DATA MANAGEMENT IN CLINICAL TRIALS USING VM/SP, CMS, AND SAS

David I. Hopp and Robert W. Staley, A. H. Robins Company

SECTION 1. STATEMENT OF THE PROBLEM

This paper addresses the use of SAS/CMS interactive techniques for data management in the area of clinical trials. Emphasis is on the reasons certain broad decisions were made, and their consequences; technical details are discussed in the Appendix.

Within the pharmaceutical industry, "clinical trials" refer to closely monitored tests of prescription and non-prescription drugs in medical practices. These trials are primarily for the purpose of receiving final FDA approval for sale of the drug; not all drugs are marketed, since they may be insufficiently effective or lack patient acceptance. FDA regulations require extensive data collection, and the assurance that company internal records be as nearly identical as possible to the physician-generated source document.

Clinical trials are carried out in accordance with a written protocol that is completed prior to beginning the trial. A report form, herein referred to as the Clinical Report Form (CRF), is designed to gather the required data. CRFs make their way back to the pharmaceutical company (or its designated testing firm) and are subject for review, data entry, and edit.

Volume is conventionally measured in pages received per month; this varies widely depending on a company's development activities, but is usually in the range from 1,000 to several tens of thousands of pages per month. Each study requires a CRF specifically designed for it, although there are usually common elements across studies. Studies may last anywhere from a few days to years. At any time, tens or perhaps hundreds of studies, and consequently CRFs, may be active.

Note that this is not a high-turnover setting, such as order entry or ticket reservations. It does, however, require extensive editing of a very large number of data elements per CRF. It is difficult to characterize this number of data elements, since the variability among CRFs is so great. Some trials require extensive laboratory data, some have a good deal of textual material, etc.; a rough estimate would be several hundred to perhaps a thousand bytes per CRF.

The challenges of data management lay mainly in the following areas: minimizing the length of time between receipt of a completed CRF and availability of data for analysis, assuring accuracy, organizing the data for analysis, and assuring the integrity of the data through the edit phase and while it is quiescent, perhaps over many years. This last area is particularly demanding, since it is sometimes necessary to reanalyze (or verify an analysis of) data that has been unused for years, and which might conform to outdated standards and practices.

Criteria for the success of this effort would have to include response to the above challenges, plus control of costs for personnel, equipment and purchase of any software. In addition, new data entry modes presently under development would have to be fully supported.

SECTION 2. ENVIRONMENTAL CONSIDERATIONS

SAS had been in use for several years, and a firm commitment had been made to continuing its use for statistical analysis, and to whatever extent possible, for data management. It is important to keep in mind, however, that SAS is not a "data-base management system," except in an elementary sense; among other things, it lacks controls for security and integrity, across-library standardization, path strategies, etc.

A second aspect of the environment was that VM/CMS had been in use for a year or so, and had proved to be very convenient and flexible. The power of XEDIT and EXEC2 were recognized; VM's acceptance in the general computing community indicated that it would receive long-term vendor support. CMS/SAS was established, and improvements made in 1981 by SAS made it quite acceptable, even though its prodigious use of resources continues to be a problem.

Several more considerations require comment. Since SAS is not a data-base management system, it would have been possible to have installed a package to be just that. However, those that were reviewed appeared to raise more problems than they solved. Our application is well defined and limited; its natural setting is of a relational type, and there would be no gain in using a data-base manager of any other kind. But such packages at present carry so much overhead that adding this to SAS, SAS/GRAPH, etc., would be counter-productive.

Complementary to this is the consideration that we wanted to keep our staff as small as possible, and devote our time to the improvement of clinical trial data management, not to the care and feeding of complex systems.

A final overall consideration was to avoid what could be called "arcane techniques." Put another way, we decided to avoid special hardware and software that were suited only to the specific task at hand, and that would necessarily restrict our future choices. This arose not merely from principle, but from the decision that we would not be looking for any sort of ultimate, fixed solution, but rather an approach that could evolve to take advantage of an increased understanding of the problem, new capabilities, etc.

