

Web-Enabling a Scientific Laboratory Using SAS/IntrNet™

Thomas H. Burger, Source Consulting®, Indianapolis, IN
Richard W. Tucker, Eli Lilly and Company, Indianapolis, IN
John M. LaBore, Eli Lilly and Company, Indianapolis, IN

Abstract

Industry research and development increasingly relies on global operations. IT divisions now more than ever are challenged with developing an infrastructure capable of providing business solutions at this scale. SAS/IntrNet offers the ability to couple powerful analysis and reporting tools with global information delivery using the Internet. Speed-to-market initiatives demand rapid implementation of this capability. We describe web-enablement of a scientific laboratory providing end-users the ability to generate dynamic reports and graphics.

Introduction

The drug development process involves mass screening, testing, analysis and reporting which may take 10 or more years to complete. A primary objective of this process is, of course, to generate information upon which to base scientific decisions leading to a compound's discovery. The volume of information generated during compound evaluation plus interdependencies among studies results in the need to process data quickly.

Important strides in this area have been made with the advent of devices for automated data acquisition and instrument interfacing. However, a potential bottleneck exists where the rate of information collection outpaces its analysis and reporting. Information systems which close the gap in information delivery for decision support reduce the risk of this bottleneck (Fig. 1).

We describe a web-enabled SAS® analysis and reporting system that helps eliminate this bottleneck.

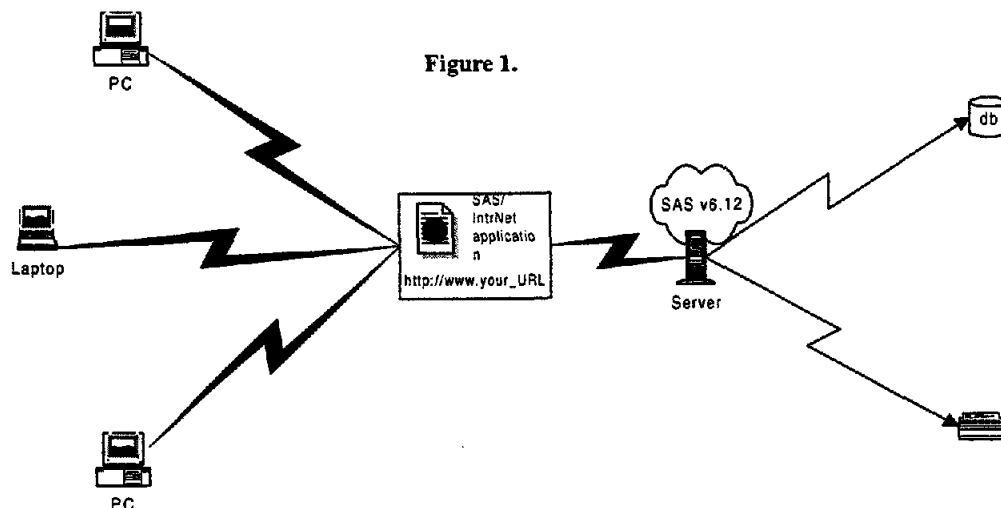
Business Drivers

A common dilemma among labs is the poor alignment of data processing with business practice. Data may exist in multiple formats or require manual intervention for standardization, error checking, subsetting, analysis or reporting. Frequently, these steps must be performed for every study. A lack of data processing automation requires manual entry or data exchange for accessibility. Further, remote access by end-users may only be a dream. At minimum, these tasks require specialized skills, are time consuming, and are riddled with duplication of effort. At worst, they interrupt scheduling, work flow, and compound evaluation. This situation is exacerbated as compounds move through the product pipeline and more researchers need access to data.

The time from data generation to product evaluation can be reduced by streamlining data processing to provide greater access to data, decision support, and electronic reporting. This can be achieved using:

- 1) An integrated software suite for building robust web-enabled analysis & reporting systems
- 2) A methodological framework upon which application developers can build web-enabled systems using a *rapid application development* (RAD) style approach.

The Internet provides a valuable means of global access; SAS/IntrNet provides an ideal mechanism for coupling powerful analysis and reporting tools to it. The methods we describe leverage the integration of SAS system components while maintaining design plasticity; this leads to robust systems. Variations in the actual implementation you select are met through planning *before* you build.



In accordance with standard software development practice, we implemented a Software Development Life Cycle (SDLC) approach to maximize return on development time (Sloan 1993). This front-end loads effort to the design phase, but saves time downstream by ensuring testability and extensibility (Newhouse 1997).

