A Web-Based Repository Manager for Brain Mapping Data

R.M. Jakobovits, B. Modayur, and J.F. Brinkley
Departments of Computer Science and Biological Structure
University of Washington, Seattle, WA

The Web provides a rapid prototyping environment for building platform-independent graphical user interfaces. A Web-based console can be implemented as a suite of CGI scripts that generate HTML code, manipulate files, execute system commands, and invoke external tools. Often these tools share data by reading and writing flat files, which must be explicitly maintained by the CGI programmer. In a repository system, metadata about each file object are maintained in a database, and access to all data is regulated by a layer of control services. This paper describes the design and implementation of a Web-based Repository Manager (WRM), which provides an application programmer's interface for controlling applications, generating HTML documents, handling Web forms, and managing multi-media data. The WRM is being used to develop a console for the Brain Mapping Framework, a system for visualizing cortical stimulation data obtained during neurosurgery.

INTRODUCTION

Due to its accessibility and ease of use, the World Wide Web is proving to be an essential medium for the global dissemination of medical research data. In addition, the Web can be used as an *intranet* for sharing documents and images between members of a research group. But the Web can be more than just a convenient platform for displaying documents. Many software vendors are offering SQL gateways which connect relational databases directly to the Web, allowing them to be queried and populated from any Web browser. Furthermore, the Web provides a rapid prototyping environment for building platform-independent graphical user interfaces. With a small amount of CGI programming, the Web can become a graphical console for launching applications that would otherwise require users to enter cryptic system commands and file names by hand.

For example, consider a neurologist studying a series of magnetic resonance (MR) images. Using a Web console, he or she might open a window into a database of patient information, click on a patient's name, and select viewing parameters from a menu. The Web server would locate the named patient's sequence of MR images, invoke an image processing program with the selected parameters, convert the resulting volume into a format recognized by the Web browser, and display the final product on the neurologist's screen. Without a Web console, the same procedure would require the neurologist to start up a database monitor, enter an SQL query to find the desired patient information, search for the MR images in the file system, figure out how to execute the image processor with the appropriate arguments, and then invoke an image viewer on the resulting file.

A Web-based console can be implemented as a suite of CGI scripts that generate HTML code, manipulate files, execute system commands, and invoke external tools. Often these tools share data by reading and writing flat files, which must be explicitly maintained by the CGI programmer. As the Web-based console grows larger in scope, the collection of data and image files tends to become unwieldy. Files become inconsistent with the scripts that manipulate them, or are misplaced entirely. The flow of data between tools becomes hard to manage, especially when the data must be translated between various formats.

One approach is to delegate as much data as possible into a relational database, and to interface the tools directly with the database. This relieves some of the data management workload from the CGI programmer, and provides traditional database features such as transactions, logging, and querying. However, a relational database is not suitable for managing images or other large file objects. Furthermore, it is often impractical to integrate existing tools with the database.

What is really needed is a unified framework for integrating the database, file collection, and applications. Such systems, known as *repositories*, are commercially available to support Computer-Aided Software Engineering (CASE) tools and Computer Aided Design (CAD) tools. In a repository system, metadata about each file object are maintained in a database, and access to all data is regulated by a layer of control services called a *repository manager*.[1]. This paper describes the design and implementation of a Web-based Repository Manager (WRM), which provides a convenient application programmer's interface (API) for controlling applications, constructing Web forms and documents, and managing multi-media data.

MOTIVATION

The WRM design was motivated by the requirements of the Brain Mapping Framework[2] being developed as part of the Human Brain Project. The system is designed to organize and visualize cortical stimulation data obtained during neurosurgery. An interactive Brain Mapping tool allows a neurosurgeon to visually match an intraoperative photo of a patient's brain surface to a 3-D MRI-based visualization. The mapping process is carried out in five stages: (i) data acquisition, (ii) alignment and cropping, (iii) cortical segmentation, (iv) visualization, and finally (v) language site mapping. The data acquisition stage involves downloading patient information and MR images from a remote server. The alignment and mapping stages involve processing the images with an in-house graphics program, while the rest of the stages are implemented in a commercial visualization package.

The heterogeneous nature of the software environment made it difficult to monitor the stages of the mapping process, and we realized that the system could benefit from a central console from which to control the data flow. We chose to implement the console as a CGI program, because of the Web's inherent rapid-prototyping and portability features.

As large volumes of data were acquired, it quickly became apparent that a database was needed to keep track of the locations and contents of files. We began utilizing a relational database to store patient information and manage metadata about the image files. We designed CGI scripts to integrate each of the software tools with the database, and to make the data accessible from the Web console. As the collection of CGI scripts grew, the entire system began to look like a repository manager. We decided to generalize the facilities for data management, tool integration, and Web interfacing so that they could be reusable with other projects. The result was the WRM architecture, described below.

