THE DEVELOPMENT OF USER SUPPORT MANUALS FOR SAS SOFTWARE: SOME GUIDELINES

David L. Masamitsu, Independent Computer Systems Consultant

Introduction:

SAS is one of the best documented and best supported systems of its kind on the market today. Despite this fact, gaps still exist in its documentation -- gaps which the SAS Institute would probably find uneconomical, impractical, or quite impossible to fill.

Examples of such gaps are:

- Job Control Language
- Overviews of the physical computer
- Basic data management techniques
- Basic data processing principles
- Magnetic tape usage
- Commonly made errors in using SAS
- Error detection and diagnosis techniques
- Logon procedures
- Site-specific differences
- Computer system & equipment peculiarities

As a consultant, I have often found that most of the user problems I encounter, have come as a lack of training and user support in these areas -- problems and frustrations that could probably be avoided with a proper system of user support. I have also found that even experienced SAS users, who may know quite a lot about SAS, but little about data processing in general, often come to me with similar problems.

Obviously, the solution is for each computer center to provide a proper system of user support -- including manuals -- which will help to solve these problems.

We live in a world of realities, however, and one of the realities that I have seen is that site data processing staff often have very little interest or motivation to provide such support. Often, computer center staff, themselves, may be novices at using SAS.

An obvious solution to this problem, is for the users to seek such support off-site, outside of their own organization. A hard reality that is often seen here, is that organizations may have rules that forbid data processing service and support that is not "in-house". In cases where such support is not forbidden, users may find it to be expensive, or just not nearby.

In my opinion, the problems that I have just mentioned are quite common in both the public and private sectors. It is my opinion that without a proper system of user support, even a system as well documented as SAS may turn into a nightmare of needless user frustration -- resulting in software that is never used to its fullest potential, users that are rightfully disappointed and dissatisfied, and organizational needs for analysis that remain unmet.

One of the complaints that I have heard from users is that while SAS can be easy to learn, gaps in its documentation and user support often combine to make it difficult to use. The point that I am making is a simple one:

When software is made easier to use, odds are that it will be used.

This paper draws upon my recent experiences in developing a user support manual for a state agency, and offers what I consider to be some important guidelines to consider for anyone else who may attempt to develop user support manuals for SAS software.

Overview of the Problem:

For 13 months during 1983 and 1984, I was on contract to the Arkansas State Department of Health.

In my opinion, many of the problems that I have mentioned were present at that time in the Arkansas Department of Health.

Additionally, there was another problem: A physical lack of hardware. At that time, the Arkansas Department of Health was in the process of replacing its hardware.

Since user needs were at least temporarily not being met, I saw the solution in obtaining access to a modern piece of hardware with SAS, and aggressively training users to do some of their own computer analysis with SAS.

Access was obtained to an IBM 4341 in another state agency. Its version of SAS was then upgraded from SAS 79.5 to SAS 82.4, and both CMS:SAS and OS SAS were purchased. Additionally, a SAS BASICS course was purchased from the SAS Institute. This was a CMS SAS course.

The hardware at the site, however, was only configured to support magnetic tape use under the OS/MVS operating system; therefore, CMS SAS was only able to analyze data on CMS minidisks that the user had entered by hand, at a terminal.

While this was perfectly appropriate for many of the other state agencies who had people attending the CMS SAS BASICS seminar; it was not appropriate for Health Department users, who had to analyze many large databases, imported via magnetic tapes.

When this was realized, I began on my own initiative to develop a formal supplementary course, a text, and a user support manual that would allow newly trained CMS SAS users to use both CMS SAS and OS SAS. The idea was to bring them from a novice level of understanding about data processing, to a more intermediate level of sophistication.

It proved impossible to schedule any formal
course, due to schedule conflicts and other logistical problems. This meant that the manual itself, would have to be made into a self-taught user course.

Since I consider it to be unwise to train people to use sophisticated software, and then fail to provide knowledgeable user assistance; this meant that the manual also had to be a general user reference for problems.

It had to be both a course, and a reference, and therefore, much more comprehensive with respect to "filling the gaps" that I have spoken about.

Manual development began on April 1, 1984. By April 22nd, the first draft was done. Twelve drafts later, on July 20th, the final version of the manual had been approved, pilot-tested, critiqued by three out-of-state experts, reviewed by 15 state agency personnel, copied and distributed.

The manual was titled:


The manual seems to have more than accomplished its goals. Comments were extremely positive. The Arkansas Department of Health continues to use SAS, and is interested in expanding SAS training with other SAS courses. The Arkansas Department of Computer Services, which has the IBM 4341 that certain divisions and bureaus of the Health Department still use; has adopted the manual as its standard SAS user support manual, and has even sent some of its own staff to advanced SAS seminars. Several other state agencies have also trained SAS users. Most importantly, thanks to the cooperative efforts of many people, many of the previous problems are being addressed and solved. A much more supportive user environment has appeared.

