Paper 014-2011

Macro Design Automation Tool (MDAT) – A methodology for storing macro design information, generating documentation and code templates, and automating parameter validation

Richard Schneck, BOGIER Clinical & IT Solutions, Inc., Raleigh, NC, USA
Mindy Rodgers, Eli Lilly and Company, Indianapolis, IN, USA

ABSTRACT

As members of a SAS® macro development team, we realized that we were spending considerable resources dealing with the basic tasks involved with the documentation and testing of each macro we develop, particularly with respect to information about the macro's parameters and source data requirements. If we could devise a systematic method of capturing this information, then it would be possible to develop a process to automate the creation of user documentation, parameter-validation code, testing documents, and even a customized template for the macro. Besides saving time and effort, major benefits of such a process would be the standardization and improved quality that would naturally result.

INTRODUCTION

When developing all but the simplest of macros, the tasks of designing, documenting, coding and testing a macro can be intensive. In addition to the primary logic and functionality of the macro, there are some repetitive tasks to be performed, primarily involving the macro's parameters and source data requirements. Basic information about each parameter must be included in the user documentation and in the source code header block; including the parameter description, the default value, whether it is case sensitive, whether a non-null value is required, and what non-null values are allowed. Furthermore, procedures for handling invalid parameter values and missing source data must be coded and documented. Independent programs must be written to verify that these procedures execute successfully.

In our regulated environment, all of this documentation is required in order to comply with policies and procedures, but producing it is time consuming. To achieve organizational objectives and allow time for innovation, we needed to free our developers from the "mechanics of validation" (Rodgers, 2007). We needed to identify consistent themes in our macro designs, and use this information to devise a method of automating these tasks. Our approach was to create a document for collecting each macro's design information, a centralized metadata library for storing the information, and a process that uses the metadata to build customized documentation and code templates. In addition, we developed a macro that dynamically validates the macro call during execution by checking the actual parameter values and source data against the information contained in the metadata.

Our goal was the development of a process that automates a significant portion of the repetitive work; is comprehensive (i.e., allows for the inclusion of non-standard parameter designs); ensures standardization; and encourages good design. In addition, since macros occasionally need to be modified after they are in use, it was important to develop a process that allows for efficient versioning.

The key components of the MDAT process are as follows:

- The MDAT Workbook - a customized file, built from a standard template, created separately for each individual macro to be developed using the MDAT process, that stores parameter design and other basic information
- The Design Metadata Sets - a series of SAS data sets that reside in a common macro library and contain the cumulative data from all of the MDAT workbooks for the individual macros
- The MDAT Loading Program - a SAS program that adds the data from an individual MDAT workbook to the design metadata sets
- The MDAT Execution Program - a SAS program that reads the design metadata observations for an individual macro, and generates customized documentation and code templates
- The Parameter Validation Macro - a SAS macro *called from within the macro being developed* that reads the design metadata observations for the specific macro, and validates the actual parameter values passed by the calling program

THE MDAT WORKBOOK

Implementation of the MDAT concept begins with a tool for collecting and storing macro design information. Our simple version of this tool is a multiple-worksheet Excel file that contains five worksheets. Certainly a more sophisticated interface could be developed, but ours is sufficient to illustrate the concept.

Each of the five worksheets in the MDAT workbook is used to collect a specific type of information about a macro. Collectively, the information in these worksheets paints a complete picture of the macro's design in terms of its general functionality, its parameters, and its source data requirements.

