Making SAS® Version 5 Work for You: The Design of a Dynamic Reporting System for All Users

Katie Hubbell, ORI, Inc.

Abstract

Several Version 5 features of the SAS system have been combined to demonstrate one way to create a generalized, easily maintainable user interface without sacrificing flexibility and response to user growth. A reporting system is used as an example. The steps taken in designing such a system are outlined, and the use of SAS/AF*, SAS/FSP*, the Display Manager System (OMS), SAS macros including command-style macros, and AUTOCALL are highlighted.

Introduction

Traditionally, when writing a user interface system, data processing professionals have had to choose between flexibility and generalizability. The popular choice would seem to be to make the system as generalized as possible, thus cutting down on code and simplifying maintenance. However, making a system completely generalized dramatically reduces the amount of flexibility presented to the user.

If a system is too generalized, it can frustrate users by only allowing them to operate in one way. For example, a menu-driven system which is written in modular, generalized form would have to gear itself toward the least experienced of its potential users in order to be usable by all. However, users who are more advanced or have used the system for a while and are very familiar with it will soon tire of having to go through the same steps over and over again and will want a shortcut. Of course, a shortcut can then be written, but this is not always feasible; why not include the shortcut in the design of the system? Predicting and allowing for user growth and varied user capabilities as well as changing user demands is essential in creating a frustration-free interface system that will be utilized to its fullest extent.

On the other hand, the maintenance side of things must also be taken into account. The system must be relatively easy to update and modify, since change is a certainty if the system is to stand up over time. Modularity is a generally accepted way of easing maintenance problems.

How, then, to create a system that strikes a balance between these two goals? Is it possible to make both the users and the programmers happy?

Illustrated in this paper are several features of the SAS system which can be used to effectively make an interface system both flexible and generalized. A reporting system recently designed for a nationwide survey will serve as an example.

Structuring the System

The dilemma of generalizability vs. flexibility really comes into play on two levels: the inner workings of the system, and the system presentation. The system presentation will be dealt with first.

Before any design was attempted, extensive research was conducted to determine: the variety of users to be targeted; the current needs of those users; and how those needs were likely to change as time went on.

It was found that the range of users extended from non-computer users all the way to advanced users already skilled in SAS. The volume and variety of reports needed by these users was quite extensive, and reports were likely to be added to the system periodically as well.

Obviously, with this wide variation in user skills, a simple generalized menu system prompting users for each parameter would not do. Similarly, the actual reports could not be merely hard-coded standard reports, since not only were there too many reports for hard-coding to be feasible, but the advanced users were certain to want to vary the reports in different ways.

It was decided that the structure of the basic system would have to allow three options: an option for novice users which would allow them to select and run a "standard" report, using simple menu prompts; a more advanced option geared toward users with a little more experience, which allowed them, still using prompts, to modify a standard report in certain controlled ways, such as subsetting the data or changing the way the data was grouped; and an option allowing advanced SAS users access to the Display Manager System (OMS) with all of the reporting system's macro tools for report formatting and compilation at their disposal. This structure not only allows for the current variation in user skills, but allows for user growth. It was anticipated that as novice user worked with the first option and became more
familiar with it, he might then feel comfortable turning to the second option, and so on. In this way, freedom of use is maximized and frustration largely eliminated from the user's standpoint.

The "Inner Workings"

The users' requests have now been addressed. It only remains to implement the system with as little programmer frustration as possible. Each of the following techniques was used with the purpose of making this extremely flexible system modular and easily maintained.

A Well-Kept Macro Library is Essential

The largest part of this system consisted of generalized macro modules. Macros were written for extracting the correct data from the data base, summarizing or otherwise manipulating the data, and formatting the data into reports.

All macros were stored in one large central macro library. The SASAUTOS = option was used, referencing this library, so that all generalized macros were available to the system with a macro call instead of INClUDE statements which could get quite large and would have to be modified each time a module was added to the system.

In addition, each of these modules was well-documented as to what other input was needed and what other modules it could be used with, thus enabling the advanced users to employ the modules to do most of their work for them when working in OMS.

Using SAS to Store Changeable Parameters

Each report in the system had certain parameters that made it a unique report, such as the column and row labels and the specific variables needed from the data base to produce that report. These parameters were stored in what came to be known as a SAS parameter library. This SAS data library maintained a SAS data set for each type of report-specific information. Identifying variables such as report category and number made each observation in the SAS data set unique to a certain report. When a report was chosen, the system subset each SAS parameter file on those identifying variables to obtain the correct information (Figure 1).

An added advantage of storing this information in SAS data sets is that PROC FSEDIT can be used to easily modify and add to these files. Rather than having to maintain and update hard code for each report, SAS parameter files were used whenever possible.

