ROBOSAS: LABORATORY AUTOMATION AND SAS®

Charles Villarrubia, Syntex Research
Seth Michelson, Syntex Research
Ronald Rhodes, Syntex Research
Edward Leung, Syntex Research

ABSTRACT

This paper describes the process of laboratory automation and the role SAS® software can play in application development. Laboratory automation is a process of converting routine, repetitive tasks, performed by humans in a laboratory environment, to machine executable tasks supervised by human monitoring. This involves seamless integration of data collection, analysis, reporting, and database storage subcomponents across disparate platforms. Pure applications, which depend on one software package to accomplish these goals, are easier to maintain and development. Mixed systems exploit a variety of software to perform these same purposes and provide flexibility to system development. Concepts and tools from systems analysis and forethought in designing applications are crucial to a project's success. An example of a laboratory automation project highlights these points.

INTRODUCTION

Due to an increasingly demanding marketplace, pharmaceutical companies are putting more emphasis on productivity and efficiency. The competitive nature of this industry dictates that therapeutic agents be brought to market in as timely a manner as possible. On a routine basis, biologists and chemists screen large numbers of compounds for biological activity in various assays. This can be a time-consuming process involving data entry, statistical analysis, and report generation in addition to the data collection process of running the experiment itself. Also, raw and summary data need to be stored in a manner that facilitates search and retrieval. To the extent possible, productivity is enhanced by automating and standardizing these procedures. Bradshaw (1988) presented a rationale for a unified analysis environment using SAS software in Basic Research at Syntex and Villarrubia et. al. (1991) reported on software and system development strategies to expedite the analysis process. This process is being taken further as tools become available for seamless integration of data collection, analysis, reporting, and database storage across disparate platforms.

OVERVIEW OF THE PROCESS

Laboratory automation is a process of converting routine, repetitive tasks, performed by humans in a laboratory environment, to machine executable tasks supervised by human monitoring. The first task of the system developer is to analyze how the biologist is currently conducting experiments. Elements of this analysis include the data acquisition and review process, technological and human interface criteria, data flow and structure for each software component, statistical analysis requirements, reporting needs including graphics (for formal reports and laboratory notebooks), and database structures.

Data are collected using a variety of devices depending on the measurements or experiments needed. Data can be taken manually from scales or paper strip recorders, semi-automatically from output files produced by electronic strip recorders or plate readers, or automatically in robotic and bar-coded collection systems. While full automation is desirable, it is sometimes not possible given the array of collection methods. Third-party vendors usually provide the acquisition apparatus and some analysis or real-time review capability.

The collection, analysis, and database storage software applications may additionally reside on different hardware platforms creating challenges in information flow and integration. Typically, data acquisition software and hardware are microprocessor-based and may comprise multiple stations networked to a file server that is connected to a mainframe or mini-computer. The appropriate networking technology must be reviewed to insure that the elements of the whole system communicate and determine how information will be transferred.

PURE AND MIXED SYSTEMS

Applications can be divided into pure or mixed configurations. Pure systems use only the tools available in one software product to achieve application goals. One of the main advantages of this design is less complicated development and easier maintenance since only one application package is needed. SAS software is a very powerful product with a wide latitude of functionality. The variety of products within the SAS system, the default procedures provided, and a flexible programming language provides application engineers with a one-stop tool for most system components. Being a high-level language, SAS software can be more easily learned and maintained by a smaller staff, and development and prototyping of applications can be achieved quickly and efficiently. One situation that arises frequently in laboratory automation is transfer of data from the collection device to the analysis software. SAS software allows many types of file formats as input. In our experience, the input file is usually ASCII format but there may be extraneous information that must be ignored or discarded before data analysis can proceed. Appendix 1 shows a file produced by a robotic system and one solution to the input problem. Sometimes multiple files are produced and the information appended to provide a master data set. The ability of SAS software to interact with the operating system (MS-DOS® in this case) was essential in producing a workable input solution.