Guidelines for Manual Development:

If these results can be termed "success", what can be learned from all of this? Obviously, the final product was by nature, somewhat site-specific. Does it have any value beyond the state of Arkansas?

I think so.

While half of the manual may only be useful to you people as a model or pattern; the 13 basic guidelines that I followed may be of value to you, if you would like to try to develop a similar manual.

1. Assume no previous knowledge by the reader.

I assumed that the reader knew absolutely nothing about computers, programming, or data processing. The only assumption that I did make, was that the user knew how to write SAS programs. I did not try to teach the user how to write SAS programs, as that was the job of a formal SAS BASICS course. I did, however, try to teach users how to write better SAS programs.

The single group of users with the greatest needs for user support, are those who are novices. In this case, they are those people who had been taught how to use SAS, but still lacked much "hands-on" SAS and computer experience.

2. Target the audience of your manual.

This manual was for "health professionals". It was not intended to be for the data processing staff in the Health Department, but instead, for the data processing users. It was designed to help users to help themselves, and to make it easier for them to do their own analysis.

Targeting your audience helps you to write a manual that uses meaningful analogies, and to facilitate learning and understanding in your readers.

3. Use meaningful analogies, whenever possible, to explain technical terms and subjects.

Prior to going to Arkansas, I had written other manuals, and I had some experience in teaching university students. I believe that using a meaningful analogy allows someone to almost instantly understand a complicated and highly technical subject.

In my manual, I likened the computer to an octopus, in order to explain how the various parts of the computer function, and how they relate to the function of the whole computer.

Another analogy that I used was to compare IBM JCL to addresses on an envelope being mailed. I also likened an assembly language program to putting a piece of moulding around a crown.

These were meaningful to my audience. People who work in an office can be expected to know what a filing cabinet is. A health professional has probably taken at least some basic biology -- enough to know what an octopus looks like, and what organs are.

4. Provide an overview of the physical computer and an overview of how software makes it work.

Showing how a computer works is a critically important conceptual step to take, if you want to show users how to use computers efficiently.

Yarmish and Yarmish, in their book Assembly Language Fundamentals 360/370 OS/VS DFS/VSE, mirror my sentiments when they state that in their experience, assembly language programmers become better programmers in other languages, because they are forced to have a deeper understanding of the workings of a computer.

5. Provide an overview of OS JCL and provide OS JCL set-ups for the user.
5. (continued)

If you're using SAS on an IBM mainframe, user knowledge of IBM's Job Control Language is probably going to be essential.

In Arkansas, macros and execs were provided for users, which automatically inserted the user's JOB CARDS for them in any submitted batch job. This meant that I only had to explain a small subset of IBM OS JCL in my manual.

This subset was further limited by the fact that I had a pretty good idea of the kinds of JCL applications that SAS users would have.

I explained JCL, and provided JCL set-ups -- "fill-in-the-blank" type JCL examples -- which showed the users exactly what their JCL should look like if they wanted to input data from a tape file or from OS formatted disks. More importantly, I showed what the user had to know to fill in the blanks on the set-ups.

JCL "word problems" were also provided, as well as examples of JCL errors. These word problems and error examples showed users how to "dissect" IBM JCL, and locate common JCL errors, as well as how to "analyze" what type of JCL they needed to make their SAS programs function. The error examples contained explanations of what went wrong, why, and how the error could be corrected.

Additionally, I provided the users with an annotated bibliography of readings and textbooks -- which included some excellent JCL textbooks for novice and intermediate level users.

6. As much as possible, make each section or chapter of the manual a complete section and a complete course or reference, by itself.

My manual was 310 pages long. It is my own experience that people will seldom read a large manual cover-to-cover.

Again, my purpose was to create a textbook and a reference. To be a good reference, each chapter should "stand by itself". It should keep references to material in other chapters to a bare minimum, and should be purposely redundant, with respect to the entire book.

Some material will overlap with other chapters. As such, redundancy becomes necessary if your manual is to be successful in showing users how material interrelates. Redundancy, however, is a double-edged sword -- it can bore readers. In order to minimize this, I purposely wrote redundant material in a slightly different way, each time.

7. Use examples, diagrams and pictures along with your written explanations.

Another problem that I had, was that there were three different methods of computer access, and five different types of terminals.

Nothing is more frustrating to a novice than sitting down at a terminal that has a keyboard that is different than the one that he or she first trained on.

My manual outlined all logon, logoff and security check procedures for every kind of access, using all possible types of terminals. This feature took up half of the manual's 310 pages.