DESIGN

The Web-based Repository Manager architecture consists of a relational database, an internal file storage area (IFSA), an RDBMS Interface, a File Control Interface, and three API modules: the Object API (for defining data types and accessing entities), the Web API (for constructing HTML documents and processing forms), and the Tools API (for invoking external applications). The three API's provide all the necessary methods for implementing CGI-based graphical Consoles. A Web server with a WRM may support multiple consoles for different applications. The design is illustrated in Figure 1. Each component is described below.

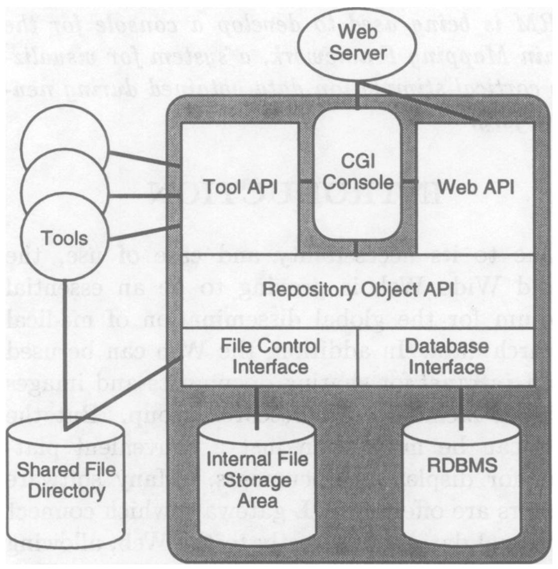


Figure 1: The WRM Architecture

The Repository Object API

The Object API is a collection of functions which allow the programmer to define new object types, import or create instances, edit existing instances, manipulate sets of instances, and pose queries over the data. An object may have more than one physical component, but the Object API allows the CGI programmer to refer to the object as a single entity. For example, an MR-image object consists of a file in the IFSA and descriptive information in a database table. A request to retrieve an image by name would invoke an Object API function that looks up the image ID in the File Description Table, then retrieves the appropriate file from the IFSA. In addition to providing a convenient interface for the CGI programmer, the Object API enforces consistency between the IFSA and the DBMS.

Every data item entered into the WRM will be represented in terms of the Repository Data Model. The model defines four classes of data objects: atomic types, files, composite types, and aggregate types.

Atomic types are strings, integers, and real numbers. Future versions of the WRM architecture may support additional atomic types, such as Dates and URL references, but currently these types can be represented by the existing atomic types.

A *File object* consists of a binary file, a unique file ID, and metadata about that file, such as its external name, format, author, date of creation, version number, etc. The file itself is stored in the IFSA, while the metadata are stored in the File Metadata table in the relational database. The file can be of any type, such as an image, an audio or video clip, an HTML document, configuration file, script, or even a binary executable.

A *composite type* is any user-defined data object, consisting of a unique file ID and an ordered list of attributes. For example, the Brain Mapper might define a *Patient* type consisting of three attributes: *name* (a string), *photo* (a file), and *exam* (a reference to another composite object). When the user defines a new object type, a new table for that type is created in the database, with columns corresponding to the attributes. For each instance created, a row is entered into the table for that type.

An *aggregate type* is a collection of references to Repository Objects. Two types of aggregate objects are Lists (ordered) and Sets (unordered). The Repository API will provide methods for creating, manipulating, and iterating over Lists and Sets. The results of a query are contained in a List. Aggregate types may be assigned IDs and saved in the repository.

Internal File Storage Area

The Internal File Storage Area (IFSA) is a private directory with a large enough capacity to contain all the multi-media data managed by the system. All access to the IFSA is mediated by the File Control Interface (FCI). The FCI is a collection of functions for maintaining revision control over files in the IFSA. A files is ***checked-out*** by an application that intends to change its contents, and ***checked-in*** when the revision is complete. A file checked out with an ***exclusive lock*** cannot be accessed by another process until it is checked back in. The FCI maintains a ***version history*** for each file object, and can supply earlier versions of files upon request.

Files submitted to the WRM are given a unique name by the File Control Interface and copied into the IFSA. When the Web server or a Tool requests a data object that consists of one or more files, the FCI copies the file(s) into a shared directory before they can be accessed.

Relational Database

Non-file data are stored in any standard relational database. The database should support SQL and allow access routines to be embedded in an external programming language. A client-server database is typical, with the server fielding requests from the repository's Database Interface. The database contains a File Metadata table describing the files in the IFSA, and a table for every composite type defined by the Object API.