Mixed systems use a variety of methods to perform different system operations. One advantage is flexibility in dealing with different components. For example, on personal computer platforms, formatting and custom reports can be generated easily with macro languages in spreadsheets. These tools can be automated to a certain level but also give the user the flexibility to review their data before proceeding to the next level of analysis. Another advantage of this kind of system is the use of lower-level programming languages to control the different application components. Calls to the operating system may be essential to check for existence of files, invoking other software products and transfer of information from module to module once processing is complete. This orchestrating function may be critical to the functioning and performance of complex systems. SAS software has limited capability as a controlling entity that mitigates its use in some sophisticated systems. The role of SAS software in laboratory automation could be enhanced by expanded competence interacting with operating systems and calls to other application programs. Mixed systems, however, are more complex and difficult to develop due to the variety of skills needed for their implementation.

Despite the benefits provided either with mixed or pure applications, many companies fall back on mixed systems because of history. As mentioned previously, collection devices, spreadsheets, graphics packages are usually micro-processor based, whereas databases reside on larger computers. This fragmented state of affairs usually exists prior to the desire or need for an integrated system. For example, many biologists keep separate databases of their laboratory operations on personal computers or Macintosh® and use the tools common to those systems on a daily basis. However, corporate and project evaluation managers, among others, need summary data from a variety of sources to make tactical and strategic decisions. Information that is scattered on individual and corporate databases needs to be merged to provide a more unified and diverse use of company data. This being the case, one of the main tasks in laboratory automation is defining data structures and information flow between modules and to perform a variety of functions over numerous potential users.

INFORMATION FLOW

System analysis

Success of a complex system augurs for rational analysis before implementation. Analysis and forethought increase the probability that each of the functions of the system (e.g. data collection, analysis, and storage) will have the information needed to carry out those tasks. Since different departments (e.g. biologists, statisticians, information services) may be involved in system development, this coordination is crucial to the success of the project. Principles of structured analysis (Yourdon 1989) are beneficial to this effort although strict adherence to specific project life cycles may not be achieved. Most projects usually have a survey phase assessing user requirements and desires, an analysis phase defining functional specifications and hardware requirements, a design, coding and testing phase, and installation. During the survey phase, biologists, managers, and other users are interviewed to determine the scope and feasibility of the system, deficiencies in the current environment, what functions the new system is to perform, and what information is needed to accomplish those goals. The analysis phase consists of formalizing the input from the survey into functional specifications, delimiting procedural changes the system requires, and modeling the user environment. In the design phase, the structure of the system is developed. Separate subcomponents are identified (e.g. data collection, analysis) and software modules and

interfaces are delineated among them. The components are coded, tested and integrated and retested using data supplied by the biologist. Once testing is complete, the product is installed and tested in the user's environment and the biologist's trained in its use.

Systems are implemented to perform some task, that is they are functional. The user survey determines what the biologist needs and desires the system to do. The survey also illuminates how the biologist conducts experiment and how information flows from collection to analysis. A model describing what the system must do to meet these requirements is called the essential model (Yourdon 1989). This model conceptualized the system the user wants not what is currently being done. The essential model consists of the environmental model and the behavioral model. The environmental model basically describes the context in which the system must operate and the interfaces between the system and the environment. Event lists describe to what environmental occurrences the system must respond, typically interactions with the biologist. The behavioral model describes the "inside" of the system or how the system deals with the environment. This model uses tools as data flow diagrams to picture the functions of a system and the input and outputs to each process such as producing a report. These are tools that help conceptualize, operationalize, and implement the application.

