Using HL7 and the World Wide Web for Unifying Patient Data from Remote Databases

drs. F.J. van Wingerde[1], Jiri Schindler[2], Peter Kilbridge M.D.[3], Peter Szolovits Ph.D.[2], Charles Safran, M.D., M.S.[4], David Rind M.D., M.S.[4], Shawn Murphy, M.D., Ph.D.[3], G. Octo Barnett, M.D.[3], Isaac S. Kohane M.D. Ph.D.[1]
[1]Children's Hospital Informatics Project, Children's Hospital, Boston, MA
[2]Clinical Decision Making Group, Massachusetts Institute of Technology, Cambridge, MA
[3]Division of Clinical Computing, Beth Israel Hospital, Boston, MA
[4]Laboratory of Computer Science, Massachusetts General Hospital, Boston, MA

W3-EMRS is an architecture designed to access clinical data from remote heterogeneous electronic medical record system (EMRS) databases. We describe the technologies used in an experimental implementation of W3-EMRS that concurrently collects data from several sources and presents them in an integrated set of views. After describing some of the organizational constraints, the architectural decisions, implementation methodology, and operation of the completed project are discussed.

INTRODUCTION

We have previously described[1] a demonstration project in which a single EMRS database at Children's Hospital, "scrubbed" to remove patient and provider identifiers, could be accessed over the World Wide Web (W3). The functionality delivered by the original prototype led to its re-implementation, running against the entire Children's Hospital Integrated Hospital Information System[2]. Access to this implementation is restricted to the hospital's "intranet", within its "firewall." Like other W3 based EMRS (W3BE) implementations[3,4], these early prototypes were limited to providing access to a single EMRS database because several functions of these W3BE depend on properties particular to the underlying EMRS (the "legacy" EMRS), particularly the legacy EMRS's information model. The W3-EMRS architecture provides independence from the specifics of any particular legacy EMRS by creating several abstractions, a set of data structures and functions common to multiple legacy EMRS. These abstractions include 1) a Common Medical Record (CMR) which describes the information model, vocabularies and transactions common to multiple legacy EMRS 2) a Visual Presentation layer which describes the layout of data elements (e.g. a time-ordered list for patient medications) in the CMR and responses to user interactions (e.g. selection of a medication) 3) a Screen Rendering layer which implements the Visual Presentation in a particular user interface technology (e.g. W3, Visual Basic, Delphi). Details of the W3-EMRS architecture can be found in Kohane et al.[2].

This paper describes an implementation of the W3-EMRS architecture, designed to deliver an integrated view of the patient record from multiple legacy EMRS, for the purpose of initial evaluation and treatment of a patient in an Emergency Department (E. D.). The data for this feasability study came from three scrubbed databases of patient data, and all connections between separate computers for the purpose of data-sharing were made inside the lcs.mit.edu domain. We emphasize how this implementation was informed and constrained by the lessons learned in implementing the first W3BE prototypes and by the necessity to adhere to industry standards.

One of the consequences of implementing a multi-institutional W3-EMRS architecture was that several important issues related to sharing data between heterogenous legacy EMRS were encountered. These include the absence of well-accepted standard vocabularies, heterogenous confidentiality policies and technologies, the lack of master patient indices, and disparate semantics even for identical terms. These issues are beyond the scope of this paper and are addressed in another paper in these proceedings [5].

DESIGN CONSTRAINTS

The design constraints for data sharing in this project were significantly influenced by the nature of the Boston EMR Collaborative. The hospital-based EMRS, at Children's Hospital[6], the Massachusetts General Hospital[7], and the Beth Israel Hospital[8], all have different information models, styles of clinical documentation, medical record numbering systems and vocabularies (e.g. for problem lists). Further, concerns of patient privacy, intellectual property, and potentially increased computational burden imposed on the legacy EMRS dictated that unrestricted, direct querying of any of the legacy EMRS database was to be ruled out. Each legacy EMRS would have to be able to control how it responded to queries external to the EMRS. Finally, to allay concerns of all parties in this experimental phase, we have only used test databases from each hospital, processed to remove all identifying

0195—4210/96/$5.00

information with subsequent manual review by two teams. These test databases do not reside within the hospitals and do not therefore threaten the confidentiality of any of the patients in the "production" databases.

