WHAM!: A Forms Constructor for Medical Record Access via the World Wide Web

Alexander Hinds[1], Children's Hospital Informatics Program, Children's Hospital, Boston, MA
Philip Greenspun, Laboratory for Computer Science, MIT, Cambridge, MA
Isaac S. Kohane, MD, PhD, Children's Hospital Informatics Program, Boston, MA

WWW-EMRS is an architecture that provides a set of abstractions of electronic medical record systems (EMRS). This architecture has enabled us to implement interfaces to heterogeneous EMRS via the World-Wide Web that provide consistent visual presentations and functionality. We describe WHAM!, a program that allows rapid generation of customized interfaces to WWW-EMRS that therefore deliver the customized functions across multiple EMRS.

INTRODUCTION

The rapid growth in the number of deployed electronic medical record systems (EMRS) has not been accompanied by convergence in their implementations. They use different database management systems, different data and organizational models, different user interfaces, different vocabularies and client-server architectures. Among the consequences of this large-scale heterogeneity is the difficulty of following the course of a patient across multiple institutions or to aggregate data across these institutions. Furthermore, each additional EMRS that a provider must use requires a substantial training effort. The WWW-EMRS is an architecture that we have developed as part of a collaborative project with the National Library of Medicine in order to address these problems. WWW-EMRS provides an abstraction of the data content and services of medical record systems that is independent of the implementation of any particular EMRS. WWW-EMRS provides access to these abstracted services via an interface to the World Wide Web (WWW) which provides us with uniform access on multiple hardware platforms throughout the Internet. We have already implemented a working prototype of WWW-EMRS [1,2] using the Clinician's Workstation (CWS) [3] as the underlying EMRS. In our earlier work, implementing the visual interfaces required programmers experienced in the HyperText Markup Language (HTML) and in the structure and functions of the WWW-EMRS abstractions. If WWW-EMRS is to be adapted to the practices of the wide range of users that typically use EMRS, then a much more straightforward and efficient mechanism for authoring functions and views must be provided. Consequently, we have built an interactive graphical tool that permits users to easily construct database query forms customized for specific clinical environments and needs. This tool, WYSIWYG HTML Authoring for Medicine (WHAM!) generates the forms linking HTML forms to component services of WWW-EMRS without requiring the author to have knowledge of either HTML or any particular database technology. A WHAM! user creates the forms' layout using graphical tools, much as with a drawing program. These forms are used by general purpose WWW clients, such as Mosaic and Netscape. Thus, a physician temporarily at an outlying clinic can still access his or her patients' charts, a network-linked referring nurse practitioner can check on a patient's progress, a clinical researcher can generate ad hoc queries, and a utilization researcher can review clinical documentation at multiple hospitals. Each provider will have a customized view of the EMRS that is consistent across the multiple EMRS that document their patients' course and treatments.

In this paper we first describe the background to the development of WHAM! including a brief description of WWW-EMRS as well as work done in providing users with graphically intuitive data-base query tools. We then review the design considerations that drove the development of WHAM! to its present implementation. To further motivate the design choices, we provide an example of designing and using a WHAM! form to query the currently implemented version of WWW-EMRS.

BACKGROUND

WWW-EMRS

The WWW-EMRS architecture has been designed to provide sufficient abstractions of the functions and presentations of most EMRS so that any WWW browser can deliver consistent views and user functionality independent of the particular implementation of the EMRS. The two principal abstractions are the Common Medical Record (CMR) and the Visual Presentation Server as illustrated in Figure 1. The CMR specifies information services over of a set of data elements that are common to multiple EMRS. The visual presentation server organizes the data retrieved from the CMR into archetypal layouts (e.g. spreadsheet, annotated picture, list) that is independent of the language that the presentation will be finally implemented in. The Visual Presentation Server also associates permitted user interactions with each presentation element. Each layer of abstraction in the WWW-EMRS requires an active process capable of mediating the messages between each layer (e.g. converting SQL queries of a local EMRS into CMR query messages). One of the major efforts in the development of WWW-EMRS is to achieve sufficient breadth in both the CMR and Visual Presentation Servers so that they satisfy the needs of several classes of users and work with the mechanisms of different local EMRS. After the success of our initial WWW-EMRS implementation with the CWS, we have been working with other developers of EMRS to ensure that the WWW-EMRS, and particularly its CMR component, can serve these other, divergent EMRS.