Vendor Procurement

All systems are mixed in the sense that the devices used to collect the data, which will be different for each type of experiment, usually do not provide adequate means of data analysis, reporting, or storage and thus other packages are needed to provide the additional functionality. Numerous vendors should be interviewed to determine if the product meets the collection needs of the biologist and can be interfaced with other components of the automation protocol. Most packages can create ASCII files containing the collected data although some create files in proprietary formats. Ease of input can be a major concern. The ability to create user-defined output files, instead of a default file with extraneous data, or worse, random "garbage" characters, facilitates the input process. Real-time review of data is advantageous; laboratory personnel or managers need to monitor the collection process. For example, electronic strip recorders can display data as it is collected. If a rat, say, has become disconnected from the leads to the device, this will be obvious on the display and the animal can be reconnected. Review capability and the descriptive statistics provided are usually not sufficient for in-depth analysis and summary across experimental units. Many collection systems do not allow user-defined fields to flexibly define treatment conditions (e.g. vehicle controls, treated group). As a result, treatment and other experimental information must be gathered off-line from the user and merged with the raw data. Selection of vendors by a team of biologists and information services personnel is essential to determine if the equipment can do the job.

MacLAB ®/SUNR LABORATORY SYSTEM

The MacLAB/SUNR system was developed to automate data acquisition, review, analysis, and storage for functional tissue studies. In the previous process, data collection involved the time-consuming task of reading raw data from paper strip recorders, keying this data into a SAS/AF® application for analysis, and manually entering the results into an ORACLE® database for storage. Additionally, these subcomponents resided on different platforms, Macintosh for data collection and VAX® for analysis and storage. With the advent of electronic strip recorders for collection, networking technology between VAX and Macintosh, and the desire to expedite this process, efforts at automation were initiated.

Description of the assay

Elucidation of agonist and antagonist affinities in functional assays using concentration-response relationships is an important element in pharmacology. Functional assays exploring these connections characterize receptor function and drug selectivity. Tissues are suspended in a solution bath between two strain gauges, stimulated with agonist alone, in the presence of an antagonist or test compound, and the resulting tension measured. A response curve is generated at multiple concentrations and characteristics of the curve such as EC_{50}, slope, minimum, and maximum are calculated. Shifts in the curves or parameter changes indicate varying binding affinities and functional consequences of the compounds being tested.

Overview of the components

A single station, driven by a microprocessor, monitors multiple tissues simultaneously. There may be multiple stations in the laboratory. Data from the tissues are collected by an electronic strip recorder and displayed, graphically on a monitor and in hard copy, in real time. A microprocessor functioning as a file server coordinates and collates data from each data acquisition station for upload and analysis. The file server and each station are connected through a local area network, with the file server additionally

connected to a VAX minicomputer. The biologist can review and select data from each station to be sent to the VAX for analysis and storage into an ORACLE database. An ORACLE interface allows the biologist to associate explanatory (e.g. compound, dose, assay) information with the raw data. Data are uploaded and analyzed in SAS using the SUNR (Seemingly Unrelated Nonlinear Regression) methodology (Leung et. al. 1992, Emmrich, 1991, Muller and Helms 1984) and passed back to ORACLE for data archival and reporting. Summarized data are also provided by the SUNR system to the microprocessor for graphical display.

Systems analysis

Protocols define how the experiment is conducted, the appropriate statistical analysis to use, and the information to store in the database. The biologist's needs were surveyed to ascertain system functions, interface criteria, data acquisition processes, and data transformations. Based on the survey and protocol information, functional specifications were written delineating the tasks and input/output requirements of each of the subcomponents (e.g. acquisition, analysis, storage) as well as the hardware needs for the system. The system would operate on different platforms since the biologists used Macintosh Quadras® as microprocessors and the ORACLE database resided on a VAX minicomputer. The networking hardware (Ethernet®, Appletalk®) and software were purchased and installed to allow transfer of information between platforms. As part of the automation process, biologists were replacing the current paper strip recorders with electronic recorders to measure tissue tension. These devices, supplied by different vendors, were evaluated regarding their collection capabilities and how that data could be incorporated into more a complete analysis and storage system. This evaluation was facilitated by the specification that described the output needed from the acquisition device. This automatic collection process and network connections obviated the need for manual transfer of data between the various subcomponents as had been done previously.