The five worksheets are summarized in the following table:

General_info General description of the macro	Info_item: A description of the information item Info_value: The information requested
Parms General parameter information with one line for each macro parameter	Parm_name: The parameter name Parm_description: The parameter description Parm_type: Whether the parameter is keyword or positional Default: The default value Required: Whether a non-null value of the parameter is required Case_sensitive: Whether the parameter is case sensitive Example: An example value of the parameter
Parm_values One line for each valid value of each parameter, or expression describing a type of valid values	Parm_name: The parameter name Value: The parameter value; can be either a fixed constant value, or [sas_dataset], [sas_variable], or [other] Other: Description of valid parameter values if VALUE='[other]'
Required_input Required source data sets and variables	SAS_dataset: The name of a required SAS data set (or parameter representing a required data set, preceded by "&") SAS_variable: The name of a required SAS variable (or parameter representing a required variable, preceded by "&") SAS_variable_type: Whether the variable is required to be numeric or character (leave blank if either type is allowed)
Special Special-case requirements, too complex to capture on the other worksheets	Parm_name: The parameter name (if the requirement is related to a particular parameter; otherwise enter '[none]') Requirement: The special-case requirement

Table 1. Summary of Worksheets in the MDAT Workbook

It must be emphasized that specific details in these worksheets can vary from the version presented here to meet the needs of your organization. However, once a template is established, it must remain constant and be followed strictly to ensure compatibility with the programs that drive the process. For each macro to be developed using this process, an MDAT workbook is created by copying the template to a project-specific location, and filling in the required information. Each MDAT workbook contains the same worksheets with the same names, the same columns and column headers required for each worksheet, and follows the same rules for entering information in the columns.

The displays that follow show the worksheets of the MDAT workbook in more detail. A workbook completed for a sample macro called "laundry_machine" is used as an example. (As will be seen in the sections that follow, our sample project is the development of version 2 of this macro. This is merely to illustrate that the process allows for macro versioning.)

Later, we will see how the information in these worksheets is used to build customized documentation and code templates.

GENERAL INFORMATION WORKSHEET

The general information worksheet, called **general_info**, is used to collect general information about the macro including the macro name, description, software version, general descriptions of the input and output, and assumptions.

Info_Item	*Info_Value*
MACRO NAME	laundry_machine
DESCRIPTION	Inputs a SAS dataset containing one or more loads of laundry, and creates a new copy of the dataset in which the status of each observation is changed from "dirty" to "clean".
SOFTWARE/VERSION #	SAS Version 9.13
INPUT	SAS data set containing dirty laundry as specified by the INDS parameter
OUTPUT	SAS data set containing clean laundry as specified by the OUTDS parameter.
ASSUMPTION 1	The macro is run from outside of any DATA step.
ASSUMPTION 2	The input dataset meets the specifications for a standard load of laundry.
ASSUMPTION 3	A LIBREF called METALIB is successfully assigned when the macro is executed.

Figure 1. General Information Worksheet of the MDAT Workbook

PARAMETERS WORKSHEET

The parameters worksheet, called **parms**, is used to collect basic information about each macro parameter, including the parameter name and description, the default value, whether a non-null value is required, whether it is a keyword or a positional parameter, and whether it is case sensitive. It contains one row for each parameter.

Parm_Name	*Parm_Description*	*Parm_Type*	*Default*	*Required*	*Case_Sensitive*	*Example*
INDS	The input SAS data set name	Keyword	[none]	Y	N	load_1_dirty
LOAD_IDVAR	The name of the variable in dataset &INDS representing the Load ID.	Keyword	[none]	Y	N	load_id
BRIGHTS_YN	A Boolean parameter indicating whether the machine should be set to wash bright-colored clothing	Keyword	Y	Y	N	N
LOAD_SIZE	The load size	Keyword	Large	Y	N	Small
WATER_TEMP	The water temperature	Keyword	Cold	Y	N	Hot
OUTDS	The name of the output dataset	Keyword	[none]	Y	N	load_1_clean
OUTDS_LABEL	A label for the ouput dataset	Keyword	[none]	N	Y	Load 1

Figure 2. Parameters Worksheet of the MDAT Workbook

PARAMETER VALUES WORKSHEET

The parameter values worksheet, called **parm_values**, is used to specify the valid values of the macro parameters. For parameters with a finite number of valid values that can be described with constant text, this worksheet contains *a separate line for each valid value*. Parameters with an infinite number of valid values, such as those representing the names of SAS data sets, appear on a single line on which the **value** column contains an expression in square brackets such as "[sas_dataset]". Parameters that do not fit into any of these categories appear on a single line on which the **value** column contains the expression "[other]", and the **other** column contains a free-text description of the parameter values.

