Medical Information Retrieval and WWW Browsers at Mayo

Christopher G. Chute, MD, DrPH
Douglas L. Crowson
James D. Buntrock

Section of Medical Information Resources
Mayo Clinic/Foundation, Rochester, MN

Medical information retrieval from "Master Sheet" entries specially indexed for research retrieval has been part of the Mayo culture since 1909. WWW interfaces have evolved from a novelty to an efficient and widely available "generic client" for distributed client server access to patient information, most practically implemented for historical case retrieval. Providing easy to use and universally available WWW access to these and other patient information databases at Mayo via browsers, shines a bright light on issues of privacy and confidentiality, user authentication, need to know, data transmission security, and technical details of interfacing disparate databases on a spectrum of platforms to many types of workstations using a variety of browsers. We review our recent experience, and generalize pertinent issues. Emphasis is given to the recent rise of platform independent programming enhancements for the client interface, such as Java and JavaScript.

Introduction

We have previously described the information intensive nature of present day health care, and the rapid growth of client server tools which can provide a uniform interface to legacy clinical information systems. The practice of medicine has become increasingly information intensive over the near century during which the Mayo Medical Records have functioned. The historical problems of identifying and retrieving patient diagnoses and procedures was operationally solved at Mayo by the introduction of a paper based "Master Sheet" where a succinct description of all major in-patient and out-patient is maintained. These entries are then indexed, using a variety of computer assisted tools for research and practice management inquiry. Thus, Mayo investigators and practitioners have had access to the repository of patient experience at Mayo as part of our corporate culture; these experiences form the basis of approximately 2,000 papers, reports, reviews, or abstracts published each year.

Inquiries against aggregate patient experience is expanding from the realm of health care researchers to embrace real time health care delivery among quality institutions. This decentralized access to patient information poses many challenges, not the least of which include security and our obligations to protect the confidentiality of identifiable patients. Much excellent work has appeared which describe how generic client browsers using the World Wide Web protocol can enhance the delivery of clinical information at the point of care. The notion of interfacing Web browsers to database has become a commodity resource in the software marketplace. This report overviews our preliminary experiences, and highlights the critical problems we confronted in this process. It adds material on the utility of downloadable generic programming environments, such as JavaScript. Not lost among these technical development issues is our dominant consideration for preserving patient privacy and confidentiality in this new world of ready and faceless electronic access to profoundly sensitive information.

TECHNICAL DEVELOPMENT
The Section of Medical information resources has for several years relied on the powerful prototyping and robust performance of the PERL language. Although it is an interpreted environment, execution speed on modern UNIX workstations is more than acceptable for virtually everything we have implemented. Application development speed is greatly leveraged by the rich system command and statement library of the language, and its interpretive nature. Most importantly, the language can support SQL interfaces to a number of relational database environments, including DB/2, Ingres, Sybase, and Oracle (e.g. the IngPerl extension set). We have substantial experience writing these database interfaces for research and development of information retrieval tools, and thus regarded PERL with SQL extensions as the obvious tool for the generation of database interfaces.

We created a prototype interface, which collects some of our database linkages onto a single home page (Figure 1). Among the advantages of WWW interfaces is the ability to enter users at any point in the Web structure, for example at this common entry point home page, or deeper into the system at a user interface page of direct relevance to them, such as the interface to Medical Index database (Figure 2). HTML readily supports linkage to programs external to the HTPPD server (the program processing requests from the remote client), evidenced by the spawning of a graphical viewer to process an image file. We created PERL scripts which were the target of mouse clicks from the remote Web users (clients) that would spawn and process user inquiries. PERL excels at parsing text strings, a property we leveraged by designing screens that would collect user responses, specify data fields, and query values. Figure 2 illustrates some of the more common variables users tend to specify when asking for the frequency and breakdown of cancer treatment experience at Mayo. The "Submit" button collects the responses on this forms-enabled screen, and passes them back to the HTPPD server. They are then handed to the PERL script, which generates and executes the appropriate SQL code on the fly.

The response to these inquiries can be a complex event. The PERL scripts process the "answer" from the SQL databases, and depending on their content, generate an HTML reply on the fly. These replies can indicate something as simple and static as an error condition, or can respond to the size of data content response and pose another HTML form that asks about refining the query, or otherwise modifying the volume of returned data (just passing on the whole mess is almost always a form option). Additionally, options about data format representation can also be posed as a function of the data, including graphical plots of data frequencies, or formats optimized for printout. The flexibility of the PERL script enables it to pass the SQL response onto a graphing utility (e.g. SAS/Graph with GIF output extensions) or to simple cross tabulation utilities. There is essentially no limit to the elements of inquiry/retrieval processing that cannot be coordinated by these PERL scripts, interacting with the user via dynamically generated HTML screens and a suite of UNIX post-query processing tools.

We are not the first to publish on the utility of PERL interfaces for dynamically generated HTML screens. At least one other group has implemented this approach. They too concluded that the HTML versions which support forms are the minimal level required for practical functionality. As a practical matter, many institutions, including Mayo, have standardized on a single preferred vendor for Web browsing software. Netscape continues to capture the large majority of this market, due in part to its support for the web extensions it pioneered.

