USING SAS/AF® SOFTWARE TO DEVELOP PRE-CLINICAL INFORMATION SYSTEMS

Kerry G. Bemis, Eli Lilly and Company

Pre-clinical information systems are characteristically interactive with small to medium size databases. SAS/AF® software is therefore the ideal tool for developing these systems. While the development of these user friendly systems is becoming increasingly popular so is the emphasis on system validation. Although the exact meaning of 'system validation' isn't clear, it is clear that documentation is an important part of it. The objective of this paper is to describe a method of automating the documentation of a SAS/AF program.

SAS/AF is an extension of SAS® that consists of a collection of objects (also called screens). There are five object types which are described in the table below:

SAS/AF® Object Types:

- ***Menu*** - interactive branching
- ***Program*** - execute SAS® code
- ***Cbt*** - question and answer tutorial
- ***Help*** - user information
- ***List*** - values for Program Object user fields

The SAS® procedure CATOUT provides a minimal level of documentation for the collection of objects that make up your program. For each object in your program CATOUT provides the following information:

PROC CATOUT Documentation for each Object:

- ***Name*** - valid SAS® name
- ***Type*** - menu, program, cbt, help, list
- ***Description*** - 40 character field
- ***Date*** - date last modified

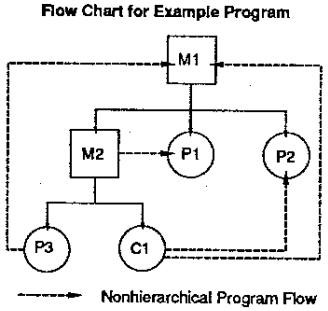
This information can be directed to a flat file or captured in a SAS data set. Let's illustrate this documentation with an example program:

PROC CATOUT Output - Example Program:

Name	Type	Description	Date
C1	cbt	tutorial	01feb88
M1	menu	main menu	02feb88
M2	menu	second menu	03feb88
P1	program	first program	04feb88
P2	program	second program	05feb88
P3	program	third program	06feb88

Program Flow is Not Clarified by This Documentation !

The critical information that's missing in the CATOUT output is the relationship among the objects, i.e. the program flow or structure. This needs to be clarified in an external document or flow chart as illustrated below:



This flow chart is relatively simple, but you can imagine how complex it would get for a SAS/AF program of more than a hundred objects. If you have to modify your program that means you have to modify the flow chart. It becomes clear why good documentation is seldom implemented or rarely updated. It is especially difficult to maintain a program that utilizes nonhierarchical program flow indicated by the dashed lines. I propose eliminating the nonhierarchical program flow and adopt the following strategy:

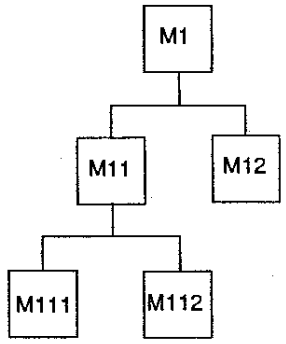
Documentation Strategy using PROC CATOUT:

1. Define program flow with tree structure
2. Let Object Name mirror structure
3. Capture Object Name in data set
4. Generate structure using Object Name (via SAS® program)

Program Flow = Tree Structure = Object Name

The hierarchical structure, i.e. tree structure begins with the menu objects which provide the backbone of our SAS/AF program. To implement our naming scheme we begin each name with the initial letter of the object type, i.e. all menu objects have names beginning with 'M'. The name of a menu object then concludes with a numerical sequence indicating its exact position in the tree structure. This is illustrated in the following figure:

Naming Menu Objects in a Tree Structure



Program flow is unambiguous !

Program flow is allowed only along the lines connecting the menus. We can now capture the menu names in a SAS data set using CATOUT and in a separate SAS program extract the numerical sequence from the name, call it 'menu level' and print the following documentation:

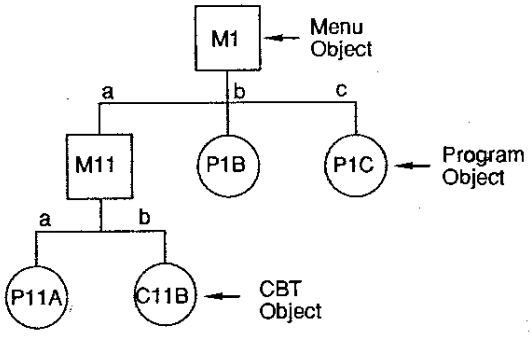
Automated Documentation (menus only):

Menu Level	Type	Name	Description	Date
1	menu	M1	main menu	01mar88
11	menu	M11	report menu	02mar88
111	menu	M111	compound menu	03mar88
112	menu	M112	study menu	04mar88
12	menu	M12	graphics menu	05mar88

Object Name Provides Character String for Menu Level !