SECTION 3. STATEMENT OF THE SOLUTION

Briefly stated, it was decided to write a SAS-oriented data dictionary-driven SAS program writer, supplemented by relatively secure interactive editing procedures; the program writer to be flexible enough to handle CRF data entry from a variety of sources. This was to be a solution designed to fit the problem, leaving room for growth, without overpowering the original problem and replacing it with a new set of even more complex problems. The program writer would use a SAS-program-like CMS file (DATAEDIT SKELETON) as its model for the
entry/edit programs, and merely insert the necessary SAS statements (derived from the data dictionary) in the appropriate places (as specified in DATASET SKELETON).

Table 1 describes the elements in the data dictionary.

The data dictionary is built interactively, which readily allows for the reuse of sections of one CRF that conform to the requirements of another. This results in a significant time savings, since there is often substantial duplication within a CRF, as well as among CRFs.

Control is maintained over the data dictionary by locating it, and the programs that manipulate it, in a "secure" account, i.e., in a CMS virtual machine to which authorized users have READ/ONLY access, and to which only the data base administrator has READ/WRITE access.

Each user, during development of additions to the data dictionary, works in his or her own account, with only the necessary data dictionary records, performs testing and when finished, requests the data base administrator to add the new elements to the main dictionary. All additions, deletions and changes are date/time stamped automatically. Indeed, deletions are actually retained, and simply marked as "deleted." This provides an additional level of integrity in case confusion arises concerning just what data was analyzed at a particular point in time.

There is only one copy of DATASET SKELETON and the programs (mainly EXECs) that use it, similarly located in the secure account. As changes are made, outdated versions are archived.

SAS libraries, including selected data elements from a CRF, are derived from specifications in the data dictionary which are supplied by the statistician. Thus the output libraries are in accordance with the use to which the statistician intends to put them (simple defaults are also available). The library is built in a READ/WRITE account and archived when appropriate.

An important adjunct is the drug dictionary, which is automatically invoked in either editing or analysis, as necessary. This allows drug names to be checked during edit, and drug characteristics (generic components, manufacturers and therapeutic class) to be available for analysis. The drug dictionary contains a synonym file, to allow for common misspellings, and also automatically produces a file of "misses", which is periodically reviewed by the data base administrator to determine what updates should be made to the dictionary.

The last part of the data management process to be discussed here is the edit control facility. When an editor discovers an error (due to data entry, source error, etc.), the input record may be changed interactively. The process appears to be the straightforward use of CMS XEDIT; actually it is carried out through a transparent EXEC using the XEDIT SOS subcommand and various macros. This automatically builds what amounts to a transaction log, recording all changes, etc. for later review if required.

Considerable attention is given to providing a "paper trail" documenting release of libraries for analysis, archiving of libraries and their associated source data and programs, etc. Since we are relying very heavily on machine-readable data, it is not necessary to fully document all important decisions regarding the use and disposition of this data.

SECTION 4. EVALUATION

As it was intended, control and standardization of data entry/edit and overall data organization have been partly accomplished. The use of a data dictionary-driven program-writer has established a setting in which standardization is, so to speak, the default. This has also decreased the amount of repetitive work that has to be done, and allowed non-programmers to successfully carry out tasks previously requiring programmers. Data security and integrity have improved since the temptation, and advantage, of scavenging have been nullified by the easy-to-use interactive processes.

The total amount of time invested has been less than two-thirds man years from start to design to the present; the programs were actually in use within three months from the start, with only the part-time involvement of one programmer. All during the project, only one of the authors has been working on this at any one time, and that as a part-time task.

An additional positive aspect has been that, instead of imposing a new methodology on this problem, we have been able to evolve an approach well suited to the problem, and avoid disruption of on-going work.

In brief, we now get more work done, with fewer errors and greater standardization.

New data entry methods are presently being implemented, and their incorporation into the existing scheme has been quite simple.

On the negative, or at least less than desirable side, are the following. First of all, we are intimately tied to VM/CMS, and a change in operating systems would cause extensive problems. This may very well never become a real problem, since we may adopt other solutions to our problem before changing operating systems. Nevertheless, it must strongly influence any corporate decisions regarding computing.

As has already been mentioned, SAS is not a data base management system, and we will still have to work with all the consequent deficiencies. This can be turned to our advantage, since it actually gives us freedom to install data base management that will "feed" SAS, but it certainly will call for more work. In the meantime, we must contend with SAS's lack of security and integrity protection, and develop procedures that will probably be replaced within a year.