Parm_Name	*Value*	*Other*
INDS	[sas_dataset]	
LOAD_IDVAR	[sas_variable]	
BRIGHTS_YN	Y	
BRIGHTS_YN	N	
LOAD_SIZE	SMALL	
LOAD_SIZE	LARGE	
WATER_TEMP	HOT	
WATER_TEMP	WARM	
WATER_TEMP	COLD	
OUTDS	[sas_dataset]	
OUTDS_LABEL	[other]	Any text string not exceeding 256 characters in length

Figure 3. Parameter Values Worksheet of the MDAT Workbook

REQUIRED INPUT WORKSHEET

The required-input worksheet, called **required_input**, is used to specify input data sets and variables that must exist in order for the macro to function correctly. The text entered in the **sas_dataset** or **sas_variable** column can be either a constant data set or variable name; or the name of one of the macro parameters, preceded by an ampersand (&), denoting a resolved parameter value. If it is the latter, the parameter must be a macro parameter that represents a SAS data set or variable, as appropriate for the column. The **sas_dataset** column must be populated on any row on which the **sas_variable** column is populated. For each required data set, there must be one row on which only the data set name is entered, with nothing entered in the **sas_variable** column. When a variable is entered in the **sas_variable** column, the **sas_variable_type** column is optional; leaving it blank indicates that there is no restriction on the variable type.

SAS_Dataset	*SAS_Variable*	*SAS_Variable_Type*
&INDS		
&INDS	ITEM_DESCRIPTION	character
&INDS	QUANTITY	numeric
&INDS	STATUS	character
&INDS	&LOAD_IDVAR	
LAUNDRY_SOAP		
LAUNDRY_SOAP	SOAP_BRAND	character

Figure 4. Required Input Worksheet of the MDAT Workbook

SPECIAL CASE WORKSHEET

The special case worksheet, called **special**, is used to capture requirements that are too complex to document on the other worksheets. The requirements may or may not be associated with a specific parameter. For requirements not associated with any parameter, the constant expression "[none]" is entered in the **parm_name** column. The inclusion of this worksheet allows the MDAT process to accommodate non-standard parameter designs.

Parm_Name	*Requirement*
WATER_TEMP	If &BRIGHTS_YN=Y and WATER_TEMP has a value other than COLD, then a warning message will be issued.
STATUS	If the value of variable STATUS is not 'DIRTY' on any observation in dataset &INDS, then a warning message will be issued.
LOAD_SIZE	If the sum total of variable QUANTITY exceeds 25 within any group of observations with the same value of &LOAD_IDVAR in dataset &INDS, and LOAD_SIZE has a value of SMALL, then an unbalanced load warning will be issued.
[none]	If a LIBREF called METALIB is not successfully assigned when the macro is executed, then a warning message will be issued.

Figure 5. Special Case Worksheet of the MDAT Workbook

DESIGN METADATA SETS / THE MDAT LOADING PROGRAM

Once a new MDAT workbook is completed for a particular macro, the MDAT loading program is executed to read the worksheets of the MDAT workbook and add the data to the MDAT design metadata library. This library consists of five SAS data sets, each one corresponding to one of the worksheets of the standard MDAT workbook. Whereas the MDAT workbook for each macro is stored in a project-specific location, the design metadata sets are stored in a centralized macro library. Each of these data sets contains the cumulative information from the corresponding worksheet from all of the MDAT workbooks, and has variables to identify the macro and version represented by each observation.