The task domain for the first experimental prototype of the W3-EMRS architecture for the Boston EMR Collaborative was defined as providing clinical information for the management of the patient arriving in the Emergency Department of any of the hospitals of the Collaborative. This domain was felt to be more compelling motivation for the use of high-speed networks than less acute multi-institutional scenarios, such as a referral to a specialist for a second opinion which might be equally well served by other technologies. The selection of a well-defined task domain also had the effect of limiting the breadth of data modeling and consensus building to the minimum required to meet the specified task.

Use of Standards

The first project taught us that the W3 is a viable medium for transporting medical data. It also showed us in a compelling way that adhering to standards and using off-the shelf components as much as possible, resulted in substantial gains in time between conception and release: coding the first instance of that project took all of fourteen days.

Since accessing multiple databases would make this project more complex, it seemed logical to again adhere to standards and use off-the-shelf components as much as possible to manage the increase in complexity.

Consistent Interface

If all participating sites would deliver their data in different electronic documents, it would be difficult for the user to piece together the full medical history of a patient that had data spread over several sites. There was also the possibility that every site would create a different interface to their medical records, thus increasing complexity for the user as the user would have to learn how to interact with every site to retrieve the desired data. Furthermore, the user would have to remember where to find the points of access for every site.

To make the system easier to use, the decision was made that the data would be retrievable from one single point of access. It also would be made possible, once the user had selected the relevant patient, to present the data on that patient as one single unified electronic document, no matter which site the data came from.

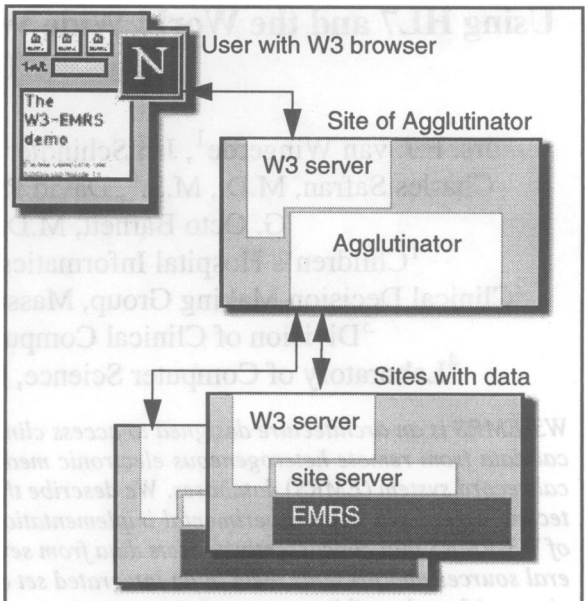

Figure 1 - Overview of the architecture

INSTANTIATION OF ARCHITECTURE

To satisfy the above conditions, an architecture was created, consisting of three elements. The three elements roughly correspond to the three tiers of W3-EMRS abstraction.

W3 Browser

The first module in this architecture is the user with a browser capable of connecting to and viewing data on the W3. It is assumed that the user is working on equipment adequate for viewing large amounts of medical data (i.e. a high-resolution color monitor of 17 inches or more attached to CPU capable of running Netscape 2.0 or an equivalent browser). We felt these requirements were not unreasonable as Netscape browsers comprise a significant part of the market for all the major platforms and hardly any desktop computers are sold anymore without color capabilities. This corresponds to the Screen Rendering Layer of W3-EMRS.

The Agglutinator

The program that mediates between the multiple EMRS sites and the user with a W3 client is called the Agglutinator. This program collects clinical data from multiple sites, formats these data in a data-type specific manner, generates the corresponding HTML, and returns the output to the user connected to the Agglutinator. To collect the medical data from various sites, the Agglutinator uses the same infrastructure as used to connect to it: W3 protocols over the Internet. The Agglutinator is capable of connecting through the W3 with several dedicated programs ("Site Servers") at participating EMRS sites, in order to collect data from

those sites. One or more Agglutinators can run at different sites to manage different user loads and network configurations. In the current experiment, we only use one. The Agglutinator corresponds to the Visual Presentation layer of the W3-EMRS architecture as it converts a set of message streams from the various EMRS sites into a defined presentation format.