[1]Visiting student from Section on Medical Informatics, Stanford Univ., CA. Correspondence should be addressed to Isaac Kohane.

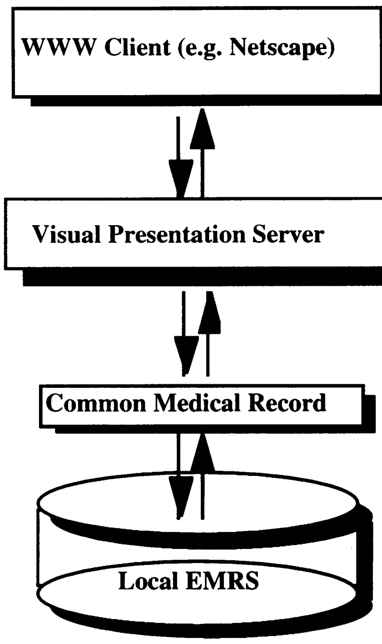


Figure 1: Architecture of W3-EMRS

WHAM! bypasses the Visual Presentation Server by directly describing visual elements in the HTML syntax and by associating CMR queries directly with the HTML forms. This enables a user to develop a WHAM! form locally for use with their WWW client program without having to modify or attach new document types to the Visual Presentation Server (which may be located at a remote site along with the EMRS being accessed).

User-Interface Authoring Tools for EMRS

Several commercial tools exist that enable developers and end-users to rapidly develop graphical user-interfaces to a large class of data-base product. For example, Powerbuilder™ (Powersoft Corp.), and Foxbase Pro (Microsoft Corp.) all provide fairly comprehensive tools to generate customized screens for data entry and retrieval. These products depend on having access to the data dictionary for the database for which they are building tools. Also, to maintain their generality, these tools do not have any knowledge of particular domains such as electronic medical record systems. In contrast, knowledge-based systems such as PROTÉGÉ-II [4] include mechanisms to acquire different kinds of task and domain-specific knowledge (e.g. knowledge about clinical protocols). The PROTÉGÉ-II system can then be used for the development of domain-specific tools such as user interfaces for data entry or knowledge acquisition. Also, the PEN&PAD system [5] uses a clinical information model to drive the generation of data input forms.

DESIGN CONSIDERATIONS

The design considerations for WHAM! fell into several areas. First, we wanted the tool to run on the roughly the same range of machines on which the WWW browsers were available and to present an intuitive interface.

Second, we wanted to build on existing networking technology and client/server solutions as much as possible. A traditional way to accomplish our goals would be to insist that all potential users of the system adopt custom-built network protocols and interfaces. However, we specifically wanted the (output of the) tool to be usable with general purpose clients, (e.g.. Netscape and Mosaic). Extending the functionality of the standard WWW browsers might unnecessarily compromise the acceptance and accessibility of the WWW-EMRS project.

Third, we wanted to provide a set of clinical display and interaction object libraries. Each object would correspond to one of several commonly required clinical data-types and editing tasks, and would in turn be used to generate parts of HTML documents which embody task or clinic-specific viewer/editors. Users would assemble various objects to form compound documents whose output would be the final HTML specification of the evolving form.

To accomplish all the above goals, we chose to write all code in a dialect of Smalltalk called SmallTalkAgents (by Quasar Knowledge Systems) for the Macintosh. STA was chosen as the development environment because it supports rapid prototyping, allows the creation of standalone double-clickable Macintosh applications, and supports Platform Independent Portable Objects (PIPOs). PIPOs allow WHAM! to run with minimal or no modification or even recompilation on a variety of platforms, including SunOS, Windows, and OS/2.

