Enabling Clinicians, Researchers, and Educators to Build Custom Web-Based Biomedical Information Systems

Rex Jakobovits, PhD[2] James F. Brinkley, MD, PhD[1]
Cornelius Rosse, MD, DSC[2,1] Ed Weinberger, MD[3]

[1]Structural Informatics Group Dept. of Biological Structure University of Washington, Seattle, WA

[2]Workhost Data Solutions Seattle, WA

[3]Department of Radiology Children's Hospital and Regional Medical Center, Seattle WA

We describe an open-source toolkit that enables clinicians, researchers, and educators to build their own web-based biomedical information systems. The Web Interfacing Repository Manager (Wirm) is a high-level application server aimed at medical professionals, allowing them to create individually tailored systems for managing their multimedia data and knowledge. We provide an overview of the features of Wirm, explaining how they meet the requirements for supporting biomedical information management, and describe four applications that are currently being developed with Wirm: MyPACS, a teaching file authoring system for radiologists, Fathom, an experiment management system for natural language processing, the Digital Anatomist Repository, an image archiving tool for medical schools, and Ontolog, a browser for medical vocabularies.

INTRODUCTION

With the advent of new technologies for generating medical research data, there is a growing need for tools that enable researchers, clinicians, and educators to manage multimedia information. The web provides a framework for exporting interfaces to share data, but there is a lack of high-level web application development tools suitable for use by scientists and health professionals who are not skilled programmers. The requirements for building a custom information management system typically extend far beyond the capabilities of the resources available to the clinician or researcher. Biomedical information systems often require individually-tailored interfaces for different classes of end-user, multiple levels of authentication and security, visual tools for modeling complex data types, customizable forms for acquisition and editing of data, efficient mechanisms for interfacing with databases and other software applications, and support for a wide range formats and protocols [1]. To build and maintain such a system would typically require a large investment in software infrastructure and hundreds of hours of skilled programming from specialized web application developers.

Readily available tools such as Microsoft Front Page allow non-programmers to create web sites for disseminating their knowledge and data as hyperlinked documents. However, sites created by such tools are limited to the dissemination of static data, and are unable to serve dynamic content or interact with databases and external programs. Building full-featured web applications requires the use of application server technology, such as Cold Fusion or IBM Web Sphere [2]. Unfortunately, application servers are complex to use, and require the services of specialized programmers. The same holds true for CORBA-based application development platforms, such as OpenEMed [3], which are used to build large-scale clinical information systems by assembling networks of distributed objects. While these systems hold promise for enterprise-level application development, they are unwieldy and overly complex for supporting small projects that may lack access to CORBA infrastructure. Consequently, the task of building full-featured web information systems remains prohibitively expensive for the majority of small labs and individual researchers, clinicians, and educators.

We have developed an open source software toolkit, the Web Interfacing Repository Manager *(http://wirm.org)*, which is enabling medical professionals to build custom web information systems without having to hire web programmers [4]. Wirm provides a browser-based interface that allows scientists and clinicians to describe the structure of their domain knowledge. Using this information,

Wirm automatically generates a web application that enables end-users to import, organize, query, and visualize domain data. By following a simple methodology, the system developer may then customize the application for different classes of end user with a minimal amount of programming using Wirm's high-level application programmer interface.

Wirm includes facilities for supporting arbitrarily complex data types and their associated metadata, as required by biomedical research projects. In addition, Wirm provides tools for handling images, managing user sessions, regulating access control at a fine granularity, and creating context-sensitive interfaces that adapt themselves to different classes of end user. This is especially significant for managing medical research data, which requires multiple privacy contexts. Wirm makes it easy to support multiple interfaces at the granularity of a data object view, rather than requiring the designer to create a separate site for each user class. In this way, Wirm reduces the task of building an information system into manageable steps.

Wirm derives its power by leveraging free, open-source software, including Image Magick for manipulating images [5], the MySQL database for storing records [6], the DBI module for database interfacing [7], and the CGI module for generating web-based user interfaces [8]. While these free software packages are readily available, they often go untapped by the domain experts who need them most, as they are aimed at Linux developers who have the expertise to understand their details and tie them together with custom code. Wirm integrates these different components into a coherent framework, providing visual interfaces and high-level abstractions suitable for use by scientists, clinicians, and educators.

Wirm was originally motivated by the Human Brain Project, as a means for managing experimental data in neuroscience. It is still being actively used to manage language data from the University of Washington Brain Project [1]. Wirm is now being applied to a number of new informatics projects in hospitals, research labs, and universities. As a demonstration of the breadth of Wirm's utility, we describe four applications that are underway: *MyPACS,* a teaching file authoring system for radiologists, *Fathom,* an experiment management system for natural language processing, the *Digital Anatomist Repository,* an image archiving tool for medical schools, and *Ontolog,* a browser for medical vocabularies.