For example, the display below shows an excerpt from the data set containing the **required input** metadata derived from the worksheet shown in Figure 4 above, along with some metadata observations from other macros. Note that variables **macro_name** and **version** are used to allow different macros, and different versions of the same macro, to be represented in the data set. (Only version 2 of "laundry_machine" is seen here; assume that observations from version 1 are also in the data set.) Metadata sets that correspond to the other four worksheets of the MDAT workbook are similarly created by the MDAT loading program and stored in the metadata library.

mdat_required_input.sas7bdat

	macro_name	version	sas_dataset	sas_variable	sas_variable_type
1	laundry_machine	2	&INDS		
2	laundry_machine	2	&INDS	ITEM_DESCRIPTION	character
3	laundry_machine	2	&INDS	QUANTITY	numeric
4	laundry_machine	2	&INDS	STATUS	character
5	laundry_machine	2	&INDS	&LOAD_IDVAR	
6	laundry_machine	2	LAUNDRY_SOAP		
7	laundry_machine	2	LAUNDRY_SOAP	SOAP_BRAND	character
8	testmacro	1	&INDS		
9	testmacro	1	&INDS	&BYVAR	character
10	testmacro	1	&INDS	SUBJID	character

Figure 6. Example of MDAT Design Metadata Set – Required Input

The code shown below is from the project-specific MDAT loading program that reads the MDAT workbook for the macro "laundry_machine", version 2 (shown in Figures 1-5 above), and adds this information to the design metadata sets. The loading program calls a macro called **mdat_load**, which automates this process. The only information needed is the location of the MDAT workbook (input), the location of the folder containing the metadata sets (output), and the name and version number of the macro (to specify the values to be assigned to variables **macro_name** and **version** on the new observations to be created in the metadata sets).

If the metadata sets already contain observations for the specified macro name and version, macro **mdat_load** deletes these observations prior to adding the data from the MDAT workbook. It does not simply overwrite them. If it did, there could be unwanted observations remaining in the metadata sets.

```
%mdat_load(macroname = laundry_machine,
           version   = 2,
           mdatfile  = c:\MDAT\laundry_machine\laundry_machine_v2_mdat.xls,
           metalib   = c:\MDAT\Macro_library\Metadata);
```

CUSTOMIZED TEMPLATES / THE MDAT EXECUTION PROGRAM

After the MDAT loading program has been executed to add the information from the MDAT workbook to the design metadata sets, the MDAT execution program is executed. This program reads the metadata and uses it to automatically generate a series of customized templates that will be very useful in the ongoing development of the macro.

The code shown below is from the project-specific MDAT execution program that reads the design metadata sets for the macro "laundry_machine" (version 2), and uses this data to create the customized templates. The loading program calls a macro called **mdat_exec**, which reads the metadata observations for the specific macro and version. The only information needed is the location of the metadata sets (input), the location of the project-specific folder in which the customized templates will be created (output), and the name and version number of the macro (corresponding to the values of variables **macro_name** and **version** in the metadata sets).

```
%mdat_exec(macroname = laundry_machine,
           version   = 2,
           outlib    = c:\MDAT\laundry_machine\mdat_output,
           metalib   = c:\MDAT\Macro_library\Metadata);
```

As a result, the following customized templates are generated:

- User Documentation Template – a document containing a summary of the macro design information
- Testing Document Template – an Excel worksheet containing a series of numbered tests organized into categories
- Testing Program Template – a SAS program file containing code for executing the tests described in the testing document
- Macro template – a SAS program file containing a template for the macro, including a complete header block

Since these templates are generated from a single source (the design metadata sets, which contain the data from the MDAT workbook), we can be assured that the information in them will be accurate and consistent. However, it must be remembered that these files are intended to be a head start, not the finished product. The macro development team is ultimately responsible for ensuring that the documentation and testing procedures are correct and complete.

We recommend copying the templates to a separate location from the output folder where they are created, and renaming them, prior to using them. This prevents working files from getting confused with the output from the MDAT process, or being overwritten when the MDAT execution program is re-executed.

The displays that follow show the customized templates in more detail. As you review this output, you may refer to Figures 1-5 above to see how the information in these templates reflects the information in the MDAT workbook.

USER DOCUMENTATION TEMPLATE

In the user documentation template created by the MDAT execution program, the information from the MDAT workbook is neatly summarized in a compact form that is suitable for formal documentation. The display below shows only the parameter information table from this file. The actual file also includes a table of general information, and a table of "additional notes" consisting of requirements that are not associated with any specific parameter.

Requirements related to specific parameters							
Parameter	**Description**	**Type**	**Default**	**Valid values**	**Required?**	**Case Sensitive?**	**Comments**
INDS	The input SAS data set name	Keyword	N/A	Valid SAS dataset names	Yes	No	1) Dataset &INDS must exist; 2) Variable ITEM_DESCRIPTION must exist in dataset &INDS as a character variable; 3) Variable QUANTITY must exist in dataset &INDS as a numeric variable; 4) Variable STATUS must exist in dataset &INDS as a character variable
LOAD_IDVAR	The name of the variable in dataset &INDS representing the Load ID.	Keyword	N/A	Valid SAS variable names	Yes	No	Variable &LOAD_IDVAR must exist in dataset &INDS
BRIGHTS_YN	A Boolean parameter indicating whether the machine should be set to wash bright-colored clothing	Keyword	Y	Y, N	Yes	No	
LOAD_SIZE	The load size	Keyword	Large	SMALL, LARGE	Yes	No	If the sum total of variable QUANTITY exceeds 25 within any group of observations with the same value of &LOAD_IDVAR in dataset &INDS, and LOAD_SIZE has a value of SMALL, then an unbalanced load warning will be issued.
WATER_TEMP	The water temperature	Keyword	Cold	HOT, WARM, COLD	Yes	No	If &BRIGHTS_YN=Y and WATER_TEMP has a value other than COLD, then a warning message will be issued.
OUTDS	The name of the output dataset	Keyword	N/A	Valid SAS dataset names	Yes	No	
OUTDS_LABEL	A label for the ouput dataset	Keyword	N/A	The label to use for the output dataset	No	Yes	

Figure 7. User Documentation Template Created by MDAT Execution Program

TESTING DOCUMENT TEMPLATE

The testing document template created by the MDAT execution program is an Excel worksheet that contains the macro requirements from the MDAT workbook, automatically numbered and organized into categories for testing, for formal validation purposes. The following categories are included: optional parameters, required parameters, valid values, invalid values, other parameter-based requirements, and non-parameter-based requirements. Only the first two categories are shown in the display below.

Test Category	Test #	Test
Optional Parameters	1.01	Verify the macro executes without errors when OUTDS_LABEL has a null value
Required Parameters	2.01	Verify the macro terminates with an explanatory message if INDS has a null value
	2.02	Verify the macro terminates with an explanatory message if LOAD_IDVAR has a null value
	2.03	Verify the macro terminates with an explanatory message if BRIGHTS_YN has a null value
	2.04	Verify the macro terminates with an explanatory message if LOAD_SIZE has a null value
	2.05	Verify the macro terminates with an explanatory message if WATER_TEMP has a null value
	2.06	Verify the macro terminates with an explanatory message if OUTDS has a null value

Figure 8. Testing Document Template Created by MDAT Execution Program

TESTING PROGRAM TEMPLATE

The testing program template created by the MDAT execution program is a SAS program file in which a macro call for each test described in the testing document template has been automatically generated, along with a comment containing the test description. This saves the tester from the tedious work of looking up each test in the testing document and manually typing the comments and building the macro calls. The tester only needs to read the test descriptions that are already in the program, enter the parameter values, and add any code necessary to produce the conditions needed for each particular test. Only the first test description and macro call are shown in the display below.

```
/*============================================================
            Testing Optional Parameters
============================================================*/

** Test 1.01 - Verify the macro executes without errors when OUTDS_LABEL has a
               null value **;

%laundry_machine(inds        = <insert value>,
                 load_idvar  = <insert value>,
                 brights_yn  = <insert value>,
                 load_size   = <insert value>,
                 water_temp  = <insert value>,
                 outds       = <insert value>,
                 outds_label = <insert value>
                 );
```

Figure 9. Testing Program Template Created by MDAT Execution Program

MACRO TEMPLATE

In the macro template created by the MDAT execution program, much of the repetitive work normally involved in building a new macro is already completed. This includes the macro definition syntax complete with parameter names and default values; a complete header block that includes general information, parameter information, and a sample macro call; and even some standard program code.