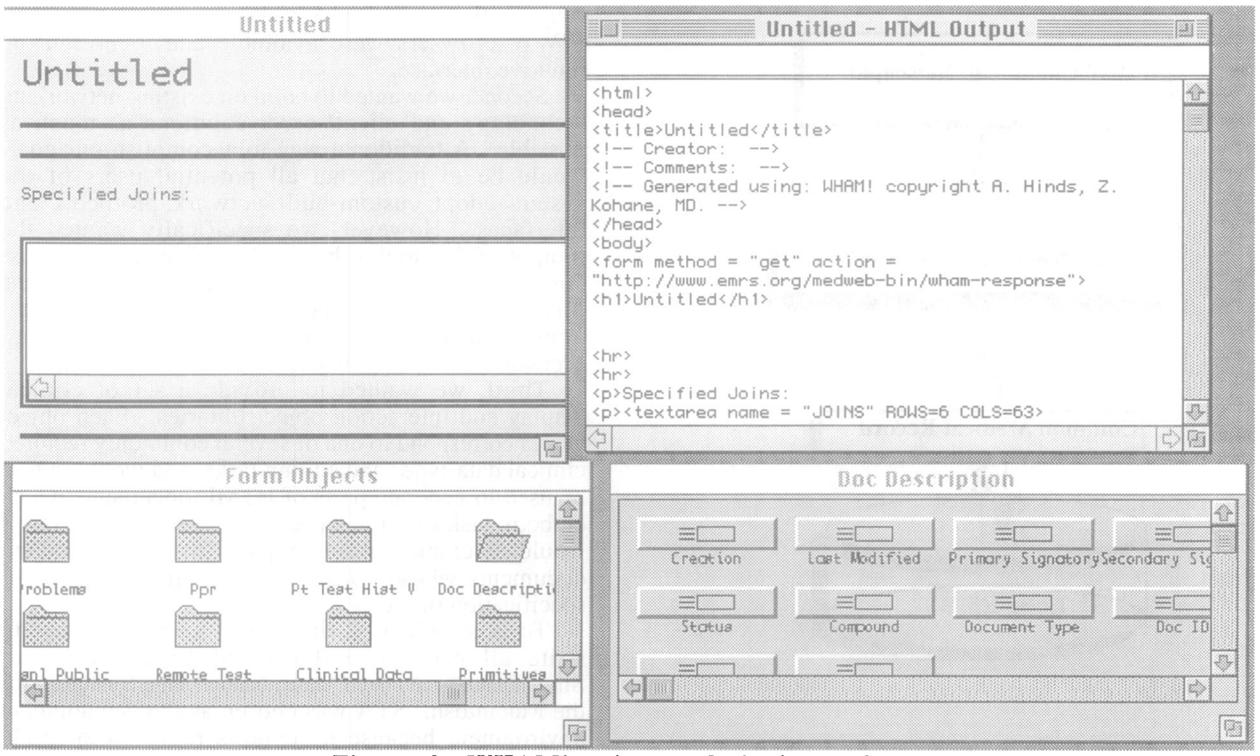
SYSTEM DESCRIPTION

Architecture Overview

All WHAM! forms are *compound documents*: documents consisting of one or more *components*. These components are graphical representations of HTML primitives, such as horizontal bars or lines of text, and more elaborate components that reflect data in the Common Medical Record (CMR) and "know" how to perform CMR transactions.

Our first step towards developing a compound document solution was the definition of a small subset of clinical document components_that are common to the viewing and editing tasks. Note that this effort is quite distinct from the specification of the format and structure of an EMRS data-base. For example, for each pathology report, a data-base may contain various tables, the text of the report, the signatories, documentation regarding processing of the tissue sample, an image of the sample, the site obtained and the name of the surgeon who obtained the specimen. Only a subset of this information is required by most clinical users and therefore the corresponding clinical document components should reflect these requirements.

The clinical document components were identified by reviewing the existing WWW-EMRS interface and identifying the recurrent presentation formats and archetypal data-relations.



```
<html>
<head>
<title>Untitled</title>
<!-- Creator:  -->
<!-- Comments:  -->
<!-- Generated using: WHAM! copyright A. Hinds, Z.
Kohane, MD. -->
</head>
<body>
<form method = "get" action =
"http://www.emrs.org/medweb-bin/wham-response">
<h1>Untitled</h1>


<hr>
<hr>
<p>Specified Joins:
<p><textarea name = "JOINS" ROWS=6 COLS=63>
```

Figure 2: WHAM! prior to designing a form.

Our compound document architecture allows special-purpose code resources to be embedded in the document . These embedded code resources consist of small fragments of transaction specifications. That is, for each visual element in a clinical document component, we are able to specify the information required from the user or from the data-base required to display that element.

