MADGE: scalable distributed data management software for cDNA microarrays

Richard A. McIndoe*, Aaron Lanzen and Kimberly Hurtz

Department of Pathology, Immunology and Laboratory Medicine, University of Florida, Gainesville, FL 32610, USA

Received on February 14, 2002; revised on May 21, 2002; accepted on June 17, 2002

ABSTRACT

Motivation: The human genome project and the development of new high-throughput technologies have created unparalleled opportunities to study the mechanism of diseases, monitor the disease progression and evaluate effective therapies. Gene expression profiling is a critical tool to accomplish these goals. The use of nucleic acid microarrays to assess the gene expression of thousands of genes simultaneously has seen phenomenal growth over the past five years. Although commercial sources of microarrays exist, investigators wanting more flexibility in the genes represented on the array will turn to in-house production. The creation and use of cDNA microarrays is a complicated process that generates an enormous amount of information. Effective data management of this information is essential to efficiently access, analyze, troubleshoot and evaluate the microarray experiments.

Results: We have developed a distributable software package designed to track and store the various pieces of data generated by a cDNA microarray facility. This includes the clone collection storage data, annotation data, workflow queues, microarray data, data repositories, sample submission information, and project/investigator information. This application was designed using a 3-tier client server model. The data access layer (1st tier) contains the relational database system tuned to support a large number of transactions. The data services layer (2nd tier) is a distributed COM server with full database transaction support. The application layer (3rd tier) is an internet based user interface that contains both client and server side code for dynamic interactions with the user.

Availability: This software is freely available to academic institutions and non-profit organizations at http://www.genomics.mcg.edu/niddkbtc.

Contact: rmcindoe@mail.mcg.edu

INTRODUCTION

A result of the human genome project is the exponential increase in the amount of DNA sequence information available to researchers to use in their experimental efforts. This increase has fueled a genomic revolution for investigators. The paradigm of analyzing a single gene effect in a biological system has shifted to a global systems analysis. Global gene expression analysis at the RNA level offers the first glimpse into the future of organizing and using genomic information. Using this technology, investigators can simultaneously monitor the RNA levels of a large number of genes or even the entire genome in the context of their biological system. In this article, we will describe a software application created to manage data generated in the creation of DNA microarrays spotted onto glass slides and use two color hybridization for data acquisition.

The basic strategy for a two color cDNA microarray experiment is to isolate RNA from two sources, a reference and an experimental sample (DeRisi et al., 1996, 1997; Eisen and Brown, 1999; Schena et al., 1995; Shalon et al., 1996). The RNA samples are converted to cDNA and labeled with a fluorophore, typically the reference is labeled with Cy3 and the experimental with Cy5. These two probes are combined and hybridized to the microarray. Following the hybridization and washes, the array is scanned at two wavelengths to detect the labeled cDNA that has hybridized to the array. The two computer images produced from the scanner are combined and the data for each spot (gene) is collected (along with background and error measurements). The data is expressed in the form of a ratio of experimental expression to reference expression. The hybridizations are repeated multiple times to ensure reproducibility and confidence in the measurement. Once the data from several hybridizations are generated, a variety of clustering and statistical methods can be used to help the investigators.

The Microarray Database of Gene Expression (MADGE) system is a 3-tier application that models the microarray workflow required to create and use cDNA microarrays, recording both the inputs and outputs generated by the processes. The MADGE system divides the microarray workflow into eight processes performed in two concurrent paths. One path focuses upon the processes necessary to transform a tissue sample into a labeled cDNA probe. While the second path focuses on the
Fig. 1. Sample workflow screen shots for the Quality Control step in the Amplification workflow in MADGE. (a) Queues page listing amplification plates ‘queued’ and ‘in process’ for the QC step of amplification. (b) The user has selected two plates to stop the QC step and must record the result for this step, either pass or fail. (c) Once the plates have passed QC they are automatically queued for purification.

creation of the microarrays themselves, including library construction/importation, clone amplification/purification, glass slide preparation and microarray printing. The two paths merge at the hybridization step and continue through the workflow terminating at the submission of the extracted feature data for the microarray.

SYSTEMS AND METHODS

The data access layer
We use SQL Server v7 as the relational database management system (RDBMS) for MADGE. The application uses two databases, one for the array workflow and the other for the employees. The ArrayWorkflow database contains 67 tables with 275 attached stored procedures. The database schema models the flow of data generated during the microarray workflow, including reagent lot numbers, control data (e.g. user IDs and system dates), and data from end deliverables (e.g. feature data and robot files). The SQL scripts needed to generate the two databases will be available in the final MADGE application package.