The Database Interface provides functions to the Object API for creating and deleting tables, inserting and removing records, accessing objects by ID, formulating SQL queries, and retrieving the results of a query into a List object.

The Tool Interface

The Tool Interface is an expandable collection of modules, one for each external application controlled by the Console. Each tool has an Invocation function, which can be called by the CGI console to start up the tool. In addition, the Tool Interface provides transfer methods for downloading files from the repository into the application, and for importing files into the repository. If necessary, the transfer methods translate between repository objects and the raw data files expected by the tools.

The Web API

The Web Interface is a collection of functions for generating HTML documents, creating and parsing interactive forms, and formatting repository data for display. The Web Interface abstracts away most of the tedious detail involved in HTML programming, allowing the CGI programmer to rapidly develop interactive consoles. For example, the Web API provides a high-level function for displaying a list of repository objects as an HTML table.

Creating form elements and parsing the results of a form submission by hand can be a laborious and error-prone task, but the Web API provides a clean and simple environment for handling forms. For example, the Web API provides a function for creating pull-down menus that returns a value indicating the user's choice.

IMPLEMENTATION

We have partially implemented the WRM on a Unix workstation for supporting the development of the Brain Mapper's Web console. Wherever possible, we are utilizing freely available shareware products, which would enable other groups to install the WRM in their own development environment with minimal expenditure. The implementation of each component is described below.

API Implementation

The Repository Manager's API modules are being developed in Perl[3], a freely available, portable language designed for easy manipulation of text, files, and processes. A large number of Perl modules have already been implemented for processing Web forms and accessing relational databases. These modules can be retrieved for free from the Comprehensive Perl Archive Network[4], and can be plugged directly into the WRM architecture.

The Web-based console for driving a WRM application is implemented as a collection of of Perl scripts residing in the CGI directory of a Web server. The Web console for the Brain Mapper is shown in figure 2. Because Perl does not require a seperate compilation stage, it facilitates fast prototyping and testing of CGI scripts. For example, when adding a new feature to the console, the programmer merely edits a Perl script and clicks "reload" on the Web browser to instantly view the changes.

Perl's rich pattern-matching and report processing features make it ideal for parsing forms and generating HTML documents. The WRM's Web API utilizes a Perl module called *CGI.pm*[5], which provides a simple interface for interpreting query strings passed to CGI scripts, and a rich set of functions for creating fill-out forms.

Perl's utilities for handling Unix processes and controlling the data flow between them makes it well-suited for implementing the Tool API. To support the Brain Mapper, we are planning to interface three tools to the WRM: a remote image server for acquiring radiology data, a commercial data visualization package called AVS, and an in-house graphics program called Skandha4. For each of these applications, the Tool API will provide high-level routines for invocation and data transfer.

File Control and Database Interfaces

The File Control Interface utilizes Revision Control System (RCS)[6], a standard Unix tool for archiving version histories.

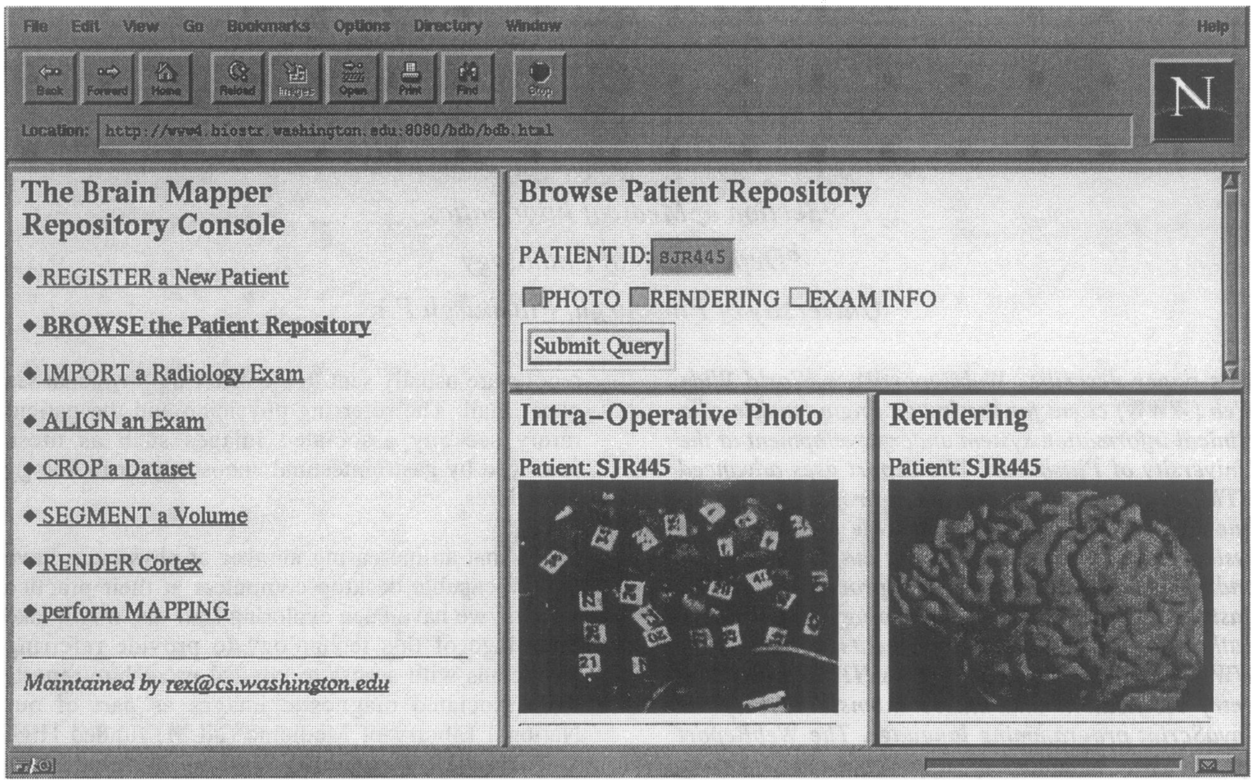
The relational database for the current version of WRM is *MiniSQL*, a lightweight client-server RDBMS[7]. Although MiniSQL provides only a subset of SQL as its query interface, it is free, speedy, memory efficient, and supported on a wide range of platforms.

