Design Considerations In Migrating an Obstetrics Clinical Record to the Web

David A. Berkowicz M.D. M.Sc. M.S., Henry C. Chueh M.D. M.S., and G. Octo Barnett M.D.
Laboratory of Computer Science
Massachusetts General Hospital, Boston, MA.

Recently the American College of Obstetricians and Gynecologists (ACOG) embarked on an effort to promote the development of nationally networked obstetrical records. The Laboratory of Computer Science (LCS) is collaborating with them to help achieve this goal through the development of a web-based prototype of an electronic medical record (EMR) which would allow the entry and display of typical clinical information for the obstetric patient. The process of porting a stand alone application to the web environment necessitated the development of a robust software scheme that could exploit the strengths of Web-based technologies and avoid some of the drawbacks inherent in a stateless environment.

INTRODUCTION

Over the last three years, the Massachusetts General Hospital (MGH) Obstetrics department has used a computer-based record for prenatal care (OBEMR). Through an initiative identified by the American College of Obstetricians and Gynecologists (ACOG), we have been involved in using this experience to provide a demonstration of computer-based records to ACOG membership around the nation.

The stated goals of the initiative are:

- Facilitate creation of a national awareness in practicing obstetricians to the concept of an interactive EMR which uses a controlled vocabulary to standardize and simplify data entry.
- Establish consistent standards for the minimal information that each physician should gather about a patient so that data sharing and analysis can be performed between different institutions or practices.

It was decided that providing a working sample of a computer-based record to clinicians was a critical factor in meeting the above goals. The existing computer-based record used by the obstetric service at the MGH was originally written as a two-tier client server architecture with a graphical application environment on the client[1] (Windows and ToolBook) and a relational database server (Oracle7). It was not realistic to distribute this software to all ACOG members, and therefore a distinct subset of the electronic record was selected for dissemination. A Web-based internet solution was identified as the best approach for this effort. In order to leverage the existing work, the specifications of the web application (OBEMR_WEB) required that it would utilize the same relational data model used in the production OBEMR , and that it would return and present data using a similar visual user interface.

There are many features of a stand-alone application that are difficult to duplicate with a set of HTML pages. Many of these limitations have been discovered in earlier Web-based efforts[2,3,4]. A stand-alone application can leverage the state connection it establishes between the client and the server to send and retrieve data in a seemingly seamless manner, thereby simplifying data persistence and data 'flow-through'. In addition, the editing or entry of information with data validation is a well defined task in this environment. The user interface is easily customizable and keystrokes can be captured to aid in navigation and data entry.

HTML does provide many graphic elements but the interactions between these elements are still relatively static and the dynamic creation of visible components on an existing page is still beyond the scope of today's browsers. The Java language does provide many powerful tools to construct robust user interfaces, but does not approach the ease of development and graphic power of HTML.

An important consideration in design was the targeting of the web program to physicians with a wide range of available resources. A Java based application would take prohibitively long to download over a slow modem and will not run on a 16-bit computer.

Because of these constraints, we designed a component based software architecture that could be delivered over the web through a set of HTML pages, and that could mimic a stand-alone application though the interaction of Javascript with a HTML browser.

1091—8280/97/$5.00

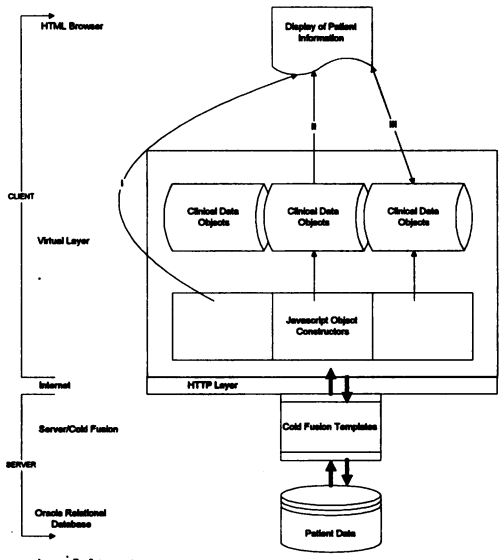

Figure 1. Multitiered Software Architecture.
The patient database is accessed through SQL statements resident in Cold Fusion scripts. The results of the queries are either returned as combined Javascript display functions and data (i), or the raw data is 'piped' through Javascript to form data objects. These clinical data objects can either act as repositories whose contents are retrievable by other HTML pages containing display functions (iii), or the data can be written to the screen by routines contained in the repositories themselves (ii).