Requirements and Design

Requirements are a formal definition of the criteria to be met by system functionality. **Business requirements** are system specifications written using business terminology to identify system features necessary from an operational perspective. **Software requirements** are system specifications which expand business requirements to a finer resolution while incorporating a translation into IT nomenclature.

Requirements definitions interface the client's business needs with the system developer's IT solution. A clear requirements definition of functional scope is the basic premise upon which a design foundation is laid. A set of business and software requirements are defined as the basis for system development.

System design provides the architectural blueprint detailing a system's processing logic and physical structure. It aids in conceptualizing what to build through schemata describing how components in the problem domain relate to one another. Its role is critical since this is where requirements are linked to system entities and where a breakdown may occur while translating requirements to code (Fig. 2). The stability, testability, efficiency and maintainability of a system rests in the architecture because it defines the context in which code is executed (Burger and Pochon 1997).

Frequently, SAS analysis and reporting 'systems' are implemented as a sequence of nonintegrated or one-off programs wherein a design is lacking. While appropriate in many cases, this has limitations from a systems engineering perspective. First, efficiency of data processing is compromised. Second, robustness is incorporated into systems largely through design. Third, SAS contains an array of hooks expressly intended to integrate its tool suite. Fourth, SAS systems development conforms nicely to the SDLC.

Our approach to building a web-enabled analysis and reporting system rests in the need for it to be simple, maintainable and extensible. We focus on design considerations to assist us with these objectives. As a design methodology, we adapted *Structured Design Methodology* (SDM). SDM is a formal method of organization for designing a system's components and their interrelationships to best meet a set of requirements (Yourdon and Constantine 1975). It uses functional decomposition and modularity to divide complex abstractions into a hierarchy of smaller elements with clearly defined solutions. We use a CASE (Computer-Aided Software Engineering) tool to model system architecture (Burger and Pochon 1997). These methods are equally suited to macro system and AF/SCL[®] application development, as well.

After defining requirements, we identified processes of the *logical design*, as illustrated in Figure 3. By partitioning the problem domain, relationships and dependencies were clarified. Physical components and processing mechanics are addressed in separate diagrams later in design.

The *physical design* of the system is constructed by decomposing its logical components into a *Data Flow Diagram* (DFD) and *Functional Hierarchy Diagram* (FHD). The DFD models the flow of data through physical structures in the system (Fig. 4).