All in all, this combination of VM/CMS and SAS allowed us to rapidly and inexpensively develop and install a flexible, easy to use system that encompasses data entry, edit and organization.
TABLE I. ELEMENTS OF THE DATA DICTIONARY

The following are the elements of the Data Dictionary primarily of use with data entry and edit:

1. Name - SAS variable name
2. $ - Specifies character/numeric variable
3. Array size - Maximum subscript if an array
4. Length - Optional variable LENGTH
5. I/O flags - Show whether variable is for input, output or both
6. Input record # - Input record on which variable is to be found
7. Input format - Exactly as per SAS, except name table is replaced with double-quote
8. Label - LABEL
9. Informat - INFORMAT
10. Proc Format - Reference to PROC FORMAT and VALUES
11. Simple edits - Name or discrete value edits carried out by XEDIT macros
12. Output formats - Format used if I/O flag set to output
13. Relations - Check variable against a table
14. Lookup flag - Check variable against a table
15. Root - Redundancy avoidance
16. Missing - MISSING values
17. %Include - INCLUDE documentation file

APPENDIX A.
EXEC, XEDIT and SAS

The process of using the Data Dictionary to write SAS programs is invoked by executing an EXEC that has three essential parts:

1. &COMMAND SAS BUILDIT
   Execute the SAS program BUILDIT. This passes thru the Data Dictionary (Illus 1) and uses PUT statements to build DDEDIT EXEC containing XEDIT commands, all preceded by &STACK. Some of these lines may be complex, subsequently causing execution of XEDIT macros. They will be applied against DATAEDIT SKELETON (Illus 2).

2. &STACK DDEDIT
   DDEDIT EXEC is executed, which places the XEDIT commands, above, in the stack (Illus 3).

3. XEDIT filename SAS
   The stack lines are applied against DATAEDIT SKELETON and the requested SAS program is built (Illus 4).

Illustration 1 - Data Dictionary

The character in position 1 identifies the record type:
D = Designation record, including project/protocol and variable name
F = Format as per SAS
L = Label as per SAS
E = Edit, used by an XEDIT macro to generate SAS code
V = VALUE for PROC FORMAT
DATA
   CARD01(KEEP=STUDY PATIENT _LIST01
   CARD02(KEEP=STUDY PATIENT _LIST02
   CARD03(KEEP=STUDY PATIENT _LIST03
   *
   *

* AUTOMATIC INSERTION OF ARRAY STATEMENTS HERE.
  *
* AUTOMATIC INSERTION OF LABELS STATEMENT HERE.
  *
* AUTOMATIC INSERTION OF LENGTH STATEMENTS HERE.
  *
* AUTOMATIC INSERTION OF INFORMAT STATEMENT HERE.
  *
* AUTOMATIC INSERTION OF FORMAT STATEMENTS HERE.
  *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *

CARD01:
* AUTOMATIC INSERTION OF INPUT01 STATEMENT HERE.
  *
* AUTOMATIC INSERTION OF MISSING01 STATEMENT HERE.
  *
* AUTOMATIC INSERTION OF EDIT01 STATEMENTS HERE.
  *
* AUTOMATIC INSERTION OF RELATION FOR CARD01 HERE.
  *
OUTPUT CARD01:PUT _INFILE_ RETURN;

Illustration 2 - DATAEDIT SKELETON

A - _LISTnn refers to a SAS macro containing the list of variables
B - This segment simply allows for the insertion of the appropriate statements
C - Each input record edit looks exactly the same
A - Beginning the EXEC, showing a SAS macro

B - Commands causing insertion of ARRAY statements

C - Partial contents of one of the "CARDnn II" sections

D - An INPUT statement, showing SAS formats derived from dictionary

E - Edits derived from dictionary; note the stacking of calls to the XEDIT macros MN and MA, followed by their arguments.

F - PROC FORMAT and VALUE statements

Note: The final entry must be &STACK FILE
Illustration 4 - SAS Program Written via Data Dictionary

A - An INPUT list macro
B - ARRAY statement
C - SAS labels
D - Simple edits generated via automatically invoked XEDIT macros