**Standards: Still Under Construction**

Confusion over the ability of a given browsers to accommodate specific HTML extension, such as table support, has only grown over the past year. While Netscape products continue to push the
features envelope and dominate the installations in use, developers must continue to exercise caution when invoking “extensions” to the HTML standards. The introduction of the JAVA programming suite by Sun Microsystems, and its explosion among web developers, introduces another dimension to this dilemma. Microsoft only recently endorsed the adoption of JAVA tools into its own Internet strategy. Despite the licensing of Java by Microsoft, the role that Visual Basic might yet play for downloadable applets in browsers remains to be determined by the marketplace. Another variation is the distinction between Java and JavaScript within the Netscape environment. Simplistically, JavaScript is a simpler, interpreted code structure that is embedded into the HTML source being downloaded. Java, on the other hand, is a “p-code” which is compiled by the browser upon download, and treated as an applet quite distinct from HTML. They are similar only in that JavaScript resembles the C++ syntax and naming structure of Java, but does not preserve the classes and inheritance properties associated with Java. For our local application development purposes, we have taken advantage of Mayo’s standardization upon Netscape to use the simpler, though still powerful, JavaScript extensions.

Performance

A question we asked early in the process, was how much performance are we sacrificing for the advantages of a generalizable tool. The answer, somewhat to our surprise is effectively none. We measured response times at several points of the process: WWW to server, PERL execution, SQL execution, and network latency at all segments. The results, in our environment, yielded the following observations:

Network latency from WWW browser to server was invariably negligible.
(Albeit, this was on Mayo's internal network. Outside Internet access to any of these servers is presently prohibited.)

SQL execution was the single largest bottleneck.

For some applications, latency to the mainframe DB/2 databases degraded performance profoundly, to the extent that we simply chose to replicate these data on UNIX Ingres

From these measures and observations, we conclude that we experienced no appreciable degradation in time/response performance using the WWW interface to clinical databases.

The WWW - PERL - SQL approach to remote information retrieval does have its disadvantages; functionality is always compromised by a generalizable tool. Recently, these limitations have been palpably reduced by the introduction of JavaScript extensions. These improvements, coupled with the reduced development time and the unburdening of remote client support, have made our adaptation of browser clients the preferred choice. Specific changes of note include:

Dynamic forms. Fields or contents of a form can be dynamically reconfigured by JavaScript elements, as a function of information entered elsewhere on the form.

While values and field content still cannot be easily propagated backwards or forwards to other pages, they can be shared within forms and the newly available frame structures (windows within a form).

Multi-window communication is now supported, by virtue of the frames mechanism. For example one can now easily browse a code book in one frame, and
use code values in another. This is a satisfactory form of interprocess communication.

Field level validation is now supported, as are range checking criteria. Figure 2. illustrates this behavior, by actively catching a year of diagnosis out of application range, and popping up a warning window.

However, user authorization/verification still cannot be practically accomplished on a field or data element level. The resolution of security and privileged access is at the level of an entire form.

Security

The hardware level of security afforded by Mayo's Internet firewall gave us substantial confidence in performing these experiments. Nevertheless, we permitted no access to patient identifying information whatsoever (see Confidentiality below.)

The useful introduction of WWW technology for confidential patient data must await the release of secure protocols, and the ready availability of browsers that support them. This issue is being driven by the economic requirement for online transaction processing using credit cards and other sensitive financial data. These tools will enhance the security and authentication of legitimate clinical users with a need to know patient information, initially in aggregate form and eventually at the patient specific level. The recent convergence of Secure Socket Layer (SSL) and Secure HyperText Transport Protocol (sHTTP) is a welcome step forward toward achieving this functionality.

Confidentiality

Perhaps deserving a manuscript in its own right, the importance patient confidentially considerations in remote database access cannot be sufficiently emphasized. Historically, Mayo research investigators and clinicians have had free access to patient information, with strongly stated policies that it should be exercised only on a need to know basis. This policy had operated satisfactorily with our unified medical dossier of in-patient and out-patient records for nearly a century.

Electronic access to the same information has many different considerations, the most notable being its relative ease and the perception of anonymity. This later point bears examination, since the reality could not be more different; virtually all electronic access to patient data at Mayo is irrevocably logged. Footprints cannot be avoided using any technology of which we are aware, although the question of whose footprints are left (simulated or real) remains.

In Mayo's emerging Electronic Medical Record System, great attention is paid to user authorization. The physical token/secret combination of a user card (the consultants ever valuable Parking Card with magnetic stripe) and a password serve this purpose. Authentication is scheduled to be undertaken using Kerberos and encrypted information transfer. We will draw upon this model, modified perhaps only to the extent that SSL and sHTTP might replace Kerberos, in our implementation of secure information retrieval tools. All these protocols and security issues pertain to access within Mayo's Internet firewall.

For our Medical Index Retrieval and related research access, we have introduced a password mechanism with our IRB (Institutional Review Board) operations. Now, investigators are assigned a secret password, which they must use to access our index of patient histories. Further, this ensures
that their access to these lists has been reviewed and approved by our IRB, consistent with US DHHS requirements, the pending Bennett bill, and good patient confidentiality practice.

Internet access and security will be addressed very simply: we just say no. The existing firewall will not permit ftp, rlogin, telnet, gopher, or WWW packets to pass. To the extent that we support or allow any Internet access, it is on sentry machines that have stripped kernels and limited binary libraries outside the firewall. The information these machines are permitted will never include patient identifying data.

Conclusion

The emergence of the WWW and ensuing HTML standards has practical and positive consequences on remote client access to clinical databases. Performance impacts at the functional interface level have become greatly reduced by the adoption of JavaScript extensions available in Netscape; there are only negligible reductions in response times attributable to browser interfaces. The development and support advantages which accrue from adopting Web tools for remote database inquiry and retrieval are overwhelming. Security and confidentiality remain the most important issues, and hold promise with the convergence of WWW standards for secure information interchange.

Acknowledgements

Supported in part by NIH grants LM05416, HS/LM08751, and AR30582. We thank Karen Elias for manuscript assistance.

1 This paper is an update of previously presented work: Chute CG, Crowson DL, Buntrock JD, Medical Information Retrieval and WWW Browsers at Mayo, JAMIA, 1995;(Symposium Supplement) 903-907.