Figure 2.
Software Development Cycle
Waterfall Model

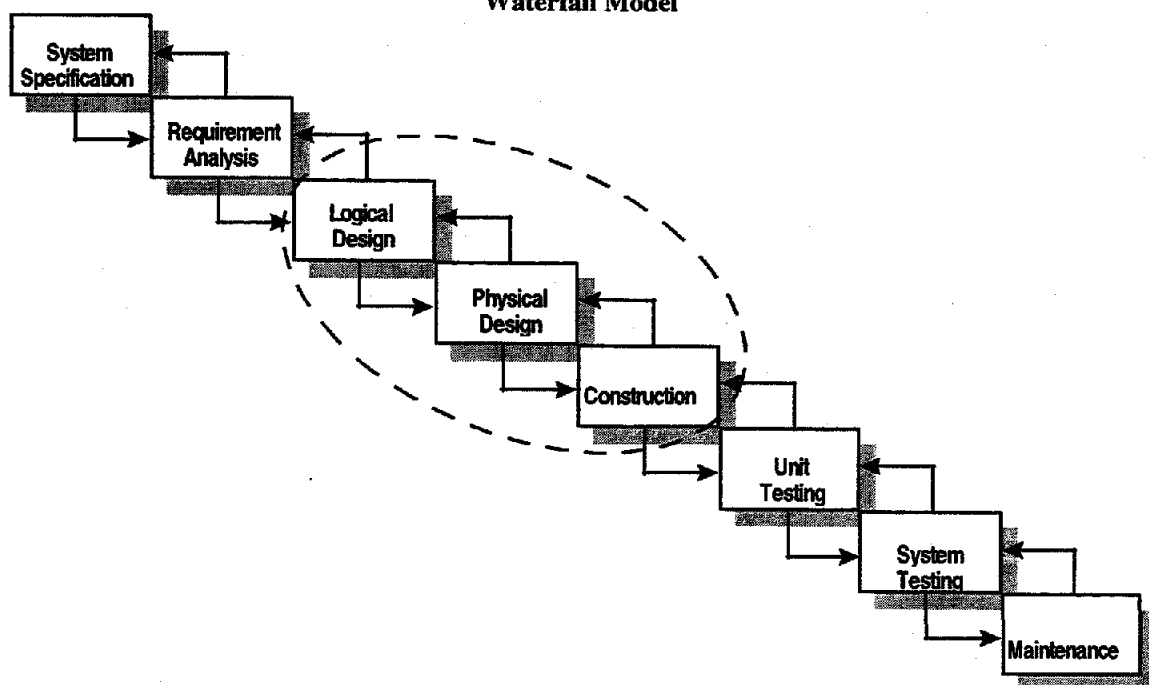
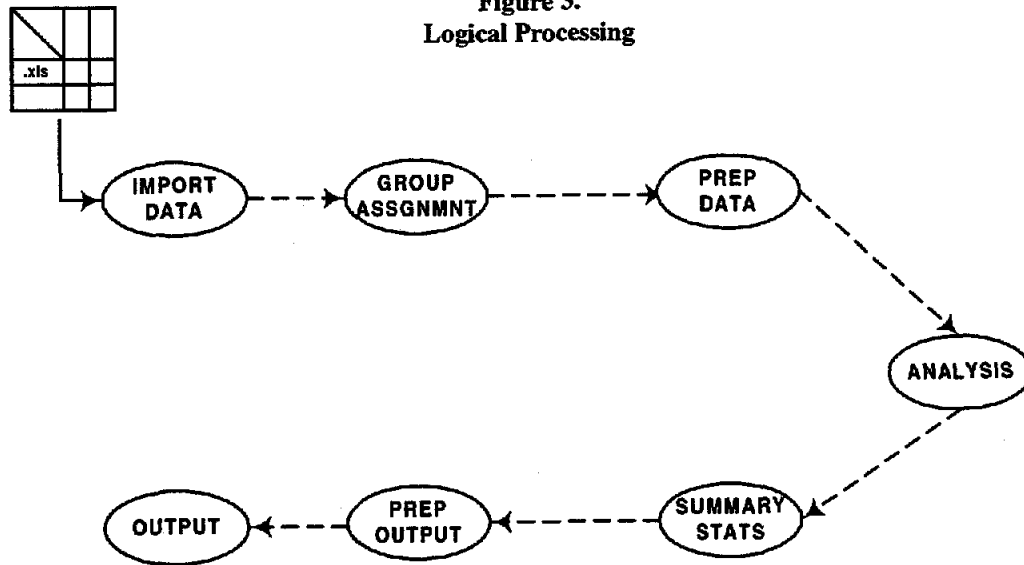


Figure 3.
Logical Processing



The *Functional Hierarchy Diagram* (FHD) models program organization. FHDs are horizontal decomposition charts that represent the hierarchy and calling sequence of a system's modules (Fig. 5). Decomposition limits redundancy by revealing sites of consolidation and shared functionality. Modularization permits the creation of well-structured processing thereby limiting the risk of spaghetti coding.

Modules are a logical grouping of functions whose collective effect is to perform a defined process. To impose hierarchy, externally defined macros are divided into control functions and terminal functions. **Drivers** control modules by calling

secondary drivers or sub-drivers. **Sub-drivers** control small modules by calling secondary sub-drivers or utility functions. **Utilities** are terminal functions that perform specific operations and are typically <20 lines of code (Burger and Pochon 1997).

Attention to design aids in the integration of system architecture; this contributes to stability and ease of maintenance. It also provides a mechanism to insure that requirements are met. Modularization produces programs with discrete tasks; this processing isolation results in high testability (Pochon and Burger 1998). Further, hierarchical structuring permits highly controlled processing and provides an extensible framework.

Figure 4.
Data Flow Diagram
(DFD)