I have found that the following format is best for teaching users how to logon and logoff:

A. Each page should cover only 1 step of the logon/logoff process.

B. At the top of each page should be a diagram of exactly what the user's screen should look like; plus, clearly identified, the commands, etc. to be input.

C. In the center of the page should be a diagram of the terminal's keyboard; with arrows pointing to the ENTER key, RETURN key, or whatever key will be used in this step of the process.

D. At the bottom of the page, in simple words, should be a short text that tells the user what to type, and if appropriate, why you type it into the computer.

E. If errors can occur in the process, there should be diagrams of each problem, and explanations of what went wrong and how to correct it.

These diagrams should be in the correct order that they will be encountered in the process.

One of our access methods was via 3270 terminal emulation. As such, it was very important to include special diagrams of error messages that the user might see if emulator failure occurred.

If any of you have read the IBM VM/SP: CMS Primer, the methods that are described above, are essentially an improvement over the format of that manual. The main improvements are that a diagram of the keyboard is provided, and that each step of the process is covered on a separate page.

8. A section on common SAS errors and common problems should be included in the manual.

Everybody makes mistakes. New users are likely to be driven into frustrated rages by mistakes. It has been jokingly said that the one common language that all programmers know, is obscenity.

One of my biggest gripes with system documentation is that it is seldom designed to assist a user in diagnosing and locating even the most common errors and problems.
It is not enough to simply show a user what to do. Users have to be taught techniques, skills and routines that will enable them to identify and solve at least some of the most common problems that they are likely to encounter.

In this respect, a user support manual can do what no other manual does: it can pull together skills, techniques and information from a great variety of sources, and present them to the user in a compact, general problem reference guide. When information that is normally found only by using many manuals, can be found in a single manual; users have a much easier time when "something goes wrong" with a program. Computer center staff time is also freed for users with more complex support problems.

Six chapters of my manual were oriented towards "what do you do if something goes wrong?".

I provided users with a chapter on OS Return (completion or system) Codes, with a detailed description of what went wrong, and how to find and correct the problem. I also provided my users with a chapter on how to use the SAS log and SAS listing to locate errors. Additionally, there was a chapter on "how to more certain that SAS gave you what you really wanted."

9. Manual chapters can be grouped into basic types, but it is not always best to arrange your manual that way.

Conceptually, individual chapters of the manual can be thought of as falling into one of three basic types of chapters:

A. Learning Chapters: which teach the user introductory, general DP concepts.
   -- overviews of the physical computer
   -- overview of how software makes it work
   -- basic data management techniques

B. Technique and Assistance Chapters: which show the user how to gain access, teach specific DP subjects, and act as a user reference in case problems occur.
   -- logon/logoff procedures chapters
   -- OS SAS and OS JCL chapters
   -- Return Codes & basic error detection
   -- Complex error diagnosis via SAS logs and SAS listings

C. Useful Information Chapters:
   -- site computer use policies, and site restrictions on computer use
   -- Important information and forms to use
   -- Glossary of DP terms

Useful Information Chapters are important for sophisticated users, as well as novices. If your computer center isn't providing users with this information, in my opinion, it indicates a very low level of user support. As such, you should consider including it in your manual.

Some examples of important information are:

Names and phone numbers of DP personnel.
Your SAS site number, a description of the computer, operating system, and version of SAS being supported.
Names of SAS site consultants.
The address and phone number of the SAS Institute.
Forms to help users keep track of what is on their disks and magnetic tapes.

Again, these are good things to include; but think carefully about how you arrange the chapters of your manual.

For example, although the chapter on data management may be a Learning Chapter; I purposely made it the 9th chapter of my manual. In this way, the material would be located after all of the chapters that teach the user about hardware and software, and show the user how to begin to use the computer. Users would first learn how to use the computer, and then learn how to use it efficiently.

10. Place your manual into a 3-ring notebook and design it to be easily updated.

A notebook is more easily used at a terminal than a bound book is, as a notebook allows users to flip pages and not have to worry about losing their place.

Placing your manual into a notebook also allows it to be more easily updated. Pages can be removed and inserted easily. Odds are that at least some pages of an old manual will still be useful. Using a notebook allows you to simply issue a packet of new insert pages to replace the obsolete pages. This will keep you from having to issue a completely new manual to each user, when changes need to be made.

Designing your manual to be easily updated is very important. If your manual contains references like: "see David Masamitsu", what happens if David quits, gets promoted, or if procedures are changed that make it someone else's duties? Such changes could force a major revision of your manual!

Make your manual less vulnerable to such changes by using JOB TITLES, instead of specific names; and by placing the following notation at the bottom of each page where a job title and phone number occur:

* /#/

This notation allows users to easily update their manuals by writing out or placing file folder stickers over the job titles and phone numbers, and providing the users with a space to write in the job title and phone number of the new person in charge. If job titles and phone numbers are changed again, a new blank sticker can be placed in the space, obscuring the previous change.
I have used this technique in other manuals that I have created. It is tempting to use specific names, but I have found that names are best limited to your Useful Information Chapters.