Notice that the menu levels are indented in a way that corresponds to the menu location in the tree structure. This tabular documentation thus provides a visual description of the relationship among the objects, i.e. program flow. Now let's extend the naming scheme to objects called as menu options. We will do this with our original example program with the nonhierarchical flow eliminated.

Naming Menu Options (program and cbt objects)



The menu options are lettered sequentially 'a,b,...,z'. Each object called by the menu (except another menu) is named as follows:

1. first letter of object type
2. menu level
3. option letter

This naming scheme of course limits the number of menu options to 26. If you have more than 26 items you need to add additional menus. We can now extract the option letter from the object name and provide a new column in our automated documentation for 'option'.

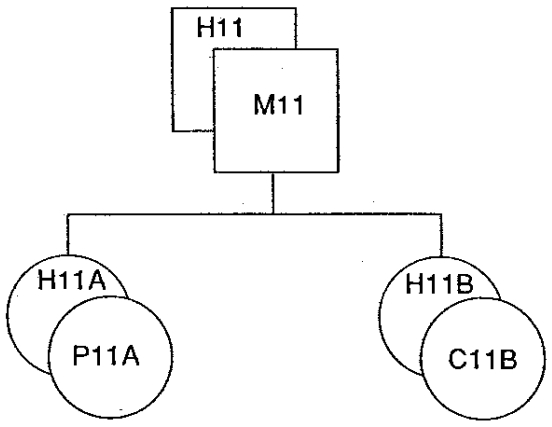
Automated Documentation (Example Program):

Menu Level	Option	Type	Name	Description	Date
1		menu	M1	main menu	02feb88
	b	program	P1B	first program	04feb88
	c	program	P1C	second program	05feb88
11		menu	M11	second menu	03feb88
	a	program	P11A	third program	06feb88
	b	cbt	C11B	tutorial	01feb88

Object Name Provides Character String For Option !

If you compare this documentation with the original CATOUT output provided for this example program you can see that the program structure is now clarified without the need of an external flowchart. The objects we have been describing so far will be referred to as primary objects. We now need to discuss attached objects that provide additional function to the program but don't effect the program flow. Menu, program and CBT objects can all reference a help object via the PF1 or HELP key. This help object will be called an attached object. The following figure illustrates menu level 11 with attached help objects. Notice that the attached help objects are shown offset from the primary object. Although the lines showing program flow connect at the attached help objects the control is really given to the primary object on the top of the stack. The connecting lines are indicating flow among stacks of objects with the primary object always maintaining control.

Help Objects Attached to Other Objects via the HELP Key (PF1)



The name of the attached help object is identical to the name of the primary object except for the first letter which is 'H'. We will now list these attached objects immediately following their primary object in the automated documentation:

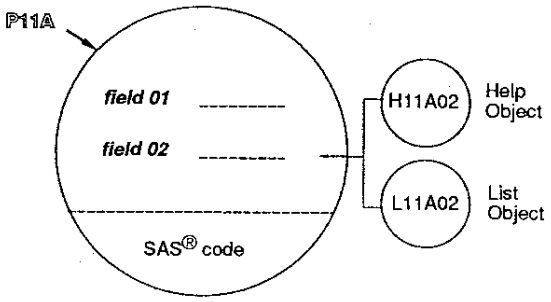
Automated Documentation (level 11):

Menu Level	Option	Type	Name
11		menu	M11
		help	**H11**
	a	program	P11A
		help	**H11A**
	b	cbt	C11B
		help	**H11B**

Help Objects Attached to Menu, Program and Cbt Objects !

