the hypertext navigation model. As one HTML document leads to another, each document is stored in a "history." Users can back up through the history to return to previous points. However, if the user has done a great deal of browsing, the history may be long and disorienting.[25] Web-based applications often address this problem by including "return" buttons. However, these buttons do not move the user *back* through the history. Instead, they add a copy of the starting document as the next point in the user's *forward* path. The user can still use the Web browser's "Back" button to move back through the history. We see two scenarios where this may be a problem.

One scenario involves data entry. We are extending the current browser to allow the user to maintain patient lists. The user hits an "Add" button to add a patient, enters the medical record number on the next screen, and is then shown the updated list. However, the old list remains in the history. If the user subsequently backs up, the old list will appear and the user might think that the addition did not work. This could result in the user's repeatedly attempting to add the patient to the list, only to be told by the application that the patient is already on the list.

Another scenario involves the issue of authorized access. We can require that the user provide a logon password before starting to access patient data. This password information is necessarily included in one of the documents in the history. The user, when done, might back up to the beginning of the history, but unless another part of the Web is browsed, the next user of the browser can move *forward* through the history to get past the password barrier. Although the logout invalidates the session key (described above), a new session key can be created by moving back to the password screen and then proceeding forward.

Addressing the incompatibility between the hypertext history approach and true dynamic applications is a current research issue. We are experimenting with some solutions. For example, we have addressed the logout problem through the use of a two-step logon sequence. In the first step, a logon screen is displayed. When the user ID is entered, the application generates a password screen that has an internal logon key. When the password is entered, the logon key is invalidated and the session key is created. Once the session key is invalidated (either by the user's logging out or through inactivity), the applications screens, while still showing patient data, will not allow a user to access any *additional* data. Now, backing up to the old password screen will not allow unauthorized use, because the logon key associated with the screen is no longer valid. Logging on requires backing up to the logon screen; when a new logon ID is entered, a new password screen is generated, obliterating the old password screen.

Advantages of the Web Paradigm

Despite the challenges inherent in the Web paradigm, its use for clinical application development offers some clear advantages. First, it exploits the existing CPMC clinical information infrastructure. Creation of the prototype merely required stringing together remote procedure calls and formatting the results to recreate the capabilities of the central CIS. The development time was measured in days, rather than the months needed for similar applications developed on our traditional platforms. Second, the CGIs and HTML documents are portable to other institutions capable of supporting the same query functions. As other institutions and commercial products move to the client-server architecture, our ability to develop Web-based clinical applications at CPMC can be duplicated elsewhere. Third, the screens produced by the system can be displayed on any platform with a Web client that conforms to HTML standards; public domain browser software is available for most computing platforms. Even within our own institution, there is great variability among the users' hardware and software; the Web allows us to satisfy their needs easily. Fourth, the HTML interface can accommodate multimedia clinical and medical information in the displays. Cardiograms, x-rays, and pathology slides, if available online, can be displayed with ease by a Web browser, regardless of platform. Fifth, browsers can be run anywhere on the Internet. This allows us to address user requests for remote access that have heretofore been accom-

plished via modem and telephone connection. Sixth, security and authentication are incorporated easily, as described above. And seventh, the system can include pointers in the display screens that link to information resources available on the World Wide Web, including FTP and Gopher servers.

Implications for the Future

The World Wide Web has opened up many exciting opportunities for application development at CPMC. Even more exciting is the possibility that the adoption of a similar architecture at other institutions might lead to sharing in ways that have been impossible in the past. For example, with the development of Community Health Information Networks (CHINs), there is now a need to transfer patient information between institutions. With an approach similar to our prototype, health care providers at referral centers or in private offices in the community could obtain information needed to support continuity of care from each institution with database and vocabulary servers. The CGIs could reside at the senders' sites, providing standard views of the data, or they could reside at the users' sites to provide customized views. We predict that sharing on this scale will eventually encourage the adoption of standards for vocabulary and database services.

The prototype described in this paper is not a replacement for our current CIS. Additional research is needed to determine how well the Web paradigm will support functions such as entering doctors' orders and notes. Another obstacle is simply that most of the workstations now deployed at CPMC are incapable of running the graphic user interfaces needed to run Netscape. Despite its limitations, the prototype has generated considerable interest among the users. Since the time when we initially submitted this manuscript, a second prototype has been developed using the same data and vocabulary services to provide a service-oriented view of patient data, which differs from the patient-oriented prototype. The new application has been deployed for use by surgical housestaff and attendings[26] and is providing new opportunities to experiment with ways of meeting the information needs of practicing physicians.

Mastery of information resources is becoming a standard requirement for clinical professionals. The American College of Physicians, for example, defines the role of the general internist to include being "a clinical information manager who can take full advantage of electronically stored data and can communicate using the tools of modern technology."[27] Those who would deny this responsibility have heretofore been comfortable in the knowledge that electronically stored data were beyond their practical reach. Our prototype supports the view that this era of abdicated responsibility is, perhaps, drawing to an end.

Conclusion

We have built a working CIS prototype that overcomes previous limitations of location, user interfaces, hardware platforms, and application development. Our approach takes advantage of the existence of readily available resources. Some of these resources, such as the Internet, the World Wide Web, and various online information sources, are publicly available. Two resources, the clinical data server and the vocabulary server, are specific to CPMC, but have specifications similar to servers in use or under development elsewhere. The pairing of the Web with CPMC's infrastructure has unshackled us from many of the tedious chores associated with CIS development. As a result, we are free to enter an exciting new environment where we can explore new ways to display patient data and integrate them with other online resources to address clinician information needs. Other researchers will undoubtedly pursue parallel efforts. Owing to the standards of HTML and the Web, our relationship with these researchers may just as easily be collaborative as competitive.