After developing a library of such clinical document components, we were able to design a variety of EMRS viewer/editors. Designers of WWW-EMRS user-interfaces (currently, only the authors) pick and choose among the available clinical document components those that best suit a particular class of clinical tasks. These are then assembled into one or more compound documents. The WWW client then makes the best presentation of the HTML document possible on the user's hardware.

We intentionally did not develop a complex compound document Applications Programmer Interface (API) as from the outset we anticipated evaluating and then choosing one of the candidate "intelligent" compound document architectures such as OLE (Microsoft Corporation) or OpenDoc [6], a cross-platform standard for compound documents and components.[2]

Components

A user works with components, which comprise the pieces of a document. These components include both document data and the component-editor code that manipulates it. In WHAM!, all the components a user interacts with are derived from the component class, *WHAMComponent*. Once instantiated, most of the objects users manipulate are graphical representations of objects within the CMR as well as the HTML graphical equivalents those types represent. WHAMComponents "know" how to create themselves from HTML, present themselves graphically, generate the HTML which they represent, and for those components where it's relevant, provide enough information for the CMR server to construct a query.

Storage

WHAM! uses ASCII text files in HTML format to save and restore documents. Each saved file can be thought of as a collection of *storage units*. A storage unit is an ordering of *properties* a single component has used to save its state. A property, in turn, is an ordering of *tag*, value pairs. Storage units and properties are not first-class objects in the sense that they have no official representation in the file. However, they are inferred by components when a given component attempts to restore its state given an HTML stream.

To allow the CMR server to return a form with its empty fields filled in, each file contains one hidden field that contains a copy of the HTML used to make up the form itself. The WWW standard for a sending a filled-out form to the server sends only field names and their contents, not the HTML comprising the form, to the server. The CMR-WWW interface gets values from the database for all the blank fields, then needs a copy of the original HTML that specified the form. The hidden field (<input name = "HTML" type = hidden value = "...") supplies this HTML and the CMR-WWW interface substitutes all the appropriate field values then returns the new HTML to the client.

[2]The OpenDoc specification is actually a superset of OLE, so henceforth OpenDoc is understood to include OLE.

Interface Description

When WHAM! starts, it opens three windows (Figure 2 includes these three and a fourth window described below). The **Form Objects** (lower left) window presents the user with an interface similar to that of the Macintosh Finder: all components the user will manipulate are contained within its hierarchical folders. Users select components by navigating the folders in a manner similar to the operation of the Finder. When he finds a component of interest, he can drag it onto the current document where it will be inserted where the mouse was released.

Interaction metaphors include *direct manipulation*. By this we mean that most elements on the form can be edited by clicking on them. For example, text can be edited by simply selecting it and typing replacement text; input lines can have their default text set by selecting them and typing in the text; and joins can be added to the evolving form by simply clicking in the join "pane" on the form (which subsequently opens the Join Construction window).

Most of the folders in the Form Objects window (with the exception of the 'Primitives' folder) correspond to data objects defined in the CMR; folder components correspond to attributes of one CMR data class.

The **Untitled** window (upper left, figure 2) is the document on which the user will operate. As shown in the figure, no data elements from the CMR have been yet been selected. The **Untitled - HTML** window (upper right, figure 2) allows the user to view the HTML output as it's being generated; that is, its contents are updated in response to most user actions that lead to changes in the corresponding **Untitled** window. WHAM! does allow the user to make a limited range of changes to the actual HTML which can be incorporated into the graphical layouts. However, WHAM! expects certain formatting conventions in the files it generates, and deviations from these conventions may result in WHAM! being unable to parse the resulting HTML for further modification The fourth window (lower right) illustrates the component attributes that appear when a folder in the Form Objects window is opened. In this instance it is the components of the document description folder.

The **Join Construction Window** (not shown) is where a user is able to specify the relations between tables.

WHAM! does not have a model of what constitutes a meaningful query in the CMR. It is currently possible for a user to create a form that specifies syntactically valid queries which are semantically unsound. This error wouldn't be apparent until run-time.

WHAM! Query Processing.

Once a WHAM!-generated form is loaded into a WWW browser, any of the fields in the form can be filled out. When the users clicks on the "Submit Query" button, the query function, along with the values entered into each field in the form are sent to the CMR. As described above, each form has an associated hidden field which describes the entire form and this too is sent to the CMR. Upon receipt of the WHAM query, an corresponding query is constructed in the data manipulation language that is native to the underlying EMRS. The query is composed by using the filled out fields in the WHAM! form as constraints on the query and the empty fields as the data elements to be retrieved. For the CWS EMRS, which is Oracle-based, this corresponds to the WHERE clause and SELECT clause of an SQL query.