Objects can be attached in other ways, for example help and list objects can be attached to user fields in program objects. A program object or screen is divided into two sections separated by a dashed line. The section below the dashed line holds the SAS code and the section above the dashed line has optional user fields where supplied information can be passed along to the SAS code. User fields are numbered sequentially 01-99 for our purposes. Consider program object P11A from our example program:

Naming Objects Attached to a Program Object

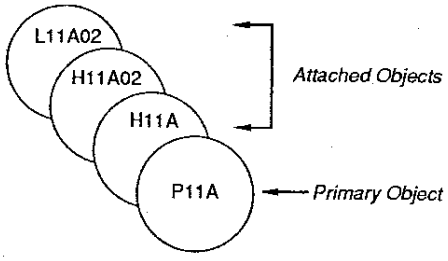


Notice that field 02 has a help and list object attached to it. The help object can be called by the user by putting the cursor on field 02 and pressing the HELP or PF1 key. The user field will only accept values listed in the attached list object. The name of each object follows the following scheme:

1. first character of object type
2. menu level
3. option letter of primary object
4. two digit field number (01, 02,..., 99)

The next figure is a visualization of the primary object and its attached objects:

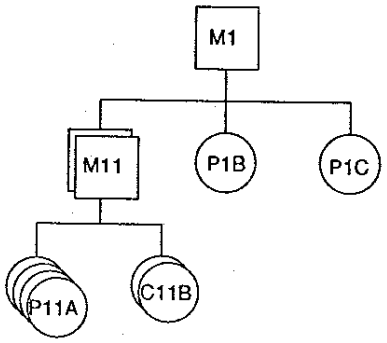
Objects Attached to a Program Object



Primary Object Maintains Control !

Think of this as a 'stack' of objects with the primary object sitting on top and maintaining control. When a program flow chart or table indicates a transfer to this stack it is always the primary object that takes initial control. The next flow chart shows our modified example program with transfer of control between stacks of objects:

Summary Program Flow with Attached Objects



The stack of objects associated with option 'a' of menu level '11' will be listed together in our automated documentation. We also extract the field number from the object name and display it in a new column labeled 'field' as illustrated below:

Automated Documentation (level 11):

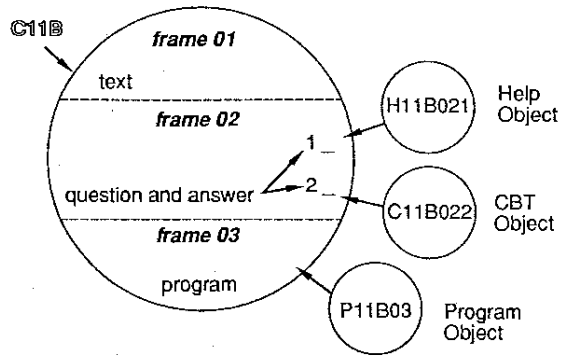
Menu Level	Option	Field	Type	Name
11			menu	M11
			help	H11
	a		program	P11A
			help	H11A
		02	**help**	**H11A02**
		02	**list**	**L11A02**
	b		cbt	C11B
			help	H11B

Object Name Provides Character String for Field !

Objects can also be attached to CBT objects. A CBT object can be visualized as a sequence of frames numbered sequentially (01,02,...,99). There

are three fundamental frame types: text, question and answer, and program. These three frame types are shown below in a diagram of object C11B from our example program:

Objects Attached to a Cbt Object

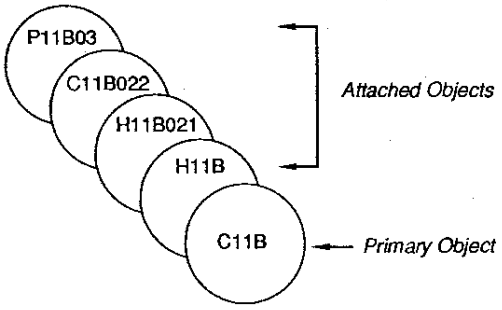


The naming scheme for objects attached to a CBT object is as follows:

1. first letter of object type
2. menu level
3. option letter of primary object
4. two digit frame number (01,02,...,99)
5. one digit answer number if needed

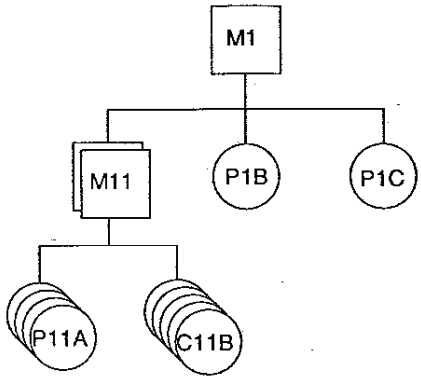
In our illustration above we have a program object attached to frame 03 and a help and CBT object attached to frame 02, answers 1 and 2 respectively. These attached objects are always required to return control to the primary object. Therefore we can again visualize a stack of objects with the primary object on top of the stack maintaining control:

Objects Attached to a Cbt Object



An overall flow diagram of our example program showing the flow among stacks of objects is shown in the figure below:

Summary Program Flow with Attached Objects



The automated documentation will now show these attached objects to C11B listed at the bottom of the stack. The frame number is extracted from the attached object's name and put in the same column as field number with a modified column heading. If a CBT frame is of the question and answer type then the answer number is extracted from the attached object's name and put in a column labeled 'answer'.