Access to the Clinical Information Browser

A demonstration of the browser is available to users of standard Web browsers. The information displayed is real patient data, obtained using the CPMC data server and vocabulary server. Due to the potential risk of breach of confidentiality, all queries through this browser result in the retrieval of data from the same anonymous patient. Some of these data have been sanitized so that no personal details are revealed. The URL for the demonstration is http://www.cpmc.columbia.edu/cisdemo.

The authors thank their collaborators in the InterMed project at Harvard University, Stanford University, and the University of Utah for their evaluation, feedback, and stimulating discussion, all via the Internet.

References ■

1. Hammond WE. The role of standards in creating a health information infrastructure. Int J Biomed Comput. 1994;34(1–4):185–94.
2. Schatz BR, Hardin JB. NCSA Mosaic and the World Wide Web: global hypermedia protocols for the Internet. Science. 1994;265:895–901.

3. Kleeberg P. Medical uses of the Internet. J Med Syst. 1993;17(6):363–6.
4. McKinney WP, Wagner JM, Bunton G, Kirk LM. A guide to Mosaic and the World Wide Web for physicians. MD Comput. 1995;12(2):109–14, 141.
5. Metcalf ES, Frisse ME, Hassan SW, Schnase JL. Academic networks: Mosaic and the World Wide Web. Acad Med. 1994;69(4):270–3.
6. Kahn RM, Molholt P, Zucker J. CPMCnet: an integrated information and Internet resource. In: Ozbolt JG, ed. Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care. JAMIA. 1994 Supplement:98–102.
7. Kruper JA, Lavenant MG, Maskay MH, Jones TM. Building Internet accessible medical education software using the World Wide Web. In: Ozbolt JG, ed. Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care: JAMIA. 1994 Supplement:32–6.
8. Kassirer JP. The next transformation in the delivery of health care. N Engl J Med. 1995;332(1):52–4.
9. Galvin JR, D'Alessandro MP, Erkonen WE, Knutson TA, Lacey DL. The Virtual Hospital: a new paradigm for lifelong learning in radiology. Radiographics. 1994;14:875–9.
10. Rizzolo MA, DuBois K. Developing AJN Network: transforming information to meet the needs of the future. In: Ozbolt JG, ed. Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care. JAMIA. 1994 Suppl:27–31.
11. Hales JW, Low RC, Fitzpatrick KT. Using the Internet Gopher protocol to link a computerized patient record and distributed electronic resources. In: Safran C, ed. Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1993;621–5.
12. McColligan EE, Samuell RL II, Jones WT, Moon WA, Pretnar SZ, Johns ML. Providing access to healthcare information resources using Internet Gopher technology as a part of a statewide medical information network. In: Ozbolt JG, ed. Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care. JAMIA. 1994 Suppl:990.
13. Lindberg DAB. Global information infrastructure. Int J Biomed Comput. 1994;34(1–4):13–19.
14. Clayton PD, Sideli RV, Sengupta S. Open architecture and integrated information at Columbia-Presbyterian Medical Center. MD Comput. 1992;9(5):297–303.
15. Johnson SB, Hripcsak G, Chen J, Clayton P. Accessing the Columbia Clinical Repository. In: Ozbolt JG, ed. Proceedings of the Eighteenth Annual Symposium on Computer Applications in Medical Care. JAMIA. 1994 Suppl:281–5.
16. Clayton PD. Integrated advanced medical information systems (IAIMS): payoffs and problems. Methods Inf Med. 1994;33(4):351–7.
17. Cimino JJ, Clayton PD, Hripcsak G, Johnson SB: Knowledge-based approaches to the maintenance of a large controlled medical terminology. JAMIA. 1994;1(1):35–50.
18. Barrows RC Jr, Allen B, Fink DJ. An X Window system for statlab results reporting. In: Safran C, ed. Proceedings of the Seventeenth Annual Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1993:331–5.
19. Branwyn G. Mosaic Quick Tour for Windows. Chapel Hill, NC: Ventana Press, 1994.
20. Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Comm ACM. 1978;21(2):120–6.
21. Cimino JJ, Johnson SB, Aguirre A, Roderer N, Clayton PD. The Medline button. In: Frisse ME, ed. Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care. New York: McGraw-Hill, 1992:81–5.
22. Barnett GO, Cimino JJ, Hupp JA, Hoffer EP. DXplain: an evolving diagnostic decision-support system. JAMA. 1987;258(1):67–74.
23. Lindberg DAB, Humphreys BL, McCray AT. The Unified Medical Language System. Methods Inf Med. 1993;32(4):281–91.
24. Cimino JJ, Sengupta S. IAIMS and UMLS at the Columbia-Presbyterian Medical Center. Med Decis Making. 1991;11(suppl):S89–93.
25. Conklin J. Hypertext: an introduction and survey. Computer. 1987;20(9):17.
26. Cimino JJ, Socratous SA, Grewal R. The informatics superhighway: prototyping on the World Wide Web. In: Gardner RM, ed. Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care, Washington, DC, 1995 (in press).
27. American College of Physicians. The role of the future general internist defined. Ann Intern Med. 1994;121:616–22.