Component Architecture For Web Based EMR Applications.

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

ABSTRACT

The World Wide Web provides the means for the collation and display of disseminated clinical information of use to the healthcare provider. However, the heterogeneous nature of clinical data storage and formats makes it very difficult for the physician to use one consistent client application to view and manipulate information. Similarly, developers are faced with a multitude of possibilities when creating interfaces for their users. A single patient's records may be distributed over a number of different record keeping systems, and/or a physician may see patients whose individual records are stored at different sites. Our goal is to provide the healthcare worker with a consistent application interface independent of the parent database and at the same time allow developers the opportunity to customize the GUI in a well controlled, stable application environment.

INTRODUCTION

The integration of clinical computer systems both within and across institutions is a very difficult problem[1,2]. The heterogeneous nature of data storage structures and the myriad user interfaces designed to access them contribute seriously to this problem. Physicians are often required to gain familiarity with numerous applications, and developers are similarly challenged to rewrite code to reproduce functionality. The advent of the World Wide Web (WWW) and the Common Gateway Interface (CGI) has made the process of data retrieval somewhat akin to the process of publishing a dynamic document. The web browser application provides a uniform and familiar user interface and enables the user ready access to review data[3,4]. However, no matter how intricate, web pages are still inherently static and data manipulation is clumsy. Additionally, each web site may have different layouts, color schemes, and organizational plans, features that reduce the utility of a simple site as a viable option for clinical record keeping. In particular, the problems posed by the stateless nature of the hypertext transfer protocol (HTTP) connection require that complex, rapid, and interactive data editing and entry be performed within a web-based application[5]. A web-based application can be defined as a collection of scripted routines (Javascript™ ,VBScript™), with or without Java™ applets or ActiveX™ components, contained within a web page. The application so formed creates an environment inside the browser that can provide mechanisms to overcome the limitations of the stateless connection.

The Laboratory of Computer Science has developed a number of clinical databases and the applications used to access them. Interfaces are being written for some of these legacy systems to allow web access by authorized users. We wanted to simplify the creation of these interfaces by establishing a library of components that could be combined on a web page and function as an interactive electronic medical record (EMR). It was our mission to create a template for component design that would result in the sharing and reuse of software components across different databases and provide a uniform, yet customizable web interface.

DESIGN AND IMPLEMENTATION

A component is an encapsulated functional element that can be used as a building block in application construction. Components can be reused if the format of the data upon which they operate is fully specified, and if a consistent environment for application operation can be provided.

Our approach to realizing the goal of reusable components in the browser environment therefore hinged upon two major challenges:

- The definition of a standard content and format for data transfer between the web server and the browser client.
- The creation of an application programming interface (API) in the client browser that would enable a "plug-and-play" mechanism for application creation.

Content and Format:

Standard content and format for medical data has long been a goal of medical informatics. HL7 was founded in 1987 to develop standards for the electronic interchange of clinical, financial and administrative information between health care oriented computer systems[6]. It is a messaging format with comprehensive content definitions, but it utilizes its own flexible syntax for messages, and it can be difficult to validate the structure of a

A. Chart XML

```
<CHARTS>
        <PATIENT>
            <LASTNAME>PATIENT</LASTNAME>
            <FIRSTNAME>TEST9</FIRSTNAME>
            <MIDNAME></MIDNAME>
            <SEX>M</SEX>
            <PATIENTID>0001</PATIENTID>
            <DOB>6/12/65</DOB>
            <SSN>000000000</SSN>
                <MEDICATIONLIST>
                <MEDICATION>
                    <CONCEPT>Pepcid</CONCEPT>
                    <COMMENT>: Pt taking... <COMMENT>
                    <DATE>12/9/96 7:01:59 PM</DATE>
                    <STATUS>ACTIVE</STATUS>
                </MEDICATION>
            </MEDICATIONLIST>
        </PATIENT>
</CHARTS>
```

B. Component XML

```
<HTMLCONTENT>
      <COMPONENT NAME="myName"></COMPONENT>
      <FUNCTION NAME="init" PARAMETERS="c">
             var listeners = new Array();
             var problemList = new Object();
             var name = "";
             function start(a, aname){

             }
           function doSomeThing(){

           }
      </FUNCTION>
      <HTML>
             <FORM NAME=fTest>
                  <INPUT TYPE=button>
             </FORM>
      </HTML>
</HTMLCONTENT>
```

Table 1. Sample XML files. A. Patient data in the PCF. B. A clinical component.

particular HL7 message. We opted to use the extensible markup language (XML) as the syntax for data exchange, and used the document type definition (DTD) inherent in the XML specification to define and validate data structures. XML is a subset of SGML and has recently been accepted by the W3 consortium as a standard[7]. HL7 has a special interest group (SIG) for SGML/XML representation of health care information and we expect this and other future efforts will contribute collaboratively to consensus standards for XML-based health care data structures.

The content definition we defined for the transmission of clinical data represents a variable time-slice of the patient chart. This Portable Chart Format (PCF) (ongoing laboratory initiative as yet unpublished) is database independent and acts as a packaging medium between the stored data structures and the application components. A simple example of a chart with some patient data can be seen in Table 1, column A. The order of the data elements within a particular level is unimportant as each data element is defined by its wrapping tags. There are a few mandatory elements in the PCF which are related to identifying the patient record (PATIENTID in the above example). Otherwise, the type of data included over a selected time span determines the size of a chart. Further detail about PCF is beyond the scope of this discussion, but will be presented in other publications.

An XML format was also defined for the transmission of software components (Table 1, column B). This enabled the reuse of parser routines and allowed us to leverage a common syntax of communication between client and server.

API

We needed to create a functional shell on the client browser inside of which the components could operate. This shell, or application framework, would supply services such as component loading, data saving, and messaging. In keeping with the idea of reusable objects making up the application, the framework was specifically designed to load and bind components together to form the final program.

Application Framework

The Dynamic Application Builder (DAB) framework, written in Javascript™ , controls component loading and provides essential functionality to the components. The framework manages client server communication through a Java applet so as to provide greater flexibility in dealing with different protocols such as HTTP, Remote Method Invocation, and IIOP/CORBA. This applet parses XML formatted data and generates an XML tree whose elements can be accessed by Javascript™ We used a Java-based XML parser available freely from Microsoft Corp, but any of a number of Java-based XML parsers would have been suitable. The application framework performs a number of sequential operations to instantiate the EMR application that then assumes control of program flow.

Component Loading:

Components are loaded once the HTML page containing the DAB has finished loading into the browser. Components are stored on a web server in the XML format, and their universal resource

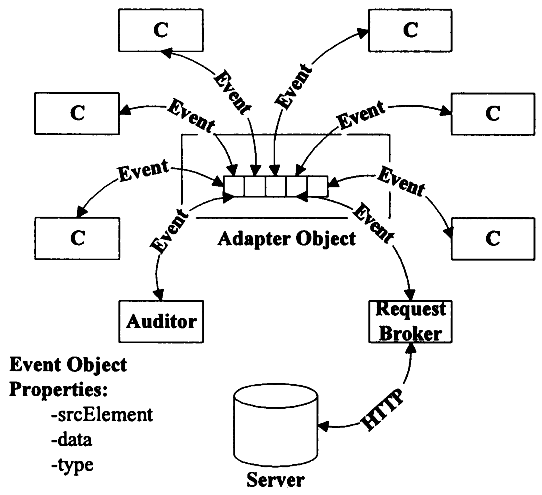


Figure 1. Component Messaging.

locations are conveyed to the application by the values of specific tags on the main page.

Example:

```
<DIV ID="ProblemList"
 URL="http://lcs-jupiter/components/ProblemList.xml">
</DIV>
```

When all the components constituting the EMR application have loaded, they are initialized and bound together by the DAB. The binding operation establishes the relationships between different components and is expressed as a messaging system.

Messaging:
The transfer of data between components is handled by the DAB in an event model similar in concept to that implemented in Java 1.1. Components designate themselves as data listeners for particular event types and event sources. This information is stored in a DAB object, the Adapter, which acts as an event router (Figure 1.). All data is packaged in an Event object that has properties and methods to ensure the accurate targeting and unpacking of information. The Adapter object enables components to function without specific knowledge of the structure of the application. However, it is clear that components registered to receive an event of a particular type need know something of the structure of the data being transmitted. This knowledge constitutes a component specific reference that is separate from and independent of the application framework. Events are also monitored by a DAB component, the Auditor, which tracks changes to data so that an accurate record of user actions can be maintained (Figure 1.).

Components:
The component model we defined requires the creation of tightly encapsulated independent objects that receive and transmit data through a standard interface. The internal processing of data is component specific and does not rely on the existence of any other code. Components also adhere to an internal design pattern and are thus similar in concept to the JavaBean™[8]. Components may extend the functionality of the API (application components), or they may interact with data (clinical components). Clinical components can be functionally divided into two broad classes, those that interact directly with the XML data tree (data components), and those that do not (visual components). The data components are responsible for the extraction of data from the XML document and therefore some knowledge of the PCF is needed. Any changes to the chart structure would mandate a change in these components. The second group of components provides a visual representation of the data that they receive from the data components. The visual components are thereby insulated from any changes in the PCF.

RESULTS

The overall architecture of the application can be seen in Figure 2. The application framework (DAB) runs in a web browser and communicates with the server through the Java XML parser applet. Components are dynamically loaded via the applet and bound into an EMR application. Clinical data in

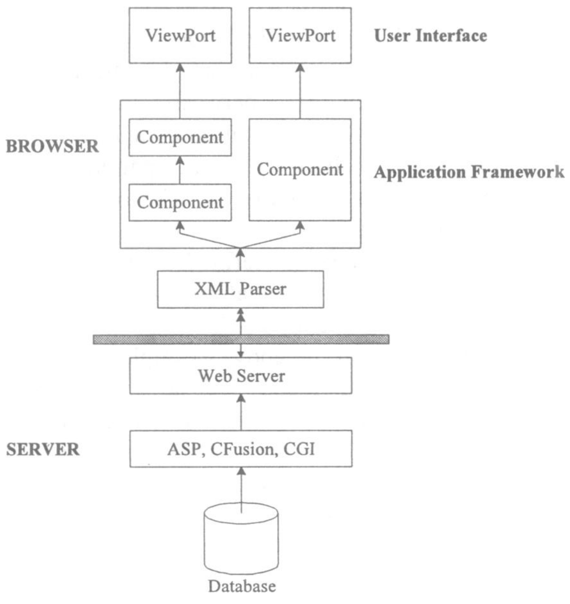


Figure 2. Overall Architecture.

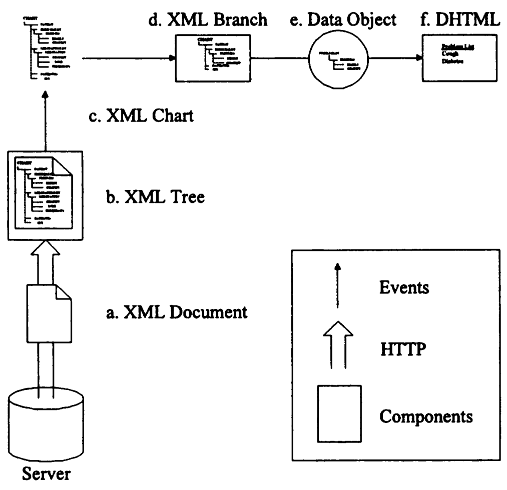


Figure 3. Data Translation.

the PCF is retrieved by the applet on user request and made available to the EMR components. Components may display data themselves or they may transfer the data to other components. Components that display data use Dynamic HTML to write directly to select portions of the screen. We called the area into which a component targets HTML a viewport. The specific appearance of each viewport was easily modified through the use of HTML style sheets. The database can take any structure and there is also no mandate for the type of web-server or CGI as long as the document transmitted contains data that conforms to the PCF. In our sample applications the middle layer was developed in the Microsoft Active Server™ environment.

Data Flow

A sequence of data transformations occur when a patient's chart is selected for review. Data is extracted from the database and wrapped with XML tags to form the PCF. This is sent as an XML document from the server to the client via HTTP (Figure 3, a.). This document is parsed by the Java applet into an XML tree (b), which is then retrieved and cached by the Chart object (see below) (c). The Chart object broadcasts an event to a clinical component (d) that further pares down the data and generates Javascript data objects (e). The data objects are relayed to the visible clinical components that translate them into HTML strings that are written to the screen (f).

Clinical Components

A number of components have been created and successfully tested in the DAB. These include components to handle problem lists, medication lists, allergy lists, past medical and social history, patient lookup, chart management, and visit notes. Figure 4 is a screen capture showing two views of problem list data. Two identical visual clinical components' viewports were placed side by side to highlight different aspects of the component architecture:

a. Both views are synchronized in the display of a deleted item; this is a reflection of a common data source.

b. The component on the left has sorted the problem list by date and the problems appear in a different order to those on the right. In addition, the component on the right displays a collapsed view of the list. This demonstrates the encapsulation of function within a component and the internal handling of user events.

API Extensions

A Chart management object was created to enable off-line browsing of clinical data. Downloaded PCF documents are cached by this object and all requests for patient charts are first routed through it before going onto the server. An Auditor object was created to monitor all data modifications. These changes are stored in the form of an XML tree whose structure closely mirrors the PCF.

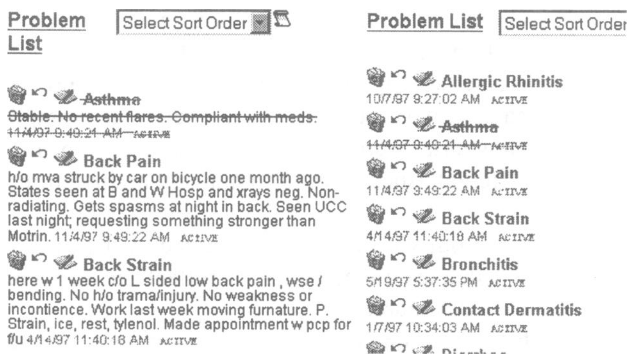
Applications

Boston Health Care for the Homeless Program (BHCHP) is an organization that uses an EMR to manage patient data[9]. We have successfully used our components to build a web-based application to retrieve existing patient information from their databases. We are working with them to create a full-featured web-based EMR.

We have also used the same components to retrieve clinical information from two significantly different production EMR systems contained in Oracle databases, and we were also able to retrieve clinical data from a MUMPS database[10]. In the latter example, we reused server-side scripts intended for static HTML pages by making minimal changes that added the XML syntax. In each of the above cases we constructed web applications with distinctly different visual interfaces by using different component combinations.

DISCUSSION

Standard browser technology was leveraged to create an extensible framework in which reusable components could be assembled to create an EMR.



~~Asthma~~
~~Stable. No recent flares. Compliant with meds.~~
~~11/4/97 9:40:21 AM ACTIVE~~

Back Pain
h/o mva struck by car on bicycle one month ago. States seen at B and W Hosp and xrays neg. Non-radiating. Gets spasms at night in back. Seen UCC last night; requesting something stronger than Motrin. 11/4/97 9:49:22 AM ACTIVE

Back Strain
here w 1 week c/o L sided low back pain , wse / bending. No h/o trama/injury. No weakness or incontience. Work last week moving furnature. P. Strain, ice, rest, tylenol. Made appointment w pcp for f/u 4/14/97 11:40:18 AM ACTIVE

Allergic Rhinitis
10/7/97 9:27:02 AM ACTIVE

~~Asthma~~
~~11/4/97 9:40:21 AM ACTIVE~~

Back Pain
11/4/97 9:49:22 AM ACTIVE

Back Strain
4/14/97 11:40:18 AM ACTIVE

Bronchitis
5/19/97 5:37:35 PM ACTIVE

Contact Dermatitis
1/7/97 10:34:03 AM ACTIVE

Figure 4. Contrasting Viewports.

The component architecture enables application designers to build complex web-based applications rapidly with a visual editor. Since the technologies being used are W3C standards, we were able to use commercial web editing software instead of designing a custom layout editor. Web technology is evolving to support more complex applications and the use of consistent data markup allows for the reuse of software across differing databases. Components can be used to maintain a consistent functional interface between applications, as well as providing flexible use within an application[11].

The DAB framework can be easily extended to provide site-specific functionality to the EMR. For example, we created a component, the Request Broker, to help manage communication between the client and the server. This object contains a set of routines explicitly tailored for a particular web-server and its role is to translate all client requests into a database specific call. Any client application wishing to retrieve data from a participating site, need only obtain that site's Request Broker in order to enable access. This means that site-specific security information and other communication protocols can be hidden from the developer and shielded from the rest of the application.

CONCLUSION

Our current plan is to finish the library of components needed to construct a complete EMR. These will be used to create the next version of the BHCHP EMR. This application will be deployed and evaluated in both an ambulatory and inpatient environment.

ACKNOWLEDGEMENTS

This work was supported in part by the National Library of Medicine Training Grant NLM LM 07092, NLM research Grant LM05854 and a research grant from Hewlett Packard.

References

1. Kohane IS, van Wingerde FJ, Fackler JC, et al. Sharing Electronic Medical Records Across Multiple Heterogeneous and Competing Institutions. Proc of the 1996 AMIA Annual Fall Symposium. 1996: 608-612.
2. Halamka JD, Safran C. Virtual consolidation of Boston's Beth Israel and New England Deaconess Hospitals via the World Wide Web. Proc AMIA Annu Fall Symp 1997;:349-353.
3. Kohane IS, Greenspun P, Fackler J, Cimino C, Szolovits P. Building national electronic medical record systems via the World Wide Web. J Am Med Inform Assoc 1996 May;3(3):191-207.
4. Cimino JJ, Socratous SA, Clayton PD. Internet as clinical information system: application development using the World Wide Web. J Am Med Inform Assoc 1995 Sep;2(5):273-284.
5. Berkowicz DA, Chueh HC, Barnett GO. Design considerations in migrating an obstetrics clinical record to the Web. Proc AMIA Annu Fall Symp 1997;:754-758.
6. Health Level-7 Standards. http://www.mcis. duke. edu /standards/ HL7/hl7frontpage.htm.
7. World Wide Web Consortium. The World Wide Web Consortium Issues XML 1.0 as a W3C Recommendation: XML 1.0 Fact Sheet 1998. http://www.w3.org/Press/1998/XML10-REC-fact.html.
8. JavaBeans™ Component APIs for Java. http://www.javasoft.com/products/jdk/1.1/docs/guide/beans/index.html.
9. Chueh HC, Barnett GO. Client-Server, Distributed Database Strategies in a Healthcare Record System for a Homeless Population. Journal of American Medical Informatics Association. 1994;1:186-198.
10. Rabbani U, Morgan M, Barnett GO. A COSTAR interface using WWW technology. Proc Annu Symp Comput Appl Med Care. In press 1998.
11. Chueh HC, Raila WF, Pappas JJ, et al. A Component-Based, Distributed Object Services Architecture for a Clinical Workstation. Proc of the 1996 AMIA Annual Fall Symposium. 1996:638-642.