Description of the system

The data acquisition system consisted of a Macintosh Quadra and MacLAB software and hardware to run the electronic strip recorder. The strip recorder, connected to eight tissue baths, records grams of tension sampled at user-defined intervals, and displays the data or produces hardcopy output. Biologists can monitor data collection in real-time and make corrections for calibration or deal with problems (e.g. torn tissue) as they arise. As defined in the specifications, the collection software can identify the tissue, a peak value of tension recorded at a specified time, and markers of events to the data stream to identify when compounds are administered to the tissue. The user desired to review the data and obtain summary statistics on the raw data before uploading to the VAX for more processing. The output from the MacLAB is pasted into a Microsoft® EXCEL® spreadsheet to facilitate the review process and ASCII files are created for upload to the VAX. Once the biologist is satisfied that the data are acceptable, a file server is used to collate the separate files from the multiple stations that are then uploaded to the ORACLE database on the VAX. An ORACLE interface queries the user for information which identifies the data, for information needed by SAS for analysis, and performs some minor data manipulation before formatting an ASCII file for SAS. Additionally, the ORACLE program invokes SAS for the SUNR analysis, and creates a file of summary information, defined in the specifications, for input into the database.

The SAS SUNR software provides the biologist with the means to fit curves from agonist and antagonist experiments when each subject in the experiment receives every dose in every curve. Curves are fit using a four-parameter logistic model estimating the parameters EC_{50}, slope, maximum, and minimum and their standard errors using SUNR methodology. For antagonist experiments, the PA_2 and pKB, other pharmacological measures, are also calculated along with standard errors. The program also checks the similarity of the curves, which if different, violate assumptions of the SUNR model, and warns the user. The curve estimates and the warning information are put in an ASCII file which the ORACLE program reads for input into the database. A second file contains data that the user can download to the Macintosh for graphing purposes. When the SAS program has finished, the ORACLE program loads the results of the analysis into the database and produces a custom report for the biologist.

The SUNR program was originally written as an interactive system in version 6.08 of SAS/AF on the VAX. This allowed the biologist to choose the model parameters and their values. The automated SUNR system needed to be invoked in the batch mode by ORACLE and possibly accessed by multiple users simultaneously. Due to this requirement, SUNR was rewritten using a macro program to call the modules of the SUNR programs which paralleled the calling structure of the original SAS/AF program. The main SAS program is a macro (Appendix 2) which uses %INCLUDE's to call the modules of the SAS/AF program and the macro language to provide the looping structure since the %INCLUDE's are

called multiple times. The ORACLE program invokes SAS with an autoexec.sas which defines libnames and the name of the main SAS macro program. Additionally, code needed to be created in a dynamic manner. DATA _NULL_'s were used to create the code fragments and %INCLUCE them back into the static code (Appendix 3). The flexibility of SAS, which allowed transfer of an interactive system using SAS/AF to a batch system using macros and %INCLUDE's, and the ability to be called by a controlling program, was a main contributor to the success in automating the SUNR methodology.

CONCLUSIONS

Automation increases the productivity in laboratory environments by allocating machine resources to repetitive, time-consuming tasks. Various software and hardware products and information are coordinated to accomplish this goal. Forethought and analysis, using tools form structured analysis aid, in this process. SAS software has powerful and flexible tools to implement laboratory automation, either in pure or mixed systems.

ACKNOWLEDGMENTS

SAS, SAS/AF are registered trademarks of the SAS Institute, Cary, NC.
VAX is a trademark of Digital Equipment Corporation.
Macintosh, Quadra, Appletalk are trademarks of Apple Corporation.
Microsoft, MS-DOS, EXCEL are trademarks of Microsoft Corporation.
MacLAB is a trademark of AD Instruments.
Ethernet is a trademark of Xerox Corporation.
ORACLE is a trademark of Oracle Corporation.

