PharmaSUG2012 - Paper TF11

Validating SAS Macros and Updated SAS Version

Sy Truong, Meta-Xceed, Inc. (MXI), Fremont, CA
Carey Smoak, Roche Molecular Systems, Inc., Pleasanton, CA

ABSTRACT

The rapid updates to the SAS software implemented in a regulated controlled environment within the pharmaceutical/biotechnology industry have created validation challenges. Users require the efficiencies of automated macros and the functionality of the new version of SAS, but the complexity of computing environments in a dynamic hardware and software platform makes it difficult to maintain a validated SAS system. This paper will share practical methods and techniques used in the validation of SAS macros and upgrade of SAS to a new version on a production server. Some of the areas it will cover include:

- **SAS Macro Validation –** Testing multi-use macros to ensure proper error checking and integrity.
- **SAS IQ/OQ** – SAS 9.2 is bundled with a series of scripts for performing installation and operational qualifications.
- **Performance Qualification** – Performance tuning and testing are not covered by SAS tools. This paper will illustrate methods and macros on how to perform this testing.
- **File Migration** – Moving SAS files while maintaining all the same permissions and related applications requires rigorous testing.

Maintaining validated SAS macros is critical in performing analysis of clinical data in a regulated environment. The lessons learned are based on real world SAS macros validation efforts within a system upgrade will be shared in this paper to save time and ensure the integrity of analysis performed in an environment constantly being updated.

INTRODUCTION

Validation of a SAS system most commonly occurs during an upgrade from an older version of SAS or implementing a new macro library. Validation of a SAS server has been previously described in terms of technical issues (Truong, Smoak 2010) and management issues (Smoak, Truong 2011) and is replicated in this paper to present a comprehensive paper on the topic of validation of SAS macros. The examples used in this paper include migrating from SAS 9.1.3 to SAS 9.2 and moving from a legacy server and operating system to the Windows platform. In all cases, similar validation challenges are confronted. It is recommended that you first acquire a global view of the system and identify the architecture. Only after gaining this perspective would it be useful to then zoom in and focus on individual components. This allows you to access the scope and interconnectedness of each component so that your validation efforts are balanced and thorough. Once the architecture is clearly understood, the requirements and functional specifications of each component are documented. These functional specifications then drive the validation and testing (Truong, Smoak 2010).

It is important to follow these steps in a systematic and orderly fashion since they are interdependent. Documentation of each step in the validation process is also essential in capturing and proving that the validation effort was done properly. Besides documenting each step, it is also important to capture the traceability of each validation task. For each test case that is performed, there is an associated functional specification which then is connected to the requirements for that particular component of the system as a whole. The map or traceability matrix that ties all these validation components together is pivotal to an auditor. Proper documentation will make the difference between a successful validation audit and a complete failure (Truong, Smoak 2010).

The main goal of the validation effort is to ensure that the installation and implementation of the SAS system and its associated macros function as intended by the vendor (SAS Institute) and your organization. The validation will

ensure this success. In addition to this goal, the documentation of your validation effort will also ensure the integrity of your computing environment and the compliance with regulatory requirements such as the CFR Part 11 within the biotechnology and pharmaceutical industry (Truong, Smoak 2010).

SAS INSTALLATION QUALIFICATION (IQ) AND OPERATIONAL QUALIFICATION (OQ)

The purpose of the Installation Qualification (IQ) is to ensure that the SAS software has been properly installed on the server. The SAS IQ tool checks the installation files and provides a checksum which documents that the SAS software has