METHODS

Overall Design:
The initial design of the OBEMR_WEB consisted of three main components or tiers: the user interface on the web browser, the Web server and its associated common gateway interface (CGI) scripts, and patient data resident in a relational database. However, to duplicate the functionality of a stand-alone application a fourth tier, a 'virtual' layer, was constructed (Figure 1). This was composed of sets of Javascript objects and routines and was designed to provide a local repository of persistent data objects.

Implementation:
The front end or user interface was designed to accommodate users with low connection speeds as well as users with low computer skills, for e.g.: complex graphics were avoided, and no Java objects were embedded in the pages. However, due to the complex design and functionality of the pages HTML frames and tables were used in the layout. Javascript was used to perform on-screen calculations and to guide data input and validation. The layouts of the pages followed the design of the OBEMR at the MGH and were made as simple as possible to use. Wherever possible selection lists were used to control vocabulary input.

The middle layer, represented by Cold Fusion scripts and Javascript objects, consisted of combinations of output templates and SQL statements. Each output template was paired with a set of database queries. This arrangement provided a simple yet powerful way to create multiple views of a database query and was an effective way to reuse query and display components. The queries used for the OBEMR_WEB were the same as those used for the OBEMR and were directed against actual patient data that had been scrubbed of all identifying information. This patient data is resident in a set of tables in an Oracle relational database.

Data Flow and Persistence:
The requirement for data persistence and data flow was addressed by the creation of a 'virtual' fourth layer. This was formed by a dynamic hierarchy of components with different scopes and persistence. The duration of the most persistent data objects was determined by patient selection, i.e.: when a different patient is selected all the data specific to that patient is replaced by the newly selected patient's information.

In general one of the following schemes was used to ensure data persistence and flow through (Figure 1):

i the data arrived bundled with the display methods, or
ii the data was passed up to a parent object which contained the methods to display the information, or
iii only the display methods were loaded and the needed data was retrieved from the persistent data repository.

Alterations or additions to data resulted in the propagation of the changes to the data repository. This persistent data repository was accessible to all visual elements and therefore those components loaded subsequent to the changes displayed the new information. The combination of these schemes allowed the sharing of data between different layouts and allowed each layout to provide its own specific display of the same data.

Clinical Data Object Creation:

Central to this web-based application was the creation of data objects from the patient information returned by database queries. The Cold Fusion CGI returns data as a set of variables output in a loop between specialized output tags. These variables can be captured by Javascript arrays, or objects, and then manipulated in the browser environment through the use of display functions that write HTML directly to the browser window. The same output function can be used to display any comparable Javascript data object, similarly the data object can be displayed in a different format as needed.

Component Architecture:

Framesets in HTML provide a useful mechanism to 'swap' pages into and out of a composite display (Figure 2). These pages can contain data, code, or both and are therefore an efficient means to transfer blocks or modules into and out of the application. The data contained in a page is accessible to all the other pages in the frameset for the duration of the page's display. A single frame in the primary parent frameset was designated to be the data repository whose contents were only flushed when a new patient was selected.

RESULTS

The OBEMR_WEB consists of approximately twenty display pages whose design and functionality closely match that of the stand-alone application (Figure 3). It performs very swiftly, as much of the processing is accomplished locally on the client. Most screens require no more than several seconds to retrieve, even over slow modem lines. The bulk of the data is retrieved at the time of patient selection and therefore most of the subsequent transactions occur between the displayed page and the local repository. A certain amount of load balancing can be achieved by distributing the calls to the database between different display framesets.

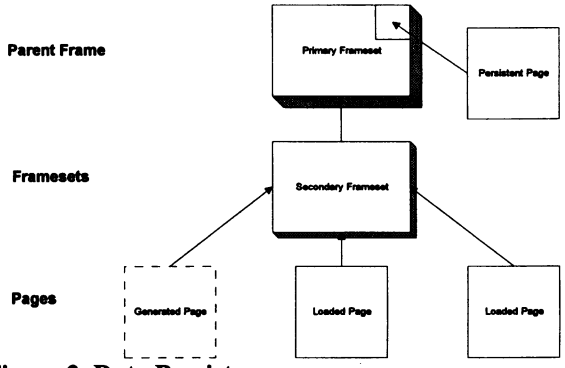

Figure 2. Data Persistence.

A persistent data repository was established as a set of Javascript objects in one frame of the parent frameset. This frame was only replaced when a new patient was selected and therefore the data's duration matched that of the current patient. The other frames in the parent frameset were host to set of secondary framesets each of which contained either dynamically generated HTML pages (dotted box), or HTML encoded pages containing Javascript functions (solid boxes). All the pages in any frameset have access to the data stored in the persistent page.

The program is robust and can operate independent of the network once all the data is loaded. The Web-based application was rapidly developed in four months mostly due to the reuse of components from the stand-alone application. It utilizes the same data model as the production application and therefore all the SQL statements which defined data retrieval could be used unchanged.

The overall visual design of the Web application closely matches that of the clinical workstation currently in use. The physician is presented with a list of patients whose data they may access. When a patient is selected a series of queries are executed. The data returned from these initial queries are stored as clinical data objects in Javascript. When data is reviewed or edited the 'page' performing these actions contains instructions on how to validate the entered data. Also, if data is altered or added then the changes are propagated to the data repository and retained for later submission to the database.

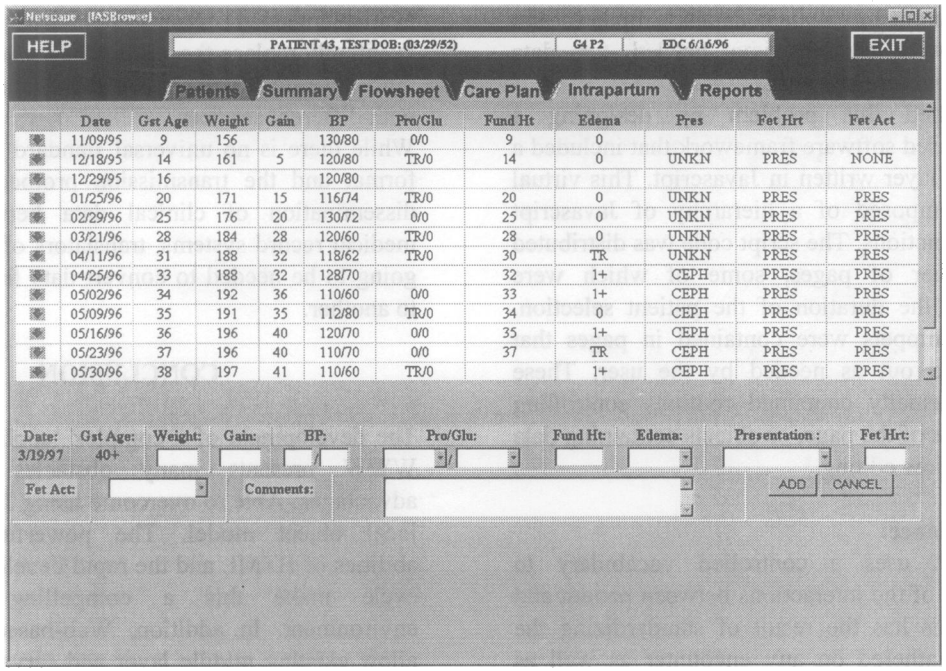

Figure 3.
This screen capture shows a page containing a sample of data. The window consists of many embedded framesets and frames, some of which were created with Javascript functions. This particular page shows flowsheet data summarizing the patient's outpatient visit history. The lower half of the screen contains the entry form where new data can be entered. The new data appears immediately in the upper half of the window once the user 'ADDs' the entry. If the user navigates to another page via the tabs at the top of the screen the new data entered will still be present in the flowsheet.

ACOG members will only be able to reach the site via the college's Web pages, which have restricted access. To facilitate feedback, there is an assessment form available for all users to record their impressions and comments.

DISCUSSION

The aim of the ACOG project is to present an EMR to obstetricians and gynecologists nationwide. The specifications of the project required the user interface to follow the design of an existing stand-alone system presently in use at the MGH. This included input control via pick lists, data persistence, and data flow from one screen to another.

Technology:

The most immediate and intractable obstacle in mimicking a stand-alone application is the stateless nature of the Web which underlies the static nature of displayed pages. Present technologies like Java, ActiveX, and Javascript provide differing solutions to overcome this limitation. None of these provide a complete solution.