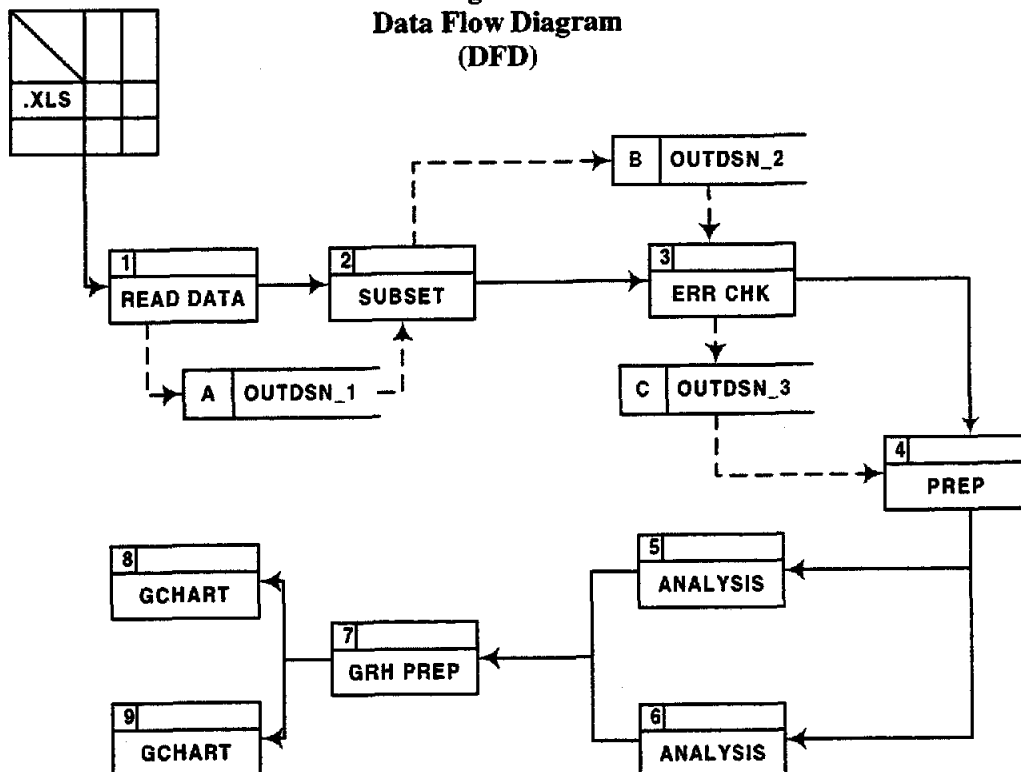
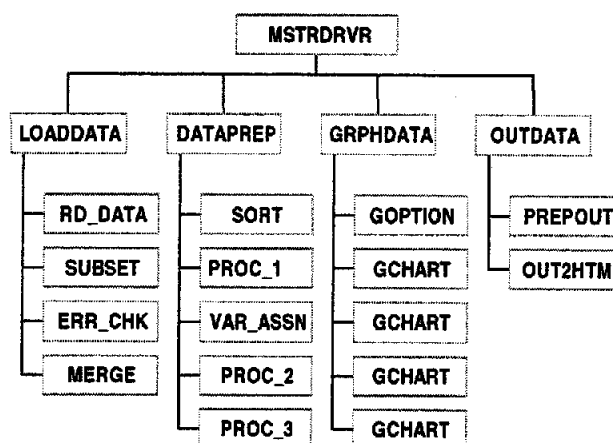


Figure 5.
Functional Hierarchy Diagram
(FHD)



Operating Environment

The application server is a Compaq® Proliant™ 6500 rack-mounted computer with two Intel® Pentium Pro™ 200 processors, 256 MB of ram, three 9 GB wide-ultra SCSI hard drives and a Compaq Netelligent™ dual 10/100 TX PCI UTP network interface. The SCSI hard drives are configured as a RAID5 array using a Compaq SMART™-2/P array controller, yielding a logical drive array of 17.351 GB, with 2 GB designated as the "C" or system drive and the balance as the "E" or data drive. The operating system is Windows NT Server™ 4.0 with Service Pack 3 SAS 6.12 and SAS/IntrNet 1.1 for NT Server are also installed. Nelson and Swirski (1998) discuss a number of factors to consider in choosing a "rightsize" server platform. Mehler (1998) provides a good overview of configuration parameters for optimal performance of SAS on Windows NT Servers. Henderson (1998) provides a "roadmap" for determining which components of the SAS System and SAS/IntrNet must be licensed to support a given application.

A Compaq Deskpro™ 66M was reconfigured for use as a web server. It contains an Intel 80486/66MHz processor, 64 MB of ram, a 2 GB SCSI hard drive and a 3Com® Etherlink™ III EISA network card. The installed software includes Microsoft Windows NT Server 4.0 with Service Pack 3, Microsoft Internet Information Server 4.0, Microsoft FrontPage™ 98 server extensions and the SAS/IntrNet Application Broker. This server also handles file and print services for a small workgroup of 10 individuals on a TCP/IP network.