Parts of this macro template are shown in the displays that follow. Once again, you may refer to Figures 1-5 above to see how the information in the template reflects the information in the MDAT workbook.

The following display shows the general information section of the header block created by the MDAT execution program:

```
/*************************************************************************************************
MACRO NAME           :  laundry_machine
DESCRIPTION          :  Inputs a SAS dataset containing one or more loads of
                        laundry, and creates a new copy of the dataset in which the
                        status of each observation is changed from "dirty" to
                        "clean".
SOFTWARE/VERSION#    :  SAS Version 9.13
INPUT                :  SAS data set containing dirty laundry as specified by the
                        INDS parameter
OUTPUT               :  SAS data set containing clean laundry as specified by the
                        OUTDS parameter.
ASSUMPTIONS          :  1. The macro is run from outside of any DATA step.
                        2. The input dataset meets the specifications for a standard
                           load of laundry.
                        3. A LIBREF called METALIB is successfully assigned when the
                           macro is executed.

-------------------------------------------------------------------------------------------------
```

Figure 10. Macro Template Created by MDAT Execution Program – General Info

The following display shows the parameter information section of the header block created by the MDAT execution program:

```
------------------------------------------------------------------------------------------
PARAMETERS:

Name          Reqd?  Default    Description and Valid Values
-----------   -----  ---------  ------------------------------------------------
INDS          Yes    N/A        The input SAS data set name (Valid values
                                include: Valid SAS dataset names)
LOAD_IDVAR    Yes    N/A        The name of the variable in dataset &INDS
                                representing the Load ID. (Valid values include:
                                Valid SAS variable names)
BRIGHTS_YN    Yes    Y          A Boolean parameter indicating whether the
                                machine should be set to wash bright-colored
                                clothing (Valid values include: Y, N)
LOAD_SIZE     Yes    Large      The load size (Valid values include: SMALL,
                                LARGE)
WATER_TEMP    Yes    Cold       The water temperature (Valid values include: HOT,
                                WARM, COLD)
OUTDS         Yes    N/A        The name of the output dataset (Valid values
                                include: Valid SAS dataset names)
OUTDS_LABEL   No     N/A        A label for the ouput dataset (Valid values
                                include: The label to use for the output dataset)

------------------------------------------------------------------------------------------
```

Figure 11. Macro Template Created by MDAT Execution Program – Parameter Info

Once the MDAT execution program has successfully created this macro template, the programmer can bypass the manual work normally involved with this task, and immediately start working on the main body of the code.

PARAMETER-VALIDATION MACRO: %PARMCHECK

Why bother creating the design metadata sets? Why not simply read the MDAT workbook directly and create the customized templates described above, in a single program? One reason is that the metadata serves as an information repository where the data from all of the MDAT workbooks is stored in one convenient location. Another is that storing the requirements in the metadata makes it is possible to update them by simply modifying the metadata, without the need to modify and re-validate the macro. But the primary benefit, which we discuss in this section, is that the metadata can be readily accessed by a calling macro and used to verify the parameter values and source data specified by a user *when the macro executes*.

Since the design metadata sets contain specific information about valid parameter values and source data requirements for any given macro, we realized that it was possible to write a macro (which we called PARMCHECK) that would be called from within any macro we develop, to validate the actual parameter values (and source data provided) against this information. For example, if the metadata indicates that a non-null value is required for a given parameter, and a null value is passed in for that parameter, then macro PARMCHECK detects this as an invalid condition. To give another example, if the metadata indicates that parameter ABC represents a SAS variable, and the actual value passed in for ABC is not a valid SAS variable name, macro PARMCHECK detects this as an invalid condition. Furthermore, if the metadata indicates that variable ABC must exist in data set XYZ, then PARMCHECK detects an invalid condition if this is not the case. In all such cases, if an invalid condition is detected, PARMCHECK prints an appropriate message in the SAS log file, and sets the value of a global error-status macro variable. After PARMCHECK has executed, the calling macro aborts if the value of this macro variable indicates that an invalid condition has been detected.