MyPACS

Wirm was used to build MyPACS *(http://mypacs.net)*, a web-based service that allows radiologists to manage their own online medical image repositories. Radiologists often keep files of interesting cases for sharing with residents or colleagues, or for use in slides or publications. Teaching files have been traditionally stored as hard copies in filing cabinets, but over the past several years a growing number of institutions are publishing their collections on the web [9]. This has the obvious advantages of widespread dissemination and retrieval by search engines. However, the task of implementing a full-featured online teaching file repository requires significant development effort and web programming expertise. Each institution creates their repository using ad-hoc methods and tools, and must maintain their own web server, databases, and application software. With MyPACS, the institution is relieved of these burdens, and radiologists can immediately begin authoring cases through their web browser, while maintaining complete control over the content and accessibility of their own collection.

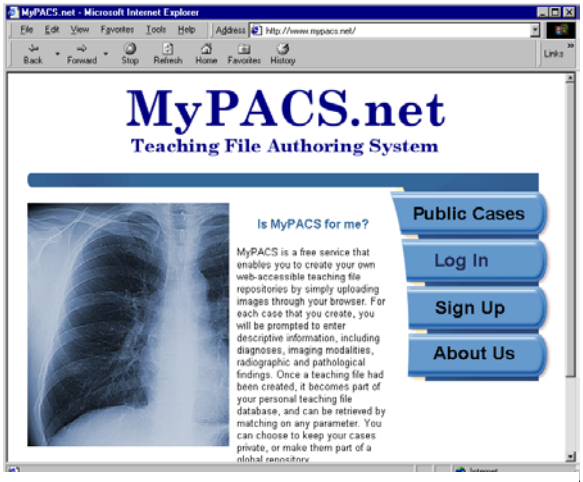

Figure 1: MyPACS Home Page

MyPACS allows the user to upload medical images, enter descriptive information about the case, and input structured data such as patient demographics, image modalities, anatomical structures, and pathological findings. The interface includes a structured reporting tool that guides the user in specifying standardized vocabulary terms. Authors may customize the appearance of their teaching files, including the layout and resolution of images, attributes to be displayed, and other presentation parameters. For each teaching file, the author may choose from two levels of access regulation: public, and private. Public files can be accessed by any user (patient identifiers are always withheld), whereas

private files are accessible by the author alone. These features were easy to implement using Wirm's user management facilities and high level application programmer's interface, which facilitates building context-sensitive views over repository data.

Teaching files may be retrieved by searching on any parameter, including date, title, pathology, anatomy, or full text searching over the findings in the case. Wirm's form interface reads the user's search criteria and translates them into SQL queries, which are handled by Wirm's database interface.

Images can be uploaded in any of over 60 recognizable formats (such as JPEG, GIF, PNG, TIFF, etc). For viewing over the web, MyPACS converts images into the browser-friendly JPEG format, but an image can always be retrieved in its original format. Users may specify a preferred image resolution for browsing files, and the images are converted on the fly whenever necessary. This enables images to be tailored to a user's screen size, and reduces transmission time. The system uses a smooth transform algorithm to preserve image quality during enlargement or reduction. By leveraging the powerful Image Magick programming interface, the entire image manipulation features of MyPACS required less than 100 lines of Perl code.

In addition to a teaching file distribution and publishing tool, MyPACS can be used as a convenient repository for storing and organizing personal image collections. Authors can upload interesting cases as they are encountered, and later download them for inclusion in publications or PowerPoint presentations. The author of a teaching file retains ownership of the images and case studies that are entered into the system, and has complete control over how they are used. Some radiologists are using also MyPACS for telediagnostic purposes. Without access to their PACS from home, radiologists can upload images from the hospital and call for remote referrals.

FATHOM

The UCLA Telemedicine group is performing ongoing research on natural language processing (NLP) over free-text radiology reports, to automatically generate structured records about the findings described in those reports [10]. Structured reports are clinically useful for decision support and outcomes research. The NLP system uses various statistical and machine learning methods to process the records and identify the properties, locations and diagnostic interpretations of each finding mentioned in the text. Wirm was employed to build Fathom, an experiment management system to support the researchers as they tested the effectiveness of various NLP algorithms. Fathom consists of two primary components: the Training Interface for creating hand-tagged training examples, and the Patient Record Manager for navigating clinical reports.