Connected to the network via Compaq PCs are laboratory instruments such as chemistry analyzers, gas chromatographs (GCs), gas chromatograph/mass selective detectors (GC/MSDs), radioimmunoassay analyzers, pH meters and balances. The PCs are installed with Windows95 and Netscape® Communicator™, and have 32MB of ram, 2 GB hard drives and a Compaq Netelligent 10/100 TX Embedded UTP network interface.

System Construction

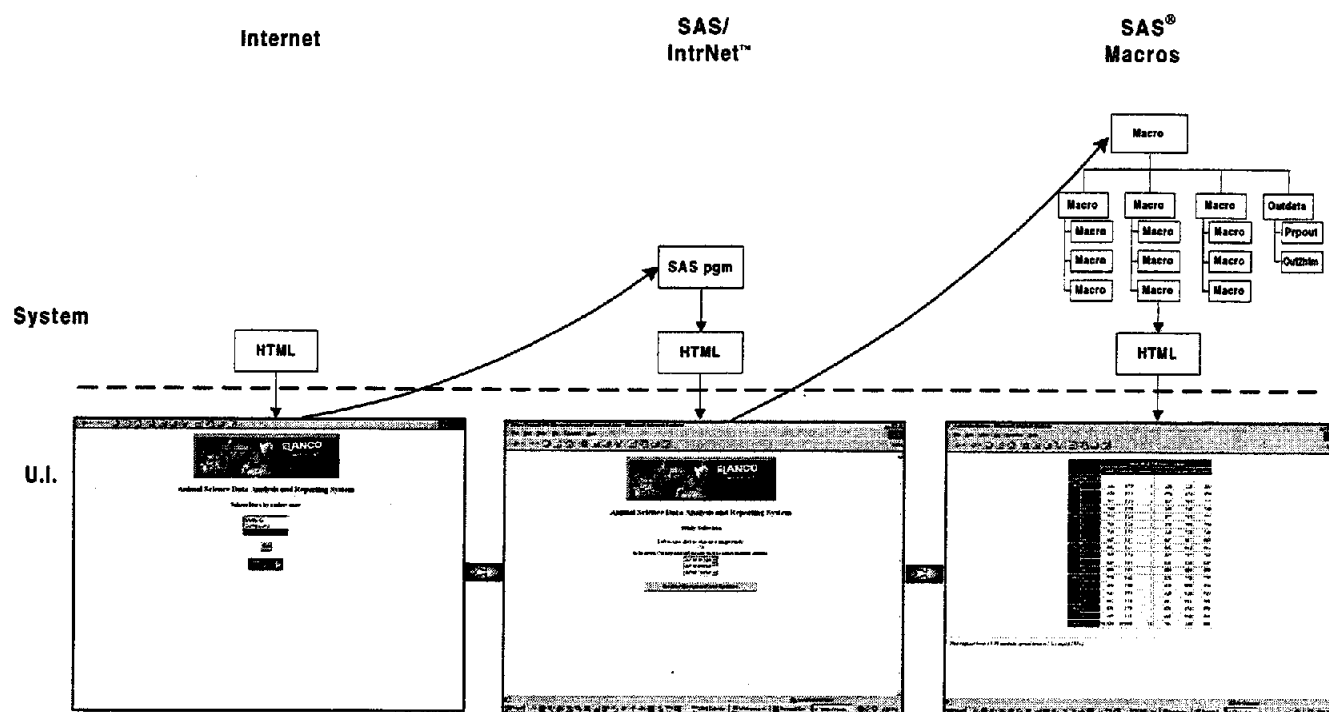
The SAS macro facility's appeal for constructing systems rests in its ability to produce powerful and robust modular components. Because it is an extension of BASE SAS, macros provide a easy means of encapsulating native routines (DATA step, PROCs). Since the macro facility uses its own language, it provides its own hooks (parameters, macro variables) for integration with other system components. Use of modular functions for conditional execution and dynamic code generation allows creation of sophisticated processing.

The SAS/IntrNet Application Dispatcher provides a web-based gateway to data and a powerful array of SAS analysis and presentation tools (Henderson et al. 1997). A significant advantage is that SAS is not required on the machine where the web client (browser) resides.