When the EMRS returns the data elements requested in the query to the CMR, these are entered into the appropriate fields in the copy of the WHAM! form that was sent in the hidden field of the original WHAM! form. Also, the values of the field that were filled out by the user are replaced in their respective positions in the copy of the form. The filled-out form is then sent from the CMR to the WWW browser via the Internet. The user of the WHAM! form can then modify the contents of the returned form and resubmit it to further explore the EMRS data.

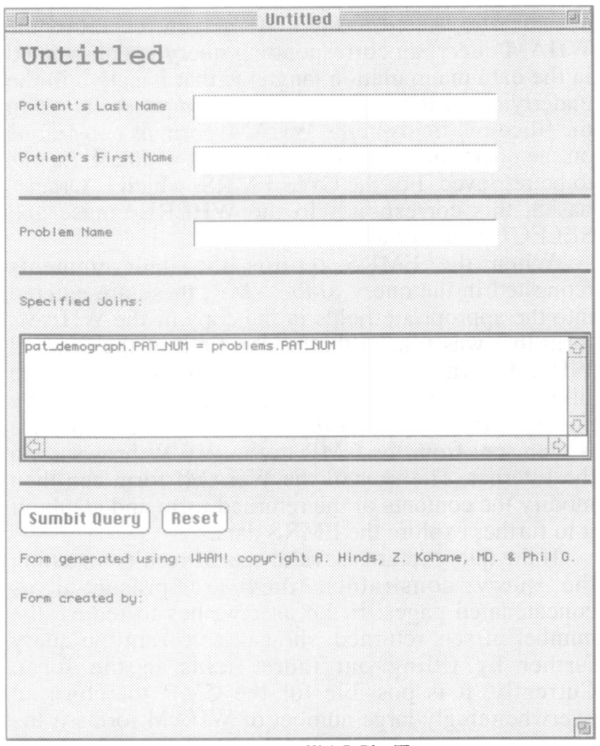
If the query returns several sets of results matching the query constraints, they are presented as concatenated pages. If the user wishes to reduce the number of sets returned, she can constrain the query further by filling out more fields in the form. Currently, it is possible for the CMR to return an overwhelmingly large number of WHAM forms if the query is grossly underconstrained. Limiting this phenomenon is one of the priorities of future work on WHAM!

RESULTS—EXAMPLE OF USE

It is often useful to know a patient's problem list, as summarized by clinicians who have cared for the patient in the past. In this example, we have used WHAM! to construct a form that includes the patient's last and first name as well as the name of any problem annotation. Although the CMR contains much more richness in its description of patients and problem annotations, this form has been limited for purposes of illustration.

To generate the form, the patient names fields were "dragged" from the demographics folder to the WHAM! form and the problem name was dragged from the problems folder. The Join Construction Window was used to tie the problem to the patient name via the patient's medical record number. Finally the Submit Query and Reset buttons were dragged from the Primitives folder. The resulting form is illustrated in Figure 3. Subsequently, the form was titled "Problem List" and saved.

Figure 4 shows the WHAM! form as it appears when displayed by a WWW browser, after the user had submitted it, having filled out the first and last name of the patient. This example form is significantly less compact and visually appealing than the pre-existing hand-crafted forms designed for WWW-EMRS. However, we do believe that the current tradeoff between ease of use and design flexibility can be readily shifted.