![Figure 1](image-url)

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>REPNUM</th>
<th>VARNAME</th>
<th>VARLIST</th>
</tr>
</thead>
</table>
| 1        | 1      | variable list
| 1        | 2      | variable list
| 1        | 3      | variable list
| 2        | 1      | variable list
| 2        | 2      | variable list

Keeping a Tight Rein on Hard Code

As in any generalized system, hard code was avoided at all costs; however, it was still used in this system for the simple reason that there were some cases that could not be taken care of with generalized modules. Some reports had special processing not found anywhere else; some users requested reports differently than the rest. Rather than sacrifice flexibility by keeping everything within the strict outlines of the modular system, it was decided to allow one "module" of hard code for each report, in case special processing was required. These modules had names based on the report category and number, and were called just like the other macro modules while the report was being processed.

These modules had to be well-maintained, well-documented, as small as possible (anything that could be generalized, was), and as free of references to parts of the system that might change as possible. They were written in a given format; since many types of specialized code (such as custom-tailoring the report title to the user's specifications) came up often, this code was always put in the same place in the module. As more reports were written, patterns began to emerge which, while they could not be generalized, could aid maintenance by being recognized as patterns.

SAS/AF - A New Twist

The foundation of this system is the menu structure, which is written in SAS/AF with an approach that differs from the "traditional" AF system. The goal here was to create as few actual screens as possible to make maintenance easier. Screens that performed the same basic function were placed in groups, and one screen was designed to do the work of all of them.

To do this, the twist was added. Rather than write a MENU screen for each separate menu, one PROGRAM was used for several similar menus. The advantage of PROGRAM screens is that any field on the screen can be linked to an associated SAS macro variable, and therefore can change from one display of the screen to the next. So, a PROGRAM screen like
the one in Figure 2 can be used for any number of choice lists, as long as the underlying macro variables are changed. Let us say that in the ATTR portion of the screen the macro variables &FIELD1 through &FIELD5 have been defined as the associated macro variables for the list, and the variable &CHOICE for the spot in which the user enters the number of his selection. The following statements will initialize the “menu”:

```
%GLOBAL FIELD1 FIELD2 FIELD3 FIELD4 FIELD5 CHOICE;
%LET FIELD1 = 1. Execute a Standard Report;
%LET FIELD2 = 2. Modify a Standard Report;
%LET FIELD3 = 3. Write Your Own Report;
PROC DISPLAY C = screen name;
RUN;
```

The Screen then looks like Figure 3 when displayed. The user makes a selection - 1, for example - and presses PF3. The value is then stored in &CHOICE, ready to be used in further processing. Then, by simply changing the values of the macro variables, the same AF screen can generate a completely differently menu:

```
%LET CHOICE =;
%LET FIELD1 = 1. Report Number One;
%LET FIELD2 = 2. Report Number Two;
%LET FIELD3 = 3. Report Number Three;
%LET FIELD4 = 4. Report Number Four;
%LET FIELD5 = 5. Report Number Five;
PROC DISPLAY C = screen name;
RUN;
```

Figure 4 shows the result.

In addition, a macro was written which determined the number of filled fields in each list and “nulled” out the remaining fields by placing an unprintable hexadecimal character into the appropriate macro variables, so that blank lines would not appear on the screen.

In this way, one AF PROGRAM screen does the work of many menu screens and cuts down on maintenance. Additionally, the SAS parameter files spoken of earlier supplied the lists for each menu, so the entire process could be generalized.

Macro-izing DMS - Adding Specific Help for Your Interface

Even the most advanced users will appreciate extra help while in Display Manager. Command-style macros were added to the macro library to enable the user to obtain needed information quickly. For example, a command-style macro called %SHOW was written. It has one list parameter which can be either "FILEREFs" or "MACROS". When the user enters "SHOW FILEREFs" on the command line, a system command is invoked to list all FILEREFs currently allocated. Using this command, a user can easily find out the FILEREF to be used for a certain part of the system, or ensure that a new FILEREF will not replace one that is already there. If the user enters "SHOW MACROS", a set
of PUT statements is generated to the log explaining in brief the names of the macros available to the system and their functions.

An additional set of macros was added to help system maintenance personnel add reports to the system. The command "ADD REPORT" calls a command-style macro which starts the process. Next, FSEDIT is used to prompt for each parameter to be appended to the SAS parameter files in order to add the new report. The process concludes with reminders and information about the hard-coded macro to be written and where it must be stored.

Conclusion

The SAS system provides a unique opportunity to create flexible systems that are also generalized. With a little imagination, and the advances of Version 5, a powerful user tool can be created which will grow as its users grow.

* SAS, SAS/AF, and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA.

The author may be contacted at:

ORI, Inc.
601 Indiana Avenue
Suite 1000
Washington, D.C. 20004
(202) 737-2666