The gateway itself is written using the Common Gateway Interface (CGI) and eliminates the need to develop one's own CGI scripts to access data and SAS tools. Users simply select items and/or fill in forms on their web browser and submit the information to the web server. Security features are available. On the web server, the Application Dispatcher passes the transmitted information to a CGI program which, in turn, passes it to a running SAS session. Once the session runs the designated SAS program, it returns the results back through the CGI program to the browser where the results are displayed for the user.

Figure 6. illustrates a segment of HTML code for the initial screen of a sample application. Note that the SAS program (which is called after selections have been made from the initial screen) is designated using an HTML "name/value" pair. In this code, the underlying SAS program is designated by the name "_program" with a corresponding value of "sample.subsettr.sas". The macro variable name to be passed to the SAS program has the name "subset" and it can take one of three values: 'studyid', 'compound' or 'species'.

Figure 9.
SAS/IntrNet™
Application Design



Summary

We have applied standard software design to SAS web-application design and deployment. Significant gains in processing simplification and interpretation can be made using a formal systems development approach. The resulting design is highly modular and easily applied to system development. CASE tools provide a useful means of representing this structure and identifying relationships. The system created using this approach is easily maintained and highly testable. Remote access is coupled with on-line analysis and reporting for streamlined data processing.

References

- Burger, Thomas H. and Philip M. Pochon. 1997. *Structured Design and CASE Tool Use in SAS Macro Systems*. Proc. of the Eighth Annual Midwest SAS Users Group Conference. Chicago, IL.
- Gill, Paul. 1997. *The Next Step: Integrating the Software Life Cycle with SAS Programming*. Cary, NC: SAS Institute Inc. 384pp.
- Henderson, Donald J. 1998. *SAS/IntrNet Software: A Roadmap*. Proc. of the 23rd Annual SAS Users Group International (SUGI) Conference. Nashville, TN.
- Henderson, Donald J., Edmund Burnette, Vincent DelGobbo and John Leveille. 1997. *The SAS/IntrNet Application Dispatcher*. Proc. of the 22nd Annual SAS Users Group International (SUGI) Conference. San Diego, CA.
- McConnell, Steve. 1993. *Code Complete: A Practical Handbook of Software Construction*. Microsoft Press. Redmond, WA.
- Mehler, Gary. 1998. *Taking Advantage of the SAS System on Windows NT*. Proc. Of the 23rd Annual SAS Users Group International (SUGI) Conference. Nashville, TN.
- Nelson, Gregory S. Barnes and Richard Swirski. 1998. *Choosing the Right Platform for SAS Client/Server Applications – An Evaluation of Factors in Selecting NT and Unix*. Proc. of the 23rd Annual SAS Users Group International (SUGI) Conference. Nashville, TN.
- Newhouse, Russell. 1997. *Validation and SAS Programming: Benefits of Using the System Life Cycle Method*. Proc. of the 22nd Annual SAS Users Group International (SUGI) Conference. San Diego, CA.
- Pochon, Philip M. and Thomas H. Burger. 1998. *Validation of SAS Macro Systems*. Proc. of the Pharmaceutical Industry SAS Users Group Conference. San Francisco, CA.
- SAS Macro Language: Reference*, First Edition, Cary, NC: SAS Institute Inc., 1997. 304pp.
- SAS Macro Facility Tips and Techniques*, Version 6, First edition, Cary, NC: SAS Institute Inc., 1994. 319pp.

Information Systems

Sloan, Faith R. 1993. *Is the System Analysis Phase Really Necessary ?* Proc. of the 18th Annual SAS Users Group International (SUGI) Conference. Miami, FL.

Yourdon, Ed and Larry Constantine. 1975. *Structured Design*. Yourdon, Inc. New York, NY.

Trademark Notice

SAS is a registered trademark or trademark of the SAS Institute Inc, in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

Acknowledgment

The authors would like to thank Eli Lilly and Company, under whose auspices this work was performed.

Author Contact

Thomas H. Burger
Source Consulting
Bank One Tower ~ Suite 3930
111 Monument Circle
Indianapolis, IN 46140
Phone (317) 631-2900

Richard W. Tucker
Eli Lilly and Company
2001 W. Main St.
Drop Code GL50
Greenfield, IN 46140
Phone (317) 277-4896

John M. LaBore
Eli Lilly and Company
Lilly Corporate Center
Drop Code 2133
Indianapolis, IN 46285
Phone (317) 277-6387

Authors' Note

This is an invited presentation based on our paper *Design and Deployment of a Web Application Using SAS/IntrNet* which appeared in the 1998 Pharmaceutical Industry SAS Users Group Conference Proceedings.