In order to serve data back to the browsers as quickly as possible, the choice was made to have the Agglutinator run continually. This saves the time used for internal initialization if the Agglutinator would have to be started separately up for each request.

The connection between the W3 server and the Agglutinator is a small gatekeeper program that the server starts up when a client requests a connection to the Agglutinator. The gatekeeper tries to make a connection to the Agglutinator with a standard TCP call to the Central Access Point (CAP), a TCP socket, of the Agglutinator. When such a request comes in, the Agglutinator will open a specialized communications channel to the gatekeeper and then clone itself (a so-called 'fork'), a duplication that does not involve significant overhead and maintains all communications and internal structures in both Agglutinators. The spawned clone, the child, can then take care of the request and return the results to the gatekeeper over the specialized channel, while the parent is free to serve the next request.

When the Agglutinator is started up, it reads in a configuration file that contains a list of all participating site servers, and puts them in an internal list. It also constructs a generic HL7 query in which certain slots have been left open for the details of the query.

The Agglutinator-clone fills in the rest of the query with the details as received from the user, and sends it over a W3 connection to the sites in its internal list of sites. The internal structure of the Agglutinator allows for restrictions on which site servers should be contacted, so as to not waste resources if it is known that information can be found at specific sites. After sending, it will then poll the contacted sites every second until all sites have returned a response or until a certain time-out value has been exceeded, currently set at 20 seconds. The Agglutinator will then decode each response for information and build an HTML page from the responses, which it will then send to the gatekeeper over the specialized channel. The gatekeeper can then pass the page on to the Web server for transport to the browser that issued the first request.

This design allows for an Agglutinator that is almost continually accessible but can still handle multiple queries concurrently by spawning copies to each take care of every request.

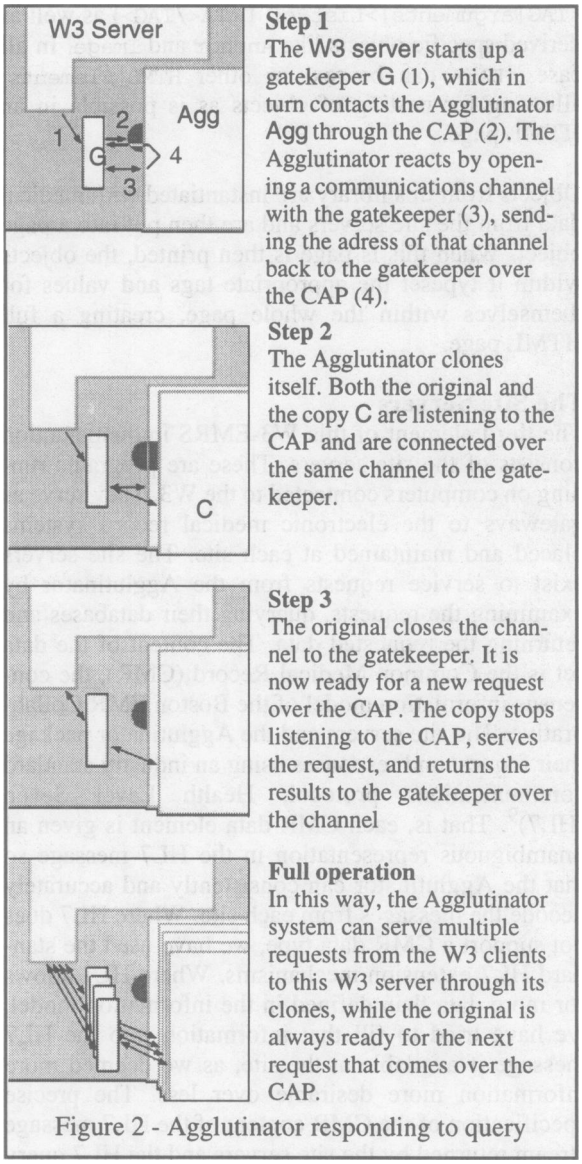

Figure 2 - Agglutinator responding to query

The Agglutinator is coded as a C++ program. It is built from objects from the lowest level up; the connections to the site servers, the requests, the returned data, the HTML page and the elements within it are all internally represented as C++ objects. This allows for easy exploitation of the similarities between the different types of medical data (allergies, problems, demographics, etc.) and a decomposable inner architecture for easier creation and maintenance.