The data services layer

We could have written an internet application that contained all the database logic interspersed within the business logic. However, this would have failed to meet our requirement that the system be scalable, manageable and portable. Our current design has the advantage of providing an object oriented API for the programmer as well as the ability to separate the application logic from the database logic. Therefore, making the system scalable, easy to use and more secure. The API is a distributable COM server (DLL) written in Visual Basic v6 and compiled using apartment model threading and optimized for pentium processors. This server contains one transactional and three non-transactional classes. Each class provides methods to retrieve, insert and update information in the system. For example, the Queues class contains 12 methods with 89 options, the transactional ArrayWorkflow class contains 14 methods with 122 options, the ArrayData class contains four methods and the Help class contains the getters and setters for context specific help.

The application layer

The application layer is the user interface for MADGE, serving as the portal the end user will use to interact with the application. We wanted to build an interface that not only gave the user an organized environment for uploading and retrieving microarray data, but also provides guidance for the day to day experimental procedures (Figure 1). In this respect it would be similar to a laboratory information management system (LIMS) for microarrays. The MADGE system uses an application model of samples moving through workflow queues by having their internal status change as work is accomplished. We store and track all relevant pieces of information, ranging from when the user started and stopped the experiments to reagent lot numbers. This data is important for tracking problems when experiments fail in the facility.

The Graphical User Interface (GUI) is an HTML based visual display generated dynamically by ASP pages. The
ASP pages use both Vbscript and JavaScript to provide application logic on the server and client sides, respectively. Code written in JavaScript includes both standard form validation and administrative functions. By utilizing client-side JavaScript for form validation, the communication between the client machine and the web server is minimized while enforcing database field data type constraints. In addition to the workflow functions, the graphical user interface provides the additional capabilities of data retrieval, administrative functions, and miscellaneous tasks. A bug reporting system was incorporated into MADGE and permits users to enter any suggestions or bugs they may have encountered while using the system.

IMPLEMENTATION

Deployment flexibility

The n-tier logical design of the MADGE application provides for flexibility in the choice of physical architectures on which the application can be deployed. Logically, the MADGE system is divided into three components: an application layer, data services layer, and data access layer. In a one-machine install, the web application, DCOM server, and RDBMS reside on the same machine and processing occurs locally. This is the easiest installation to complete. In our deployment of the system, we chose to place both the data services and data access layers on the database server machine, while the application layer resides on a separate web server machine.

Documentation

To facilitate the deployment and use of the system, the MADGE application package includes several documents including a user manual, programmer documentation, and installation instructions. The 112-page programmer document was written for programmers and explains all enumerated cases for each method included in the madgesvr DLL. The inputs, outputs, and functionality of each method is outlined and an example of how it is called from the ASP pages is given along with a generic example of its usage.

The formal user guide contains step-by-step instructions for the use of the system. Each page of the 120-page document contains illustrations and an explanation of functionality and instructions for using that part of the application.

Future development

Throughout the testing and deployment phases of the MADGE system, the development team has identified several opportunities for improvement in the system.

Remove file dependencies with XML structure. The system currently requires the end-user to specify the format of the files to be uploaded in order to parse the file correctly. Each file format coincides with a software program written specifically for a given robot or third party vendor. A possible solution to resolve the issue would be to develop an XML schema for various data entry points, effectively shifting the focus of existing code to the common XML schema. In addition, the team will add support for exporting the data in a MAGE-ML (Brazma et al., 2001) compliant file. MAGE-ML has been submitted to the OMG by collaborating industrial and academic organizations as an XML standard for sharing microarray data. MADGE stores all the information necessary to produce a MAGE-ML compliant data stream. We will also add additional export options, such as SOFT at NCBI’s GEO, in future versions.

Add data analysis capabilities. The system currently stores all inputs generated during the workflow and allows the user to view and filter the outputs. However, the system lacks the capabilities of analyzing microarray data. In future versions, we will add interfaces for data analysis techniques such as clustering, principal component analysis and our custom statistical software.

ACKNOWLEDGEMENTS

The authors would like to thank Dr JinXiong She and his laboratory for helpful discussions and guidance in the design of MADGE. This work is supported by grants to RAM from the NIDDK Biotechnology Center and Howard Hughes Medical Institute.

REFERENCES