LITERATURE CITED

Bradshaw, M. 1988. The selling of SAS software: toward a unified analysis environment in basic research. Proceedings SAS User's Group International. 13:475-476. SAS Institute. Cary, NC.

Emmrich, D., L. Archer, C. Villarrubia, G. Silva. 1991. SAS automated IC_{50} parameter estimation with repeated measures data. Proceedings SAS Users Group International. 16:1154-1156. SAS Institute. Cary, NC.

Leung, E., S. Michelson, C. Villarrubia, L.A. Perkins, and R.M. Eglen. 1992. Analysis of concentration-response relationships by seemingly unrelated nonlinear regression (SUNR) technique. Journal of Pharmacological and Toxicological Methods: 28:209-216.

Muller,K.E. and R.W. Helms. 1984. Repeated measures in nonlinear models. Presented at ASA meetings in Washington, D.C.

Villarrubia, C., G. Silva, B. Ingraham, and S. Michelson. 1991. Toward a unified analysis environment: Two years later. Proceedings SAS Users Group International. 16:1220-1223. SAS Institute. Cary, NC.

Yourdon,E. 1989. Modern structured analysis. Prentice-Hall. Englewood Cliffs,NJ.

AUTHORS CONTACT

Charles Villarrubia
Syntex Research
3401 Hillview Avenue
MS S3-1B
Palo Alto, CA 94306
415-855-6524

APPENDIX 1
EXAMPLE OF FILE FROM ROBOTIC COLLECTION SYSTEM AND SAS CODE FOR INPUT

```
DATA INP;
  INFILE 'INPUT.DAT' PAD;
  INPUT IN $ 1-3 @;
  IF UPCASE(IN)=:'TOT' THEN DO;
    STOP;
  END;
  IF IN NE '' THEN DO;
    DELETE;
    RETURN;
  END;
  ELSE IF IN EQ '' THEN DO;
   INPUT
        PEAK     25-27
        AREA     32-35;
   END;
  IF PEAK=. AND AREA=. THEN DELETE;
RUN;
```

INPUT FILE:

```
Report Format File: UNTITLED
Detector Condition Summary
Cell Size(mL):     0.500  HPLC Flow Rate(mL/min):
2.00  TRLSC: Normal
Run Length(min):  107.60     LS Flow Rate(mL/min):
2.00  Cell Type: Liquid
Graph PLA2 appears here.
8.00 x 5.00 (in) - Portrait
----------------------------------------
```

```
FLO-ONE\Beta A500  v1.50/-----  S/N:--------  Date:
3-17-93  10:39  Page: 2
Peak Area Report            Channel 1
        Ret. Time        C-14 CTS
Name        (min.)  Pk#   Area  % Tot  Conc.   Units
------------ ------  ---+-------+-----+-------+--------
            1.50    1     32  0.74    0.00
            2.70    2     67  0.54    0.00
            3.90    3     64  0.59    0.00
            5.10    4    129  0.68    0.00
            6.30    5    299  0.72    0.00
FLO-ONE\Beta A500  v1.50/-----  S/N:--------  Date:
3-17-93  10:39  Page: 3
           58.10   50      0  0.73    0.00
           59.10   51      0  0.73    0.00
           60.10   52      0  0.64    0.00
           61.10   53      0  0.69    0.00
           62.10   54      0  0.84    0.00
           63.10   55      0  0.79    0.00

Total Area:              55990

Sum of Adjacent Peaks Report
--- No Adjacent Peaks Summed ---
```

APPENDIX 2
PART OF MAIN CALLING PROGRAM FOR SUNR ILLUSTRATING MACRO CONTROL STRUCTURE

```
*** MAIN PROGRAM TO CALL MACROS FOR
SUNR;
OPTIONS SOURCE2 SYMBOLGEN MPRINT;
*** CALL PROGRAM TO ENTER DATA FROM
ORACLE;
%INCLUDE  "&_datalib.SENTR.SAS";
%macro main;
%let _gpair=0;
%*** CALL ANTAGONIST MODULES;
%if &_agant=ANT %then %do;
  %do i=1 %to %eval(&_pr);
    %include "&_datalib.SANASKDS.SAS";
    %include "&_datalib.SANINIT.SAS";
    %include "&_datalib.SANPARMS.SAS";
    %include "&_datalib.SANORPOL.SAS";
    %include "&_datalib.SAN1_2.SAS";
%if &_tc=YES %then %do;
      %include "&_datalib.SANDR.SAS";
    %end;
    %if &_tc=NO %then %do;
      %include "&_datalib.SANDRNO.SAS";
    %end;
  %end;
  %include "&_datalib.SANGODS.SAS";
  %* pkb conv;
  %if %eval(&_gpair)=1 and %eval(&_pr)=1
  %then %do;
    %include "&_datalib.SANPKB.SAS";
    %include "&_datalib.SANPRTSU.SAS";
    %include "&_datalib.SANPERR.SAS";
    %include "&_datalib.SANPRTSU.SAS";
  %end;
%end;
%*** END ANTAGONIST MODULES;
%mend main;
%main
```

APPENDIX 3
EXAMPLE OF DYNAMIC CODE GENERATION

The following DATA _NULL_ creates statements for PROC NLIN:

```
/*****************************************************
  CREATE PRED STATEMENT OF THE FORM:
  PRED1 = EXP[A+B*DOSE]/1+EXP[A+B*DOSE]
 *****************************************************/
DATA _NULL_;
FILE "&_DATALIB.&_JUNK" NOPRINT;
LENGTH LINE $200;
DO I = 1 TO &_NUM;
  line=compress('pred'||i||'='||'exp(a+(b*ldose'||i||'))'
  ||'/(1+exp(a+(b*ldose'||i||')));');
  put line;
END;
  line='model tresp=';
  put line;
/*******************************
   GENERATE DUMMY STATEMENT OF THE
FORM:

DUM1*(A1*PRED1+A2*PRED2+A3*PRED3+A4*PR
ED4+A5*PRED5+A6*PRED6+A7*PRED7+

A8*PRED8+A9*PRED9+A10*PRED10+A11*PRED1
1+A12*PRED12) +
 *******************************/
 DO I=1 TO &_NUM;
   line=compress('dum'||i||'* (');
   put line;
 DO J=1 TO &_NUM-1;
 ANT=BYTE(I+64);
  line=compress(ant||j||'*pred'||j||'+');
  put line;
 END;
  line=compress(ant||&_num||'*pred'||&_num||')');
 IF I < &_NUM THEN line=compress(line||'+');
ELSE line=compress(line||';');
 put line;
 line='run;';
 put line;
RUN;
```

```
PROC NLIN DATA=TRANS METHOD=DUD
OUTEST=NLINEST CONVERGE =
0.00001;
&_PARM1;
%INCLUDE "&_DATALIB.&_JUNK";
```

Statements created by DATA _NULL_ for PROC NLIN:

```
PRED1=EXP(A+(B*LDOSE1))/(1+EXP(A+(B*LDOS
E1)));
PRED2=EXP(A+(B*LDOSE2))/(1+EXP(A+(B*LDOS
E2)));
PRED3=EXP(A+(B*LDOSE3))/(1+EXP(A+(B*LDOS
E3)));
PRED4=EXP(A+(B*LDOSE4))/(1+EXP(A+(B*LDOS
E4)));
   MODEL TRESP= DUM1*( A1*PRED1+
A2*PRED2+ A3*PRED3+ A4*PRED4)+ DUM2*(
B1*PRED1+ B2*PRED2+ B3*PRED3+
B4*PRED4)+ DUM3*( C1*PRED1+ C2*PRED2+
C3*PRED3+ C4*PRED4)+ DUM4*( D1*PRED1+
D2*PRED2+ D3*PRED3+ D4*PRED4); RUN;
```