At the time of writing, there were no C++ libraries implementing HTML available to us, so we wrote our own. The HTML library contains an abstract class `HTML_Element` from which all elements of an HTML page are eventually derived.The library contains some general objects like tagged element (of the format `<TAG[arguments]>DATA</TAG>`) and list tagged element (of the format

<TAG[arguments]>List of DATA</TAG>) as well as derived specific objects like Anchor and Image. In all cases, DATA can be text or other HTML_Elements, allowing for nesting of objects as is possible in an HTML page.

Objects from this library are instantiated with medical data from the site servers and are then put into a page object. When this is page is then printed, the objects within it typeset the appropriate tags and values for themselves within the whole page, creating a full HTML page.

The Site Servers

The third element of this W3-EMRS implementation consists of the site-servers. These are programs running on computers connected to the W3. They serve as gateways to the electronic medical record systems placed and maintained at each site. The site servers exist to service requests from the Agglutinator by examining the requests, querying their databases and returning the requested data. The content of the data set is the Common Medical Record (CMR), the consensus information model of the Boston EMR Collaborative. The site servers and the Agglutinator package their request and responses using an industry standard communications protocol, Health Level Seven (HL7)[9]. That is, each CMR data element is given an unambiguous representation in the HL7 message so that the Agglutinator can consistently and accurately decode the messages from each site. Where HL7 does not support a CMR data type, we have used the standard HL7 extension mechanisms. Where HL7 allows for more data than defined in the information model, we have tried to fill that information into the HL7 message if available at the site, as we deemed more information more desirable over less. The precise specification of the CMR content of the HL7 message stream returned by the site servers and the HL7 query that the site servers respond to constitute the Common Medical Record abstraction of the W3-EMRS architecture.

For this feasibility study, three site servers were implemented using databases extracted from the three hospital EMRS and scrubbed to remove identifying data. Two databases were loaded into Oracle 7 relational database management systems located at MIT, coded in the language C with additional Oracle libraries to interface the program with the database and allow electronic access to the data. One database and its site server were implemented on a Forté client/server system at MGH. The W3 server at the sites starts up the site server when a query is sent to it, and feeds the query to the site server over standard input. The site server, after processing the query and fetching the relevant data, can then simply put the HL7 message on standard output for the W3 server to pick it up and send back to where the original request came from.

Security Technologies

Medical records are generally sensitive and need to be handled in a way that does not allow them to be accessed by those not so authorized, at any time, including when being transported over W3. This and concerns over patient autonomy, and patient control over the release of information led the Boston EMR Collaborative to invest significant effort in defining a common set of policies, and procedures to secure appropriate patient, provider, and institutional authentication, and patient consent. The description here is limited to the security technologies that we have already implemented.

Currently there are two competing ways to handle secure W3 transactions. The first is an implementation from Netscape Inc. called Secure Socket Layer (SSL)[10], which is an additional network layer put on top of TCP. The second is a protocol called secure HTTP (SHTTP)[11]. SHTTP negotiates the level of security as well as the optional authentication scheme before the actual transmission of data.

Although SSL is already built into Netscape's browser while SHTTP is still in development, we believe that the latter is better suited for our purposes for various reasons. Unlike SSL, SHTTP is completely abstracted from the network layer and is therefore independent of implementation. The ability to negotiate different authentication and encryption algorithms is very useful in a heterogeneous environment of differing hospital information system with varying security levels and encryption algorithms. It also can be used to selectively limit access to different parts of the medical record, allowing access to more sensitive data only at higher levels of authentication and encryption.

DISCUSSION

The Agglutinator is now up and running on a Sun SPARCStation 10 with Solaris 2.5. All of the defined datatypes are being transferred from the site servers through the Agglutinator into HTML documents upon request. The system is technically sound in that all the connections are made and data transfer is sustained in a predictable and orderly manner. The three tier architecture clearly works in our laboratory.

