PharmaSUG2011 – Paper MA03

A Six-Sigma/DMAIC Approach to Defining a SAS Macro Repository Strategy

Sandy Paternotte, PPD, Inc., Wilmington, NC
Tom Fritchey, PPD, Inc., Wilmington, NC

ABSTRACT

Most programming organizations eventually gravitate to developing a strategy to store in some repository reusable SAS® macro code for common use by their staff of SAS programmers. Such strategies should be developed to meet their organization's goals and needs. What is the right strategy for your organization? How do you begin to develop the most effective strategy? This paper will offer a sound Six-Sigma quality-based approach to a common-use macro repository strategy using the DMAIC (Define > Measure > Analyze > Improve > Control) process that will yield a guide to assessing what macros should best be placed into your firm's global macro repository.

INTRODUCTION

There can be a wide variety of approaches to building a common-use macro library or repository. Organizations may go as far as developing a reporting system where almost all generation of statistical analysis of clinical trial data is fully automated. Others may deploy a different strategy where a large number of very granular "building block" utility macros are developed and maintained. Still other organizations may have no clear strategy for common-use macro development and simply end up storing their programmers' favorite SAS macros.

Selecting the right approach to developing common-use SAS macros for an organization should be approached with caution and wisdom. This strategic decision will have a long-lasting impact as it sets the standards for how reusable code is designed, implemented, and maintained for years to come and will ultimately determine how well efficiency and effectiveness goals are achieved.

This paper introduces the Six-Sigma quality continuous improvement methodology, DMAIC, and how its specific application of the D and M phases can significantly assist in launching your organization's strategy for selecting and developing SAS macro reusable code. For the purposes of clarity and consistency throughout this paper, the phrase *SAS macro repository* will be used to describe the general collection of SAS macros for your organization's common access and application. Note that your organization may need multiple such repositories, each for a distinct purpose (e.g. common-use utility, therapeutic-specific, CDISC, safety, efficacy, reporting, figures, etc.).

RATIONALE FOR A DEFINED PROCESS FOR SELECTING SAS MACROS

At first glance, perhaps having a process for selecting SAS macros for a repository seems like overkill? Why not take every macro proposed by programmers and dump them all into a sharable repository? Doesn't this maximize reusability and sharing? The answer to this is found in the principle that *functionality and complexity are linked*; i.e. the more you want your SAS macro to be able to do, the higher its complexity will be.

Examples of desired functionality for a single, reusable macro may be its ability to handle various data structures, conditionally perform multiple operations, or produce customizable output. The subsequent increase in complexity may be manifested as advanced code which can increase the costs of debugging and maintenance or require a complicated interface (e.g. macro parameters, input data structure) which can increase the costs of training for staff.

In the previous example we examined a single SAS macro, but this same principle applies to your SAS macro repository. Every macro you add to the repository (increasing the functionality) comes with maintenance and training costs. The more macros you add, the more you have increased the complexity and costs of your repository. *A well-defined process for selecting these SAS macros will enable you to strike the right balance between functionality and complexity, an essential step to maximize the efficiency of your repository!*

AN INTRODUCTION TO THE DMAIC METHODOLOGY

The Six-Sigma DMAIC methodology is a general approach that can be applied when any actions are to be taken to improve a process, tool, etc., and can be a highly useful method to apply to selecting a strategy for SAS macro repository development. The table below introduces DMAIC in a general form.

Figure 1 – DMAIC (general)

Phase	Name	Description
D	**Define**	• Define the problem • Begin understanding the objectives of the improvement effort
M	**Measure**	• Look at the objectives from the Define phase and establish the criteria by which a proposed solution or improvement alternative will be measured • Develop the detailed, weighted list of evaluation criteria
A	**Analyze**	• Brainstorm alternative solutions or improvements • Evaluate these alternatives, measuring each against the criteria from the Measure phase and defining the risks associated with their implementation or application • Based on the assessment of each solution's performance and risks, select the best solution to implement
I	**Improve**	• Develop all plans to be used in implementing the improvement: ✓ Implementation sequence/plan ✓ Training Materials and Plan ✓ Risk/Contingency Action Plans ✓ Control Plan (follow-up usage measurements to assess success and define further process improvements) • Execute implementation of improvement
C	**Control**	• Measure improvement effectiveness by executing the Control Plan

DEVELOPING A STRATEGY FOR THE SAS MACRO REPOSITORY USING DMAIC

The table below introduces how DMAIC can be applied to developing a strategy for defining and maintaining any single SAS macro repository.

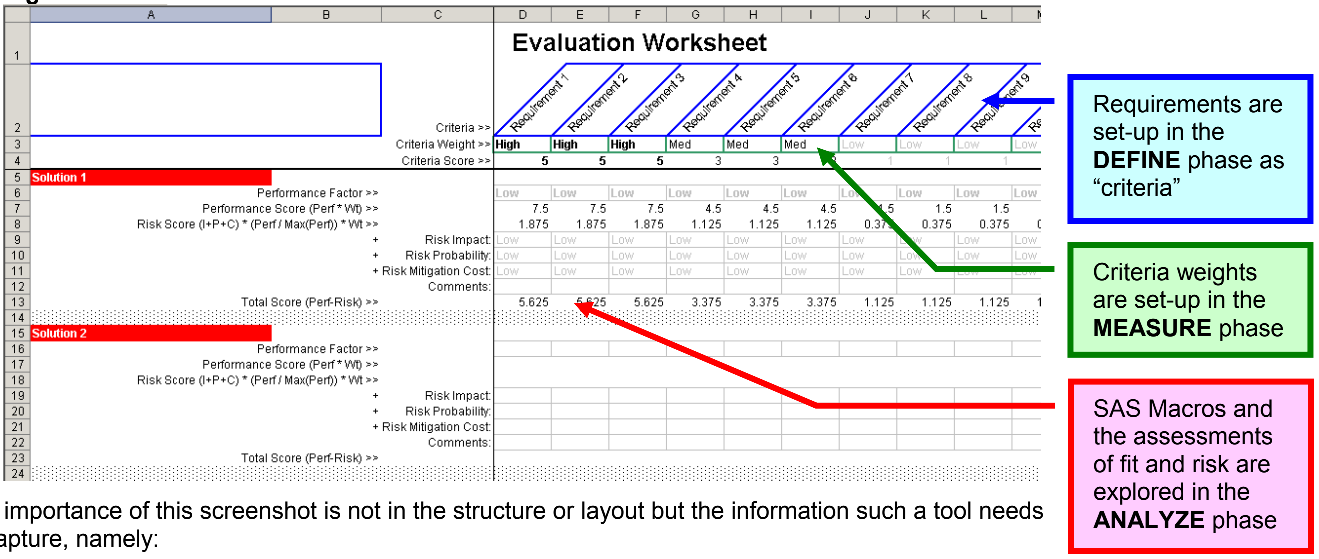
Figure 2 – DMAIC (for SAS macro repository strategies)

Phase	Name	Description
D	**Define**	• Define the requirements and attributes desirable for SAS macros that make up the repository • Once these requirements and attributes are defined, we have described *what* our SAS macro repository will eventually contain
M	**Measure**	• Look at each SAS macro requirement/attribute and consider its relative merit (weight) as compared to all requirements/attributes we have defined • Once our requirements/attributes are weighted, we have described *how* we will evaluate any SAS macro candidate for its worth in implementing it as a member of the SAS macro repository
A	**Analyze**	• If a SAS macro has already been programmed, evaluate it against the measurement criteria set up in the M phase • If a SAS macro has yet to be developed, but the need (basic functionality) has been defined, use the measurement criteria set up in the M phase as a guide for its design
I	**Improve**	• Develop all plans to be used in implementing the SAS macro: ✓ Implementation sequence/plan ✓ Training Materials and Plan ✓ Risk/Contingency Action Plans ✓ Control Plan (follow-up usage measurements to assess success and define further process improvements) • Execute induction of the SAS macro into the repository
C	**Control**	• Measure effectiveness of the SAS macro by executing the Control Plan

SETTING UP AN EVALUATION TOOL FOR THE DMAIC PROCESS

Shown below is an Excel-based tool for implementing the D, M, and A phases of the DMAIC methodology.

Figure 3 – D>M>A SAS Macro Evaluation Tool



	A	B / C	D	E	F	G	H	I	J	K	L
1			Evaluation Worksheet								
2		Criteria >>	Requirement 1	Requirement 2	Requirement 3	Requirement 4	Requirement 5	Requirement 6	Requirement 7	Requirement 8	Requirement 9
3		Criteria Weight >>	High	High	High	Med	Med	Med	Low	Low	Low
4		Criteria Score >>	5	5	5	3	3	3	1	1	1
5	Solution 1										
6		Performance Factor >>	Low	Low	Low	Low	Low	Low	Low	Low	Low
7		Performance Score (Perf * Wt) >>	7.5	7.5	7.5	4.5	4.5	4.5	1.5	1.5	1.5
8		Risk Score (I+P+C) * (Perf / Max(Perf)) * Wt >>	1.875	1.875	1.875	1.125	1.125	1.125	0.375	0.375	0.375
9		+ Risk Impact:	Low	Low	Low	Low	Low	Low	Low	Low	Low
10		+ Risk Probability:	Low	Low	Low	Low	Low	Low	Low	Low	Low
11		+ Risk Mitigation Cost:	Low	Low	Low	Low	Low	Low	Low	Low	Low
12		Comments:									
13		Total Score (Perf-Risk) >>	5.625	5.625	5.625	3.375	3.375	3.375	1.125	1.125	1.125
14											
15	Solution 2										
16		Performance Factor >>									
17		Performance Score (Perf * Wt) >>									
18		Risk Score (I+P+C) * (Perf / Max(Perf)) * Wt >>									
19		+ Risk Impact:									
20		+ Risk Probability:									
21		+ Risk Mitigation Cost:									
22		Comments:									
23		Total Score (Perf-Risk) >>									
24											

Requirements are set-up in the **DEFINE** phase as "criteria"

Criteria weights are set-up in the **MEASURE** phase

SAS Macros and the assessments of fit and risk are explored in the **ANALYZE** phase

The importance of this screenshot is not in the structure or layout but the information such a tool needs to capture, namely:

1. The defined requirements and their relative importance.
2. The solutions being considered with an evaluation of their performance against your requirements and basic risk assessment.

DEFINING THE ATTRIBUTES FOR THE SAS MACRO REPOSITORY

In the DMAIC Define phase we specify the basic requirements and desired attributes for SAS macros that will go into our repository. Some example questions that might help form these requirements and attributes might be:

- ✓ Compute-platform independence? (I.e., can any macro have environment-specific code in them? E.g., are Unix-only constructs acceptable?)
- ✓ Will our macro code potentially be shared with parties outside of our company?
- ✓ Does the macro functionality need to be limited to a given purpose of the repository? (Examples might be: operate on only Oncology data? Be only general utility and not data-specific? Only support Table/Listing/Figure programming? Only support setting up analysis data/variables?)
- ✓ Does the macro have to fulfill a critical business need or will our macros fulfill any purpose so long as they are handy?
- ✓ What is our long term aim? To eventually maintain a collection of hundreds of small macros or only a critical few?
- ✓ Is it expected that our programmers will be trained on these macros? Or do our macros need to be designed based on having a very small budget for training?
- ✓ Are there critical programming standards to which any macro must comply?

As you review the above example list of *criteria*, other more basic questions might need to be answered regarding the different repositories that may be needed, budgetary concerns for training, macro maintenance hours longer term, process standards, programming standards, etc.

As the list of requirements/attributes takes shape, it will be useful to think of them as "gold standards" by which any macro would be compared. As your staff programmers escalate potential SAS macros for assessment against these standards or as requirements for a new macro are being developed, think of these requirements as being tests that the macros will have to pass in order to be approved for your repository.

Using the example list of questions above, a requirements list might be:

- ✓ The macro must be free of any platform-specific language
- ✓ The macro will not be shared with any parties outside of our company
- ✓ The macro functionality must be limited to table programming and must only support our company's standard Summary Table designs
- ✓ The macro must fulfill a unique need and cannot duplicate the functionality of any existing macro in the repository

- ✓ As we aim to host only the critical few macros for future maintenance, the macro must demonstrate significant functionality that replaces (automates) multiple processing steps
- ✓ The macro and its arguments must be simple to understand by observing the parameters' names (positional parameters are not allowed) and by reading the macro header

These are only examples. Your organization may have many more or different requirements for your macro repository.

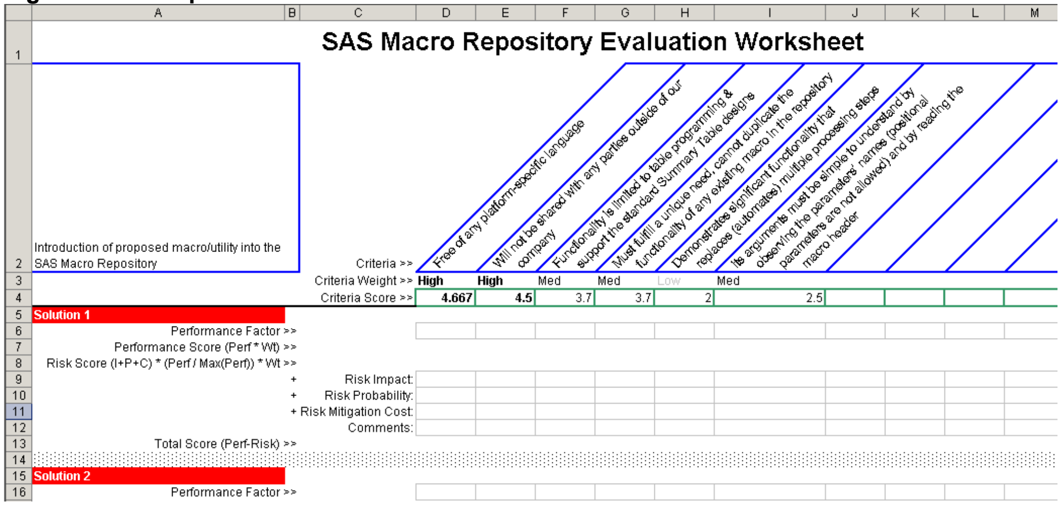
MEASURING THE ATTRIBUTES OF THE SAS MACRO REPOSITORY

From our language in the previous section, it should be clear that your list of attributes may include some items that are strict requirements while other items may add value but ultimately be optional. It is in the Measure phase that we distinguish these attributes by assigning them relative weights. Again, these will be specific to your organization.

It is the “gold standard” criteria and criteria weights from the Define and Measure phases that allow us to compare SAS macro candidates with a standard approach.

Using the examples we have used above, shown below in Figure 4 is the Excel-based evaluation tool that has been updated with *criteria* as defined from our Define phase and the *criteria weights* as defined from our Measure phase. We now have our gold standard and are ready to test SAS macro candidates against it!

Figure 4 – Example Evaluation Tool After the D and M Phases



SAS Macro Repository Evaluation Worksheet

Introduction of proposed macro/utility into the SAS Macro Repository

	D	E	F	G	H	I	J	K	L	M
Criteria >>	Free of any platform-specific language	Will not be shared with any parties outside of our company	Functionality is limited to table programming & support the standard Summary Table designs	Must fulfill a unique need, cannot duplicate the functionality of any existing macro in the repository	Demonstrates significant functionality that replaces (automates) multiple processing steps	Its arguments must be simple to understand by observing the parameters' names (positional parameters are not allowed) and by reading the macro header				
Criteria Weight >>	**High**	**High**	Med	Med	Low	Med				
Criteria Score >>	**4.667**	**4.5**	3.7	3.7	2	2.5				
Solution 1										
Performance Factor >>										
Performance Score (Perf * Wt) >>										
Risk Score (I+P+C) * (Perf / Max(Perf)) * Wt >>										
+ Risk Impact:										
+ Risk Probability:										
+ Risk Mitigation Cost:										
Comments:										
Total Score (Perf-Risk) >>										
Solution 2										
Performance Factor >>										

ANALYZING SAS MACRO CANDIDATES FOR THE REPOSITORY

In the Analyze phase SAS macro candidates for the repository are evaluated in two ways. First and most obvious, how does the candidate perform against our gold standard as defined by the criteria and criteria weights? It is this analysis where the value of the efforts in the previous two phases is realized. Development teams should consider performing this analysis on their conceptual SAS macro designs to highlight areas of weakness. This proactive step can often increase the value of the initial release of a SAS macro. Furthermore, this analysis helps to spot SAS macro candidates that, while good, may not offer much value (i.e. increased efficiency) for their cost (e.g. complexity, training, maintenance).

In addition to analyzing performance against the measures of success (criteria), a high-level risk analysis should also be performed against the SAS macro candidate. This risk assessment will help to highlight development efforts that, though they seem promising, may backfire and lower efficiency if they fail to reach expectations. Again, this analysis should be used proactively to identify mitigation strategies or consider an altogether different approach.

As an example from Figure 4, we might assess that the *risk impact* is quite high for the last criteria (that the macro arguments must be simple to understand) because should the macro arguments end up being actually quite complex, programmers may not use the macro correctly resulting in significant rework and wasted cost or they may end up not using it at all. Further, we may rate the *risk probability* quite high as well because perhaps the candidate macro we are evaluating is, by its very nature, very complex. Through this objective evaluation effort during the analyze phase, we have brought into focus the need for risk mitigation. Perhaps in this example, we all agree that we will invest in significant training for this macro when it is introduced into the repository. Conversely, perhaps we realize that this macro is not a good candidate after all.

As you can see, without the rigor of applying the DMAIC methodology, significant risks exist that could jeopardize the otherwise great cost-savings potential of a SAS macro repository.

IMPLEMENTING THE SAS MACRO INTO THE REPOSITORY

As SAS macro candidates are evaluated and approved for addition into the repository, take care not to overlook the importance of a well-planned and executed implementation. This includes:

- ✓ Ensuring that the SAS macro meets any defined coding standards.
- ✓ Developer documentation (e.g. flowcharts or algorithm details) to assist with and potentially lower the costs of future maintenance.
- ✓ Planning and executing testing or validation sufficient for the intended use.
- ✓ Compiling and publishing training and help documentation.

Failing to properly implement a SAS macro in the repository can lead to confusion and wasted time, misuse and errors, or complete neglect.

CONTROLLING THE EFFECTIVENESS OF THE SAS MACRO REPOSITORY

The last phase in the DMAIC methodology points to continuous improvement. While the repository may be functioning well today, we know that our business environment is not static. Indeed, ignoring this phase cripples the ability of your process to proactively adapt to change, often resulting in change happening only in reaction to critical needs. Such a reactionary cycle typically puts management and development teams under pressure to implement changes without proper planning or resources. Besides the obvious heightened risks that the changes contain errors and that staff suffer due to overtime, this also increases the risk that changes are introduced to the repository that don't measure up to your gold standard, ultimately increasing costs and decreasing efficiency.

Controlling the effectiveness of a SAS macro in the repository includes:

- ✓ Measuring the practical value (e.g. actual efficiency gains vs. predicted, an assessment of the return on investment) of the SAS macro in use
- ✓ Identifying milestones to proactively trigger when to improve, replace, or retire the SAS macro
- ✓ Gaining feedback to improve the measures of success for the repository in order to keep your gold standard in line with changing business needs

The organization that consistently follows through with the Control phase will have the information needed to successfully and proactively meet future challenges. That being said, of all the phases in the DMAIC methodology, the Control phase is the hardest to execute well.

CONCLUSION

This paper has demonstrated that the Six-Sigma DMAIC (Define > Measure > Analyze > Improve > Control) process lends itself quite well to an objective way to set up and maintain a SAS macro repository that complements your company's goals.

- In the Define and Measure phases, you define your *gold standard* for SAS macros. These phases may be repeated as often as your company's goals change and/or as you learn where you may need to adjust your standards.
- The Analyze phase is repeated when you are ready to test a new or updated macro against your gold standard.
- The Implementation phase is repeated as you roll out a new or updated macro to your repository and into use by your staff.
- The Control phase helps provide the information you need to adapt to changing business needs and ensures your long term success.

RECOMMENDED READING

The authors recommend browsing the internet for information about continuous process improvement and the DMAIC methodology. There is a huge volume of information and real-life application of Six-Sigma methodologies spanning decades of experience across many different industries.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:
Name: Sandy Paternotte and Tom Fritchey
Company: PPD, Inc
E-mail: sandy.paternotte@ppdi.com and thomas.fritchey@ppdi.com
Web: http://www.ppdi.com