Macro PARMCHECK, therefore, eliminates the need to manually write the code to perform these checks.

Note: The only metadata sets used by PARMCHECK are those that correspond to the PARMS, PARM_VALUES, and REQUIRED_INPUT worksheets of the MDAT workbook. Special-case requirements entered on the SPECIAL worksheet, and those with a value of "[other]" on the PARM_VALUES worksheet, are not used by PARMCHECK. Any coding necessary to verify these requirements must be added by the macro programmer following the call to PARMCHECK.

The following display shows the portion of the macro template created by the MDAT execution program that contains standard program code, including the call to PARMCHECK, and a place to enter any additional parameter-validation code that is not covered by PARMCHECK.

```
%*=========================================;
%*                                         ;
%*        Begin Parameter Checking         ;
%*                                         ;
%*=========================================;

%** Automated parameter checking **;

%parmcheck(laundry_machine,2);


%** Other parameter checking **;

/* <PARAMETER CHECKING CODE NOT PERFORMED BY PARMCHECK,
      I.E., "OTHER" PARAMETER VALUES, AND REQUIREMENTS
      FROM THE "SPECIAL" METADATA WORKSHEET */


%** Stop execution if errors have been detected **;

%if &error_status %then %do;
  %put NOTE: Macro LAUNDRY_MACHINE will stop executing.;
  %return;
%end;

%*=========================================;
%*                                         ;
%*     Begin Main Processing Section       ;
%*                                         ;
%*=========================================;




%mend laundry_machine;
```

Figure 12. Macro Template Created by MDAT Execution Program – Standard Code

Note that the two positional parameters in the call to PARMCHECK represent the macro name and version of the macro as recorded in variables **macro_name** and **version** in the metadata sets.

SUMMARY

Initial implementation of the MDAT process involves the creation of an MDAT workbook template suitable for your organization. After the template is finalized, the SAS programs that load the MDAT data into metadata sets, build the customized documentation and code templates, and perform automated parameter validation must be developed.

Once the process is implemented, using it for a specific macro development project involves (1) copying the template to a project-specific location, and completing it for the specific macro design; (2) running the MDAT loading program to update the metadata sets; (3) running the MDAT execution program to generate the customized templates; and (4) copying the customized templates to the appropriate locations for ongoing project development. Once the macro is completed, any program that calls it must access the metadata library in order to execute macro PARMCHECK to perform automated parameter validation.

The flowchart that follows shows an overview of the MDAT process:

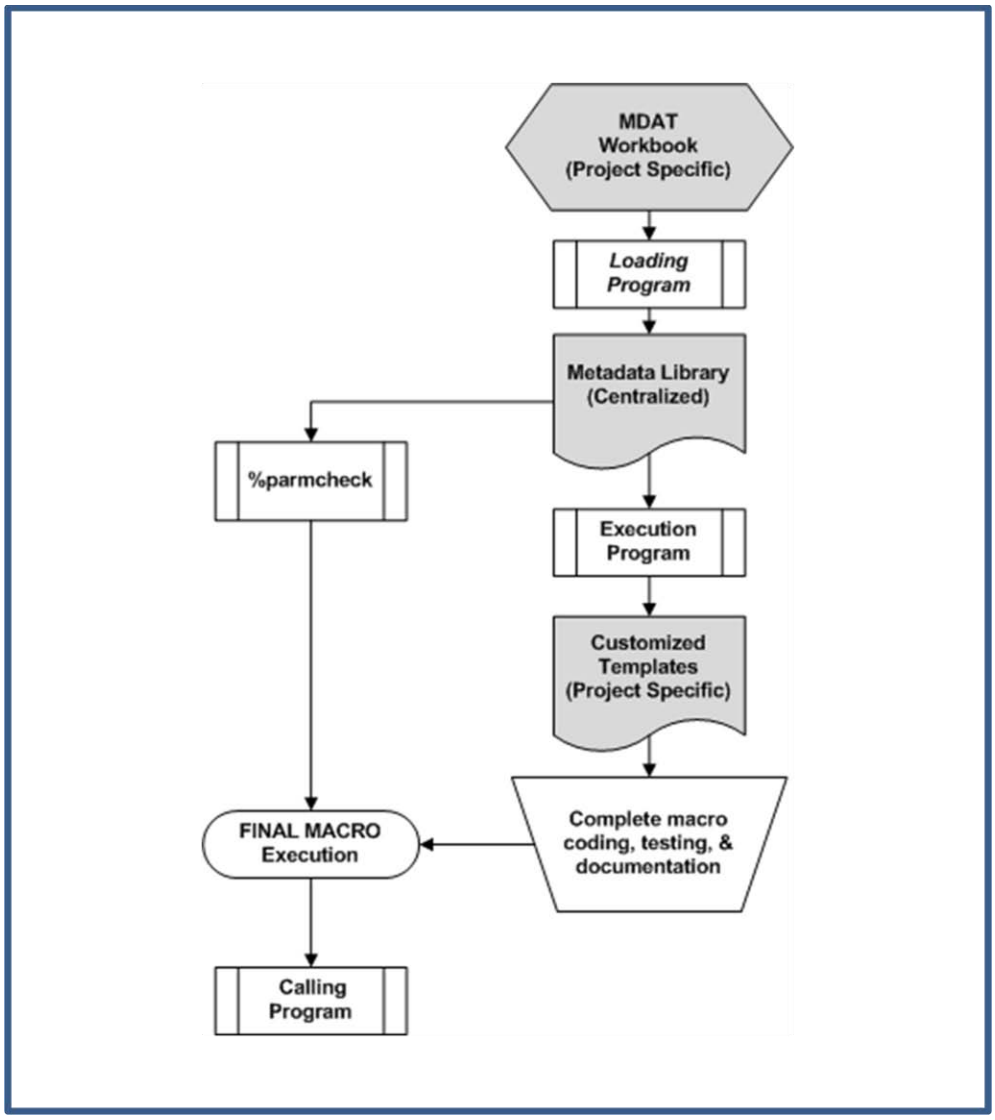

Figure 13. Overall MDAT Process Flow

CONCLUSION

The value of the MDAT process is not only its potential for significant time savings, but also the improved quality that naturally results from standardization. Using the templates generated by this process ensures that the documentation and code headers for every macro will contain the same types of information, in the same format, with consistent language to describe similar design elements. The inconsistencies and omissions that inevitably result from human error are eliminated. Finally, a standard convention for organizing and cataloguing requirements for formal validation is built into the process, and automatically implemented in the generation of the testing documents.

Many variations of the MDAT process described in this paper are possible. The process can be customized for the needs of any macro development team depending on its particular organizational needs, its documentation requirements, and the type and complexity of the macros that it usually develops. Enhancements might include the addition of new parameter value types. New worksheets can be added to automate the validation of multi-parameter value combinations and conditional defaults. The MDAT workbook can be enhanced to include Excel VBA, or replaced with a GUI.

But the core concept remains the same: a single source of basic macro design information from which all related documents and code templates are derived, resulting in uniformity, improved quality, and greater efficiency.

REFERENCES

Rodgers, M./Steffens, G. "The Art of Code Validation" *Proceedings of the SAS Global Forum 2007 Conference*. Available at http://www2.sas.com/proceedings/forum2007/TOC.html

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Greg Steffens, Craig Hansen, Kallin Carter, Retha Bogier, and Jason Morgan for their vision, participation, expertise, and support toward the successful development and implementation of this concept and paper.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name:	Richard Schneck	Mindy Rodgers
Enterprise:	BOGIER Clinical & IT Solutions, Inc.	Eli Lilly and Company
Address:	900 Ridgefield Drive, Suite 390	Lilly Corporate Center
City, State ZIP:	Raleigh, NC 27609	Indianapolis, IN 46285
Work Phone:	(919) 816-2535	(317) 277-7062
E-mail:	richard@richardschneck.com	msrodgers@lilly.com