One of the problems noticed was that it was very hard to get an overview of the data available at a site without retrieving the data in full. This is a function of the HL7 dialog that we used; there were no slots or definitions for sending overviews of the data within the protocol. For example, to know who wrote the notes of a patient, it is necessary to retrieve all the notes in full,

which results in a lot of data being transferred that may not be of interest to the user.
Another problem with this architecture is the statelessness of all the elements in it, a problem that appears in many forms. First of all, the HL7 protocol defines a sequencing protocol for the messages: every message has a number and the protocol states how the number of a message between the Agglutinator and a certain site server is supposed to be assigned based on the previous message. The site servers run once for each query and then terminates. Consequently, they do not know what the previous message-number of a message to their site was. That number could be recorded at several locations but as other developers of W3 systems have noted, overcoming the statelessness of these systems is awkward at best and was not attempted in the current implementation.

The Agglutinator doesn't record any of the data it receives; after passing data to the browser, it exits and releases all used memory. This means that if a user requests to view the data again, a whole new query has to be created, submitted and returned for data that has already once been retrieved. In the current implementation, we did not attempt to implement any caching schemes to overcome this performance limitation.

FUTURE DIRECTIONS

Remote Servers

Two of the three site servers constructed are physically and virtually very close to the Agglutinator. The test-system has now run smoothly for months with those servers on a separate machine from the Agglutinator, but both machines were inside the same domain. We would like to test the architecture in a more realistic situation of having all site servers outside our domain, created and maintained by our collaborating partners, accessing their data from various databases.

New Internals

The Agglutinator is now one big monolithic application - decoding requests, handling HL7 dialogues and creating HTML pages are all handled in the same process space. While this construction has served us well, new features that we are examining are forcing us to redesign the Agglutinator. This will probably result in a decomposition of the Agglutinator into more easily manageable executables, each written in a language optimized to their task. The core of this new architecture may be a database so as to keep state inside the Agglutinator and allowing for some caching of frequently queries or recently requested data.

Acknowledgments

This research was supported by the National Library of Medicine (U01 LM05877-01, LM05854, LM4-3512), the Agency for Health Care Policy and Research (HS 08749), and in part by t Sun Microsystems and Oracle Corporation.

References

1. Kohane I, Greenspun P, Fackler J, Cimino C, Szolovits P. W3-EMRS: Access to Multi-Institutional Electronic Medical Records via with World Wide Web. *Hripsak G, ed. Spring Congress of the American Medical Informatics Association. Boston, MA,* 1995.
2. Kohane IS, Greenspun P, Fackler J, Cimino C, Szolovits P. Building National Electronic Medical Record Systems via the World Wide Web. *JAMIA* 1996;3(3):191-207.
3. Willard KE, Hallgren JH, Sielaff B, Connelly DP. The deployment of a World Wide Web (W3) based medical information system.*Gardner RM, ed. Symposium on Computer Applications in Medical Care. New Orleans, Louisiana: Hanley & Belfus, Inc*, 1995:771-775.
4. Cimino JJ, Socratous SA, Grewal R. The informatics superhighway: Prototyping on the World Wide Web. *Gardner RM, ed. Symposium on Computer Applications in Medical Care. New Orleans, Louisiana: Hanley & Belfus, Inc*, 1995:111-115.
5. Kohane IS, Wingerde FJv, Cimino C, et al. Sharing Electronic Medical Records Across Multiple Heterogeneous and Competing Institutions. *Cimino J, ed. Submitted to Proceedings of the Symposium for Computer Applications in Medical Care. Washington, DC: Hanley & Belfus, Inc.*, 1996.
6. Kohane IS. Getting the Data In: Three-Year Experience with a Pediatric Electronic Medical Record System. *Ozbolt JG, ed. Proceedings, Symposium on Computer Applications in Medical Care. Washington, DC: Hanley & Belfus, Inc*, 1994:457-461.
7. Barnett GO. Computer-stored ambulatory record (COSTAR). *DHEW,* 1976:
8. Bleich H, Beckley R, Horwitz G, et al. Clinical computing in a teaching hospital. *New England Journal of Medicine* 1985;312:756-764.
9. Health Level Seven: An application protocol for electronic data exchange in healthcare environments; Version 2.2. *Chicago, Illinois: Health Level Seven*, 1990.
10. Hickman KEB, Elgamal T. The SSL Protocol. Internet Draft. *Netscape Communications Corporation,* 1995:
11. Rescorla E, Schiffman A. The Secure Hypertext Transfer Protocol. Internet Draft. *Enterprise Integration Technologies*, 1994: