SAS Macro Library -- An Easy Way to Increase the Productivity and Improve the Quality of SAS Programming

Lu Xu
Procter & Gamble Pharmaceuticals, Inc.

INTRODUCTION

SAS provide tools to support a SAS macro autocall library which can be accessed by any SAS user who has the access authority. The user can simply invoke the macros in the library from one's program "without" defining them in the program. Macros can be shared easily. The productivity of SAS programmers can be increased dramatically.

A SAS macro library can be set up in any computer system which supports SAS autocall facility. In the IBM MVS/TSO system, Partitioned Data Set (PDS) can be used to support the library; while in the IBM CMS system, the MACLIB utility can be used to support the library. In the VAX VMS system, source statement directory or a VMS text library can be used to support the library. There are many ways to organize the library: centralized, decentralized or hybrid.

No matter what computer system and library organization are used, the library must be reliable, and the macros must be correct. To make a quality product, a good process is essential. To build and maintain a quality SAS macro library a software engineering approach should be adopted. This is because building a library is a group activity, and during the life time of the library there will be multiple versions of the macro code and some personnel changes. The configuration management plan, quality assurance plan and design of the library are the important aspects of the process.

A configuration management plan defines all policies related to the project. It provides the methods for managing configuration items during the life time of the project, including the maintenance. A quality assurance plan describes the standards and the procedures for the project. Design answers the questions addressed in the requirement of the project in technical detail.

This paper discusses some of the issues of building a SAS macro autocall library.

CONFIGURATION MANAGEMENT

There are two kinds of things that need to be considered: configuration items and policies.

Configuration Items

For a SAS macro library, there are three types of items that need to be considered:

1. All the documents, including the requirement document, design document, user manual, configuration management plan, some reports, etc.

2. Source Code files.

3. Help library files.

Organizations and Policies

An organization to manage the project needs to be set up. The responsibility of each job needs to be defined. The process of evaluating and approving or disapproving changes to any project configuration item also needs to be established. All the configuration items should be controlled by their corresponding policies.

The following are some of the issues which should be addressed by the policy governing the source code file:

1. Current release repository

2. Backups and history

There should be a policy to handle an abnormal situation. For example, if fault is discovered in a macro, the macro needs to be taken off the library and the user should be informed by e-

mail immediately, also the help library needs to be updated.

Usually two documents are used to process change:

1. The new macro request -- document filled up by the user to indicate the desire of having a new macro.

2. The discrepancy report -- document filled up by the user to indicate faults of the existing macro in the library.

During the life time of the library, macro code will be updated either because SAS releases its new version or a better design and implementation for a macro is introduced. A policy dealing with when the library should be revised and how it should be done is necessary.

The key point regarding policy is that policy should be useful, easy to follow and relevant to the project.

QUALITY ASSURANCE

A quality assurance plan describes the standards and the procedures for the project. Some important standards and procedures related to a SAS macro library are:

Documentation Standards

It specifies the format for all documents.

Design and Coding Standards

It specifies some "should"s and some "should not"s. For example, in coding, all SAS keyword should be in upper case. In design, only the semantics which SAS manual tells explicitly should be used, "*short cut*" not supported by SAS should not be used.

Other Standards

If staffs are experienced, a cleanroom software development approach may be used. If so, the cleanroom approach is another standard of the project.

Procedure for Accepting User Provided Macros

This procedure describes the steps to accept a user provided macro. Some necessary steps would be: to complete requirement and design documents, recode according to coding standards, and pass the verification.

Code Review Procedure

This procedure describes how code should be reviewed. Use code walk through and/or code inspection techniques.

The key point regarding standards and procedures is that overall they should make the job easier.

REQUIREMENTS OF THE LIBRARY

The following are some of the library requirements:

Library Security

A library should be protected against unauthorized and erroneous access which may damage or destroy the library.

Information Hiding

For each macro defined in the library, only the necessary information for using the macro, (i.e., the interface), should be provided to the user, but not the implementation of the macro.

Low Coupling

Macros should not be coupled with each other or with the macros which a user defines in one's program.

No Side Effects

Macros should have no side effects. Neither undesired data set change and deletion nor undesired changes of macro variables' value should happen.

Consistent with SAS Tradition

SAS traditions, such as default data set name, should be followed as much as possible. For example, if no input data set is specified, the most recent data set will be used as the input data set; if no output data set is specified, the output data set will be the same as the input data set.

Macro Name

The macro name should be meaningful and expandable in a structured manner.

DESIGN

The following are some of the general design considerations:

The Macros Should Be General

For example, a certain operation can be applied on a series of variables, not just one variable, and the variable names do not have to be VAR1, VAR2 ... VARn.

Consistent in the Way of Invoking a Macro

There are two ways to invoke macros: statement-style and name-style. If combining elements of name-style and statement-style macro invocations in the same call, unexpected results may occur. There are several reasons to use name-style rather than statement-style invocation:

1. It is easy to identify a name-style invocation in SAS code.

2. Macros can be invoked anywhere in SAS code by name-style invocation.

3. Statement-style invocation may cause conflict if the macro name and variable are the same.

The Information Needed for Macro Executing Should Be Passed Through Parameters

Coupling and side effects will be minimized because all parameters are local variables. The keyword parameter passing method should be used, which can incorporate default values of parameters easily, make the order of the parameter irrelevant and help user to understand the function of the macro.