However, Javascript is supported by both major browsers (Netscape and Internet Explorer), although JScript (Microsoft) does have some differences to the Netscape version it is possible to write browser independent code. Our work has shown that it is possible to maintain an object model with Javascript and thereby create reusable components. In combination with framesets this property can be exploited to give the illusion of data persistence.

Data Persistence:

Data persistence on the web is difficult to accomplish as information contained on a HTML page is inaccessible once that page is unloaded. Therefore data entered by a physician on a screen is lost to the system once that screen is replaced. This makes the flow of updated information from one screen to another very difficult. One approach to this problem requires the server database to be updated with each alteration in the patient record. The flow of data can then be achieved through repeated database querying.

This method has many drawbacks including the overhead inherent in frequent database updates and queries as well as the time needed for data transmission over the network.
We approached this problem by designing a component based software framework that included a virtual fourth layer written in Javascript. This virtual layer was composed of a hierarchy of Javascript objects and functions. The script code was distributed over a number of pages, some of which were persistent for the duration of the patient selection. Other code snippets were contained in pages that loaded in and out as needed by the user. These components usually contained routines controlling the appearance of the page, or objects containing data that did not allow editing.

Specific Audience:
The OBEMR uses a controlled vocabulary to describe most of the interactions between patient and physician. This has the result of standardizing the information gathered on any encounter as well as simplifying data entry.
The design of an EMR that is targeted to a specific audience provides distinct opportunities. For example, the system contains a prenatal questionnaire with sets of selection lists appropriate for the specialty. In this way a very granular controlled vocabulary can be implemented and an information rich environment created. The Web-based approach is well-suited for this type of layout and requires very little overhead. Another advantage to creating an EMR for a specific audience is that the physicians will be presented with a subject with which they are very familiar and therefore even the most computer naïve should feel comfortable.
This project also explored the notion that a sample EMR can be used as an education device over the Web. The questions contained in the EMR framework provide physicians with a base set of inquiries with which to gather patient information. These questions implicitly define a core data-set and provide guidelines for collecting this information. Ultimately, this can lead to the definition of 'a standard of care' in a given medical specialty.

Future directions:
The OBEMR should be viewed as one of a set of modules that will eventually form the patient's EMR. Communication between these modules will be very important as will be the ability to retrieve data from many sources. The reusable component architecture and expandability of the application can be used to form other modules and will allow for the rapid interchange of elements as they become available/needed. Non-visual Java applets will eventually replace the Javascript because they can be multi-threaded and can establish state connections with different services.
While there is no universal standard for the content format and the transmission protocol used in the dissemination of clinical data between different medical record systems, translators of some type are going to be needed to convert data from one format to another.

CONCLUSION

The development of a complex 'application' for the WWW presents many obstacles as well as advantages. Able to overcome using Javascript and a local object model. The powerful presentation abilities of HTML and the rapid development/display cycle make this a compelling development environment. In addition, Web-based architectures allow existing middle layer and server side services to be reused, shortening development cycles.

Acknowledgements

This work was supported in part by the National Library of Medicine Training Grant NLM LM 07092 and NLM research grant LN05854. We acknowledge the support of Judith Piggins, Dyan Blewett, Michael F. Greene M.D., Frederic Frigoletto M.D. and the ACOG.

REFERENCES

1. Chueh, C., Greene, M.F., Piggins, J.L., Dupont, H.M., and Barnett, G.O. A Clinical Workstation for Obstetrics Replaces the Paper Record and MGH. In Hripcsak, G. ed., AMIA, Spring Congress, pp. 30.
2. Sittig, D.F., Kuperman, G.J., Teich, J.M., et al. WWW-based Interfaces to Clinical Information Systems: The State of the Art. In Cimino J.J, ed., JAMIA, Symp. Sup, pp. 694-698.
3. Cimino, J.J., Socratous, S.A. Just Tell Me What You Want!: The Promise and Perils of Rapid Prototyping with the World Wide Web. In Cimino J.J, ed., JAMIA, Symp. Sup, pp. 719-723.
4. Kittredge, R.L., Estey, G., Pappas, J.J., et al. Implementing a Web-Based Clinical Information System Using EMR Middle-Layer Services. In Cimino J.J, ed., JAMIA, Symp. Sup, pp. 628-632.