Untitled

Patient's Last Name

Patient's First Name

Problem Name

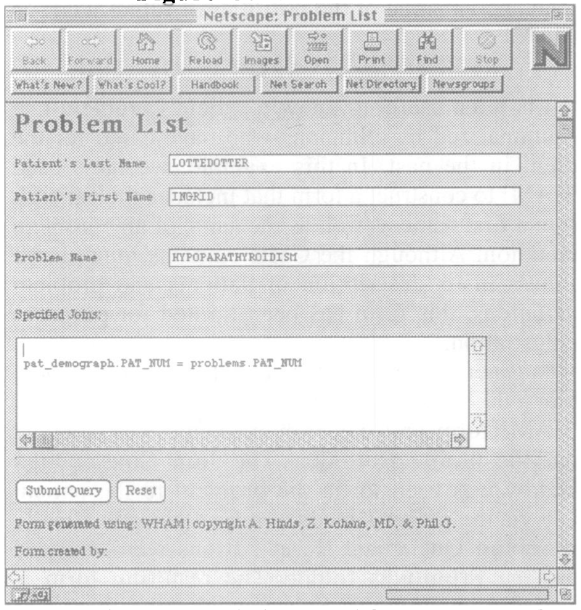
Specified Joins:

```
pat_demograph.PAT_NUM = problems.PAT_NUM
```

Form generated using: WHAM! copyright A. Hinds, Z. Kohane, MD. & Phil G.

Form created by:

Figure 3: WHAM! Form



Problem List

Patient's Last Name	LOTTEDOTTER
Patient's First Name	INGRID
Problem Name	HYPOPARATHYROIDISM

Specified Joins:

```
pat_demograph.PAT_NUM = problems.PAT_NUM
```

Form generated using: WHAM! copyright A. Hinds, Z. Kohane, MD. & Phil G.

Form created by:

Figure 4: WHAM! form with query result

CONCLUSION

The WWW-EMRS architecture provides a powerful, extensible, multiplatform architecture for accessing data from multiple heterogeneous EMRS. However, it does not provide of itself the means for easily customizing the display of data and modifying the functionality of the interface. WHAM! makes a substantial first step towards this goal. Users of WHAM! can design functional forms without any knowledge of the underlying EMRS or the architecture of WWW-EMRS. This enables clinicians and other users to generate ad hoc queries against multiple EMRS to which they have authorized access without necessarily requiring major technical support. It also allows information services groups to provide rapid prototype functionality to diverse groups of end users in addition to using the pre-assembled forms that come with the WWW-EMRS.

Our initial experience with WHAM! suggests several tasks to improve its performance and utility. These include: 1) Improving the management of underspecified queries. We will be investigating knowledge-based mechanisms to determine how much data should be returned for a given query before prompting the user to reconsider or respecify the query. 2) Adding the same range of visual presentation types (e.g. flowsheets, lists) to WHAM! as is currently available in the hand-crafted WWW-EMRS forms. 3) Adding support for cryptographic/security services for individual components of WHAM! forms. 4) Supporting additional transaction types with the CMR. 5) Investigating the adoption of document component architectures such as OpenDoc for the WHAM! component architecture to permit interoperability of components from different developers.

References

[1] Kohane, I. S.; Greenspun, P.; Fackler, J.; Szolovits, P. Accessing Pediatric Electronic Medical Record Systems via the World Wide Web. Pediatric *Research* **1995**, *37*, 139A.

[2] Kohane, I.; Greenspun, P.; Fackler, J.; Cimino, C.; Szolovits, P. W3-EMRS: Access to Multi-Institutional Electronic Medical Records via with World Wide Web To appear in: *Spring Congress of the American Medical Informatics Association.* Boston, MA: 1995.

[3] Kohane, I. S. Getting the Data In: Three-Year Experience with a Pediatric Electronic Medical Record System In: *Proceedings, Symposium on Computer Applications in Medical Care.* J. G. Ozbolt. Washington, DC: Hanley & Belfus, Inc., 1994:457-461.

[4] Musen, M. A.; Gennari, J. H.; Eriksson, H.; Tu, S. W.; Puerta, A. R. PROTÉGÉ-II: Computer Support for Development of Intelligent Systems In: *Eighth Wolrd Congress on Medical Informatics.* Vancouver, Canada: (in press), 1995:

[5] Healthfield, H. A.; Hardiker, N. R.; Kirby, J. Using the PEN&PAD Information Model to Support Hospital-Based Clinical Care In: *Proceedings Symposium on Computer Applications in Medical Care.* J. G. Ozbolt. Washington, DC: Hanley & Belfus, Inc., 1994:452-456.

[6] "OpenDoc Programmer's Guide," Apple Computer Inc, Developer Press, (c), Seed Draft 11/16/94 1994.

Acknowledgements

This research was supported by the National Library of Medicine (U01 LM05877-01), and in part by the Oracle Corporation and the Charles Hood Foundation.