The Data Set and Macro Variable Created by the Macro Should Be Named in a Special Way

In order to eliminate the side effects, the data set and macro variable created by the macro should be named in a special way and let all SAS users who will use the library know this. These names will be reserved. All the data sets created within the macro are temporary which will be deleted at the end of the execution to keep the memory usage at a minimum.

Reference Environment and Nested Macro Invocation

In the SAS macro system, symbolic substitution method, which is similar to pass by value parameter passing method, is used for macro variable reference. Reference environment is dynamically determined based on invoking sequence. Macros do not need to be defined in a nested way. If a nest reference is needed, the only thing to do is to nest the invocation. Because of the dynamic characteristic, sometime the RUN statement determines the reference environment. See example 1 in the appendix for comparing a nested definition and invocation, and effect of the RUN statement. The general rule is that the RUN statement should always be used for the DATA and PROC steps.

The following are some of the specific design considerations:

Library Security

The write authority to the library should only be granted to one person (or account), who usually is the library administrator. The library project member should have the read authority; while the user should only have the execute authority.

Low Coupling and No Side Effects

The information should be passed by parameters for communication between open SAS code and macro, and between macro and macro. All local variables should be declared local to limit their visibility.

Consistent with SAS Tradition

To be consistent with SAS in handling default data set name, the input data set name of a macro should have a default value of _LAST_, and the output data set name should be determined by the statement: %if %length(&dsnout) = 0 %then %let dsnout = &dsnin; See example 1 of the appendix.

SOME OTHER ISSUES

Well-designed test cases are very important to a macro library. Macros must be verified (code review and testing) before being put into the library. The functions of a macro should be limited, otherwise the long and complicated codes will cause difficulties in verification.

CONCLUSION

To build and maintain a useful SAS macro library is not an easy job, but its reward is tremendous. A software engineering approach is very helpful during the life time of the library.

APPENDIX

Example 1

Macro definition A -- nested macro invocation

```
%MACRO prtmnby1(dsnin = _last_, byvar =,
var =);
 PROC SORT data = &dsnin;
  BY &byvar;
 RUN;
 %meanby(byvar = &byvar, var = &var)
 PROC PRINT data = &dsnin;
 RUN;
%MEND prtmnby1;

%MACRO meanby(dsnin = _last_, dsnout = ,
byvar =, var=);
  %if %length(&dsnout) = 0 %then %let
dsnout = &dsnin;
 PROC MEANS data = &dsnin noprint;
  BY &byvar;
  VAR &var;
 OUTPUT out = &dsnout;
RUN;
%MEND meanby;
```

Macro definition B -- nested macro definition:

```
%MACRO prtmnby1(dsnin = _last_, byvar =,
var =);
 %MACRO meanby(dsnin = _last_, dsnout =
, byvar =, var=);
   %if %length(&dsnout) = 0 %then %let
dsnout = &dsnin;
  PROC MEANS data = &dsnin noprint;
   BY &byvar;
   VAR &var;
  OUTPUT out = &dsnout;
 RUN;
 %MEND meanby;
 PROC SORT data = &dsnin;
  BY &byvar;
 RUN;
 %meanby(byvar = &byvar, var = &var)
 PROC PRINT data = &dsnin;
%MEND prtmnby1;
```

SAS code for invoking macro prtmnby1:

```
Title 'Correct Title';
%prtmnby1(byvar = group dose time, var =
ci)
Title 'Wrong Title';
```

Macro resolutions for A:

```
PROC SORT DATA = _LAST_;
BY GROUP DOSE TIME;
RUN;

PROC MEANS DATA = _LAST_ NOPRINT;
BY GROUP DOSE TIME;
VAR CI;
OUTPUT OUT = _LAST_;
RUN;

PROC PRINT DATA = _LAST_;
RUN;
```

The title for the printout is "Correct Title".

Macro resolutions for B:

They are the same as those for A except that these is no RUN statement after PROC PRINT,

which causes the title for the printout to be " Wrong Title".

Example 2

The JCL code for adding a macro to the library.

```
// EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=PDSname,DISP=MOD
//SYSIN DD *
./ ADD NAME=membername,LIST=ALL
%MACRO macroname;
  macro code
%MEND macroname;
./ ENDUP
/*
```

Notice that the member name in JCL and the macro name in SAS macro definition should be the same.

Example 3

The CMS command MACLIB for creating a macro library, adding a member to and deleting a member from the library.

1. Creating a library
   MACLIB GEN libname fn

2. Adding a member to the library
   MACLIB ADD libname fn

3. Deleting a member from the library
   MACLIB DEL libname membername

libname: file name of a macro library, file type must be MACLIB.
fn: name of macro definition file, file type must be COPY, file must contain fixed length, 80-character records.

REFERENCES

SAS Language Reference Version 6 First Edition -- SAS Institute Inc.

SAS Guide to Macro Processing Version 6 Second Edition -- SAS Institute Inc.

System/370 Job Control Language Second Edition -- Gary DeWard Brown

VM/SP CMS Command Reference Release 6 -- IBM

Fundamentals of Software Engineering -- Carlo Ghezzi

Concepts of Programming Languages -- Robert W. Sebesta

Comments and information, please contact:

Lu Xu
Procter & Gamble Pharmaceuticals Inc.
17 Eaton Ave.
Norwich, NY 13815
Tel. (607) 335-2992