The Database Interface module consists primarily of the *Msql Perl Adaptor* (Msql.pm), a simple interface to MiniSQL that allows a CGI script to establish connections with the database server, create and delete tables, retrieve metadata, and pose standard SQL queries. The Repository Object API passes an SQL query string to the Database Interface, which returns a Perl array populated with the query result.

DISCUSSION

Although the WRM is still in early stages of implementation, it is already proving useful as a programming environment for the rapid development of Web applications. The modular nature of the WRM architecture allows each component to be implemented separately, and Tool interfaces are added incrementally. The Web API is a highly effective utility in its own right, and can be used as a standalone interface for general purpose Web development.

Because the WRM is being implemented with off-the-shelf freeware products, when the develop-


File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
N
REGISTER a New Patient
BROWSE the Patient Repository
IMPORT a Radiology Exam
ALIGN an Exam
CROP a Dataset
SEGMENT a Volume
RENDER Cortex
perform MAPPING
Submit Query

Location: http://www4.biostr.washington.edu:8080/bdb/bdb.html

The Brain Mapper Repository Console

Maintained by rex@cs.washington.edu

Browse Patient Repository

PATIENT ID: SJR445

[✓ PHOTO] [✓ RENDERING] EXAM INFO

Intra-Operative Photo

Patient: SJR445

Rendering

Patient: SJR445

Figure 2: Example of a Web console utilizing the WRM

ment stabilizes it can be freely distributed for use on any Unix platform. In addition to the Brain Mapper console, we foresee the WRM supporting a wide range of applications, such as a picture archive, an anatomic knowledge base, and an experiment management system. To support clinical information systems, the components could be upgraded, such as reimplementing the API in C++ for higher efficiency, and replacing MiniSQL with full featured commercial RDBMS.

In addition to the standard facilities of a RDBMS, a repository manager manages multimedia data objects and integrates a diverse set of tools through a common data model. As the metadata management requirements of Web applications become more demanding, the use of a repository manager can significantly improve a CGI programmer's productivity.

References

1. P.A. Bernstein and U. Dayal. An overview of repository technology. In *Proceedings of the 20th VLDB Conference*, September 1994.

2. B. R. Modayur, J. Prothero, C. Rosse, R. Jakobovits, and J.F. Brinkley. Visualization and mapping of neurosurgical functional brain data onto a 3-D MR-based model of the brain surface. In *AMIA Fall Symposium*, 1996.

3. L. Wall and R.L. Schwartz. *Programming perl.* O'Reilly & Associates, Inc., Sebastopol, CA, 1991.

4. The comprehensive perl archive network. http://www.metronet.com/perlinfo/.

5. L. Stein. CGI.pm - a perl5 CGI library. http://www-genome.wi.mit.edu/ftp/pub-/software/www/cgi_docs.html.

6. W. F. Ticy. RCS - a system for version control. *IEEE Software Practice and Experience*, 15(7):637–654, 1985.

7. D. Hugues. Mini SQL: A lightweight database server. http://bond.edu.au/People/bambi/-mSQL/.