Automated Documentation (level 11):

Menu Level	Option	Field/ Frame	Answer	Type	Name
11				menu	M11
				help	H11
	a			program	P11A
				help	H11A
		02		help	H11A02
		02		list	L11A02
	b			cbt	C11B
				help	H11B
		02	**1**	**help**	**H11B021**
		02	**2**	**cbt**	**C11B022**
		03		**program**	**P11B03**

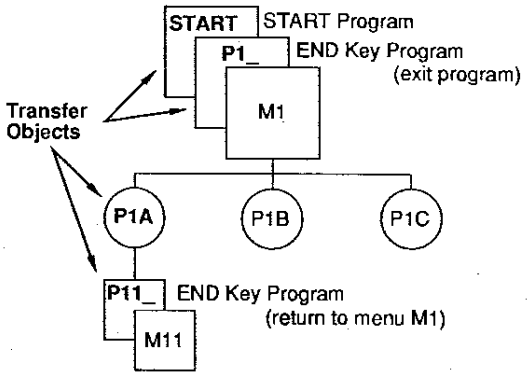
Object Name Provides Character String for Frame & Answer !

Many SAS/AF programs will need objects that serve a transfer function that is usually transparent to the user. The objects used are called transfer objects. Transfer objects are usually program objects but other types can be used. Many times the developer will want to set system options and initialize macro variables and other such things before bringing up the first menu. This is done via a START object. The START program is usually transparent to the user and transfers control immediately to the first menu. The START object is considered attached to the first menu. To exit the SAS/AF program we attach a program object to the END key (PF3) as a parent object . This program object will have an 'endsas;' statement in it returning control to the operating system. All objects attached to END keys are named as follows:

1. first letter of object type
2. menu level
3. underscore (_)

The object attached to the END key is the transfer object for progressing backwards thru the menu structure. It is most likely a program object. You may also want a program object intervening in a forward transfer thru the menu structure. In this case the transfer object is named just like other menu options. These more advanced concepts are illustrated below:

Special Program Objects



The transfer objects are listed in our automated documentation as follows:

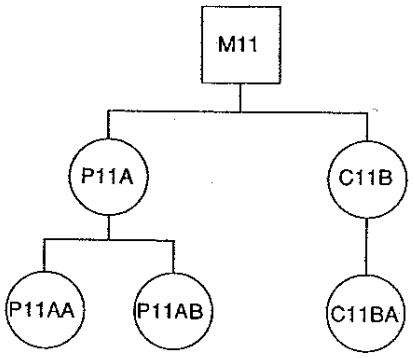
Automated Documentation:

Menu Level	Option	Type	Name	Description
1		menu	M1	
		program	**START**	**go to M1**
		program	**P1_**	**exit system**
	a	**program**	**P1A**	**go to M11**
	b	program	P1B	
	c	program	P1C	
11		menu	M11	
		program	**P11_**	**return to M1**

Start and End Key Programs Have Special Notation !

Most SAS/AF programs can be structured within the guidelines suggested above. However, in some special cases a single primary object is not enough. There are cases when we may need to call one program object from another or develop a tutorial with a sequence of CBT objects. These cases can also be incorporated into our naming scheme if we limit this 'substructure' to be hierarchical in nature. We develop option 'letter' trees analogous to the menu 'number' trees. This is illustrated for our example program as follows:

Hierarchical Structure Extended with Option Trees



The automated output we would generate based on this would have a column of 'option level' instead of 'option'. The option tree as such consists of primary objects with program flow proceeding in a hierarchical fashion. Each of these objects might be at the top of a stack of attached objects, etc. The overall naming scheme developed thus far can be summarized as follows:

1. first letter of object type
2. menu level, (number sequence)
3. option level, (letter sequence)
4. 2 digit field or frame number
5. 1 digit answer number

Conclusions:

1. Develop the following equivalence relation:

   Program Flow = Tree Structure = Object Name

2. Capture Object Name in data set with PROC CATOUT.

3. Generate Structure Chart with SAS® program !

Reference:

Bemis, Kerry G. (1987), Integrated Information Systems, SAS Users Group International Conference Proceedings, Cary, NC: SAS Institute Inc., 661-668.