The acid test of a manual is whether or not users can actually demonstrate that they understood the material. The only way that you can be sure of this, is to conduct pilot testing before the final version of the manual is completed.

Ask your test personnel for specific suggestions as to how a manual can be improved or recorded. It is not enough to have them simply comment on whether the manual is good or bad.

12. Teach basic data management techniques.

I consider this to be an absolute necessity for proper user support; yet it is one that is very often completely ignored.

It is criminally stupid to teach someone to use a system, and fail to teach them anything about how to use it efficiently!

I have seen many academicians who preach that users should be responsible for their own data, and I have seen many computer centers with staff that constantly complain about how "aggravating, stupid and irresponsible" their users are. In nearly all of these instances, these academicians and DP staff failed to recognize that these problems can be largely avoided by simply teaching users basic data management techniques.

Much of the basic core of data management is very easily understood and easily taught:

- Flowcharting.
- Documenting files and programs.
- Keeping paper records of what is on a disk or magnetic tape.
- Learning to periodically back-up/delete files and programs on disk.
- Writing code books/decoders for raw data files and databases.
- Calculating the number of tracks in use.
- Calculating the length of a file on a magnetic tape.
- User tape librarianship.
- Learning to use tape dump utilities.

My own experience has shown that these things can be taught in an understandable and very meaningful way, even to novices.

My own preference is to provide users with forms to record such information, and teach them work routines that basically say:

"Here is a form that you can use to keep track of what is on your magnetic tapes. This is why you need to fill them out, and here is an example of a properly completed form..."

It is also a good idea, particularly with respect to flowcharts, to show users examples of "bad" flowcharts and poorly filled out forms.

13. Form a user's group, if possible.

There was no time to do this, when I was in Arkansas; however at the time I left, there was an expressed desire to do so.

A proper system of user support goes beyond a manual. If support from your computer center staff is slow in coming, users must band together to help themselves.

A user's group is a fine way to do this. It can allow you to do some "classroom" type training. It can give you a forum for generating new ways to improve your manual. Moreover, a user's group forms a basis of support that is both local, and comprised of people who have an active interest in using SAS. In some cases, it may be the only avenue for users to get assistance with problems.

I have seen sites that openly consider all novice users to be "unworthy" of having any access, and "overly troublesome" to support. I have also seen sites where computer center staff consider the support of anything other than COBOL or a higher level language to be "a waste of time and system resources".

Bad data processing professionals like these can be found everywhere. Often they hide under the euphemism of "traditional data processing". This is a euphemism for incompetence, selfishness, narrow-mindedness, and foochardy staff turf protectionism. It only serves to prevent an organization from ever accomplishing its own objectives.

With such people, users will simply have to do the best they can to fend for themselves.

Conclusions:

The development of user support manuals and user support systems can be very time consuming, but it is certainly worth the time and effort.

Even in situations where your hands are very strongly tied, and where your other efforts are completely negated; a user support manual can still do a lot.

A properly developed user support manual can form the cornerstone of such support systems, and it can open both eyes and doors on a manual by itself is only a piece of the solution.

Aggressive user training is very important. The formation of user's groups are also very important, as is the provision of knowledgeable and experienced user assistance.

Most importantly though, DP professionals are going to have to learn to develop proper attitudes:

We're going to have to come down off of our high...
horses, and remember how it was when we were novices.

True professionals realize that data processing exists to serve its users, and that it is their job to be reactive to user needs, to be helpful in identifying problems, to help solve problems, and to provide a supportive total environment for the users, that emphasizes user training.

Data processing is an integral component of proper project and organizational planning and management; it is not "just" a support service.

Sophisticated users cannot always be simply hired. It is often best to provide an environment that allows users to become sophisticated.

If we aren't more sympathetic, and if we don't do a better job at providing answers, instead of setting up barriers; we're going to be out of a job.

The gaps that I spoke of, can be filled, must be filled, and it is certainly in the best interests of all concerned to fill them quickly.

FOOTNOTES

   (page iii of book is paraphrased)

2. International Business Machines Company. IBM MVS/370 System Product: CMS Primer IBM Form Number SC 24-5236 File Number S370/4300-39
   (Release 3 of this book is referred to)

The author may be contacted via mail at:

David L. Masamitsu, MSPH.
Systems Consultant
P.O. Box 7704
Mission Hills, CA 91340-7704

* SAS is the registered trademark of the SAS Institute, Inc. Cary, NC., USA.

IBM is the registered trademark of the International Business Machines Corporation.

CMS, OS/MVS, and OS JCL are registered trademarks of the International Business Machines Corporation.