In order to evaluate the recall and precision of the coreference resolution processor, the results must be compared to thousands of hand-coded candidates. The task of coding these results is tedious and

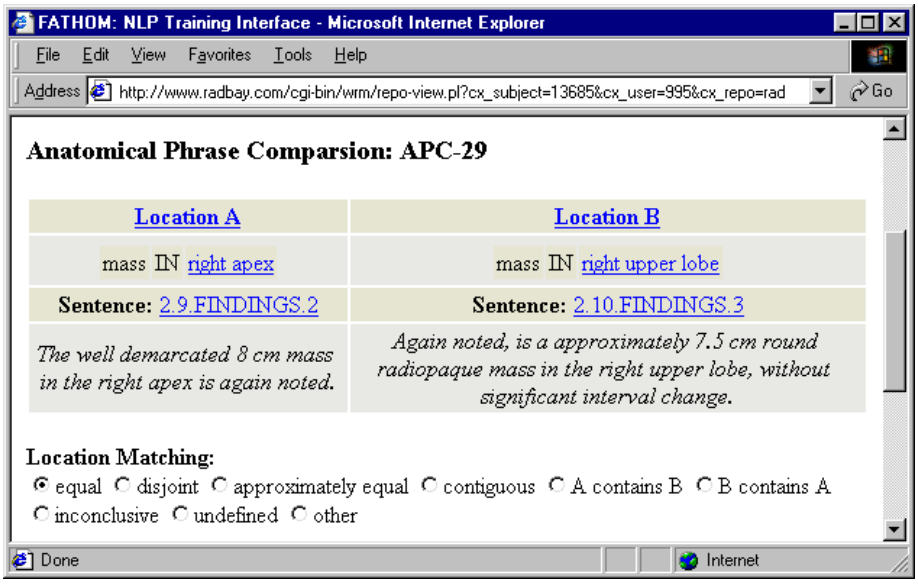

Figure 2: Fathom Training Interface

requires judgments based on expert knowledge. Wirm allowed us to quickly build a Training Interface for presenting the candidate comparisons to domain experts, providing convenient menus and buttons for indicating coreference decisions, and recording their choices in a database for later comparison with the automated decisions. For example, Figure 2 shows the Anatomical Phrase Comparison form of the Training Interface, in which the system is trying to determine whether two sentences mentioning a "mass" refer to the same or different findings. The medical expert is shown the anatomic context of each finding, and asked to identify the relationship between those locations. This information can be stored in the NLP system's statistical repository, for use in making future decisions about resolving references to masses.

The Patient Record Manager imports records from the hospital information system, providing navigation and search interfaces over the full text of the records. Users are able to organize records by patient, by experiment corpus, or by physician. In addition to the raw records, the system holds intermediate data structures used by the various stages of processing. For example, the syntactic parser splits the records into sentences, which are each assigned a sequential identifier for reference by the semantic interpreter. The various processing stages express their output as XML records, which are read by the Patient Record Manager and made available for structured browsing. Perl's built-in text parsing abilities greatly facilitated the process of importing these records.

DIGITAL ANATOMIST REPOSITORY

The Structural Informatics Group at the University of Washington required a way to organize and disseminate their large image collection, which includes thousands of medical illustrations, radiological images, and computer-generated graphical models from publications, multimedia atlases, and other sources. Wirm was selected as the tool of choice to build a web-based archiving system, depicted in Figure 3. The educators designed a list of the attributes that they wanted to classify their images by, including anatomic location, format, and intended uses. Images are uploaded into the repository and automatically converted into formats suitable for viewing through a web browser. Users may retrieve images through a search form, constraining on any attribute. Implementing a full-featured prototype required only 17 hours of effort with Wirm.

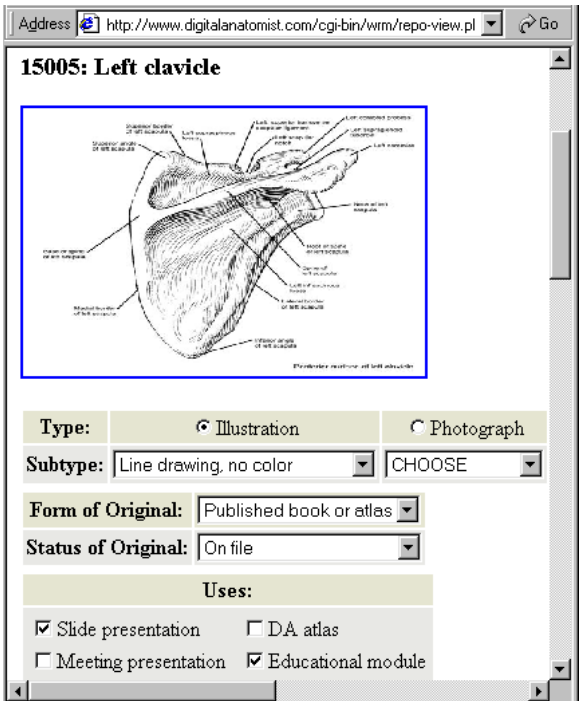

Figure 3: Digital Anatomist Repository

Wirm is being used to continually evolve and improve the metadata schemas and user interface. New features will include the ability to create slide shows, label regions within an image, and integrate medical vocabularies for indexing. The interface will be deployed as an image archiving system for the entire medical school.

ONTOLOG

Medical terminologies such as SNOMED and the UMLS consist of tens of thousands of terms organized into conceptual hierarchies. While these knowledge resources provide a powerful framework for indexing, integrating, and organizing medical data, there is a lack of manageable interfaces for navigating and visualizing the relationships between the concepts. For example, a part of the Foundational Model of Anatomy [11] (the UWDA component of the UMLS) can be accessed with a java applet called the Foundational Model Builder [12], but that interface is limited to showing a single ontology at a time. Similarly, SNOMED is distributed with a simple text browser, but it is difficult to explore the vast terminology without a more flexible navigation system. This was our motivation for implementing Ontolog, a frame-based browser for medical terminologies.

The first step consisted of importing the files from their respective data sources into a Wirm repository.

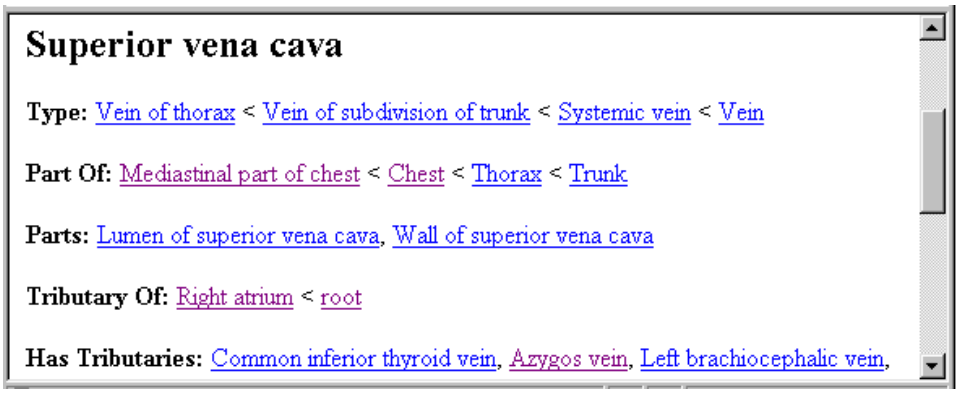

Figure 4: Ontolog Frame Browser

Using Wirm's schema tool, we specified a data model for containing the terms and relationships. Then we used Wirm's repository object API to build a simple parsing script that connected to the UWDA and SNOMED data sources and import the vocabularies into Wirm's repository. Once the structures were defined, Wirm automatically generated an interface for navigating the frames.

Ontolog allows the user to view a concept in its full context, not limited to a single attribute dimension. For example, Figure 4 depicts the frame for Superior vena cava, which shows the concept within all its relative hierarchies. To move to a new frame, the user simply clicks on the desired link.

CONCLUSION

From our experience as application developers for research labs, university departments, and hospitals, the domain expert has the best vantage point for understanding the requirements of their own data interfaces. By reducing the programming effort required to build custom biomedical information systems, Wirm gives domain specialists far greater control over the design of their schemas and interfaces. Rather than having to translate these requirements to a web programmer who is unfamiliar with the details of their work, Wirm will allow the experts to do it themselves. As the expert's understanding of their own interface requirements evolves, the desired changes can be quickly implemented.

While the current version of Wirm enables non-programmers to model their data and administer their own information systems, some programming is still necessary to fully customize application behavior. We are closing this gap by extending the capabilities of WIRM's visual interfaces to control application logic, context-sensitivity, and workflow management. As we continue to improve the Wirm application server, the task of creating web information systems will become less reliant on programming and more controlled by visual interfaces.

Acknowledgements

This publication was made possible by NIMH grant R43-MH61277-01, Human Brain Project grant DC/MH02310, and the University of Washington Initiatives Fund.

References

1. Jakobovits R, Soderland S, Taira R, Brinkley J. Requirements of a Web-Based Experiment Management System. 2000 AMIA Symposium, pp.374-378.
2. Benfield S. The Application Server Marketplace. Web Techniques, February 1999.
3. http://telemed.lanl.gov/OpenEMed
4. Jakobovits R, Brinkley JF. Managing medical research data with a web-interfacing repository manager. AMIA Fall Symposium, 454-458, 1997.
5. http://www.simplesystems.org/ImageMagick
6. http://mysql.com
7. http://dbi.symbolstone.org
8. http://stein.cshl.org/WWW/software/CGI
9. http://info.med.yale.edu/diagrad/teach_link.html
10. Taira R, Soderland S, Jakobovits R. Automatic Structuring of Radiology Free Text Reports. RadioGraphics, January 2001, 21:1.
11. Rosse C, Shapiro LG Brinkley JF. The Digital Anatomist Foundational Model: principles for defining and structuring its concept domain. Proc AMIA Symp 1998;820-4.
12. http://tela.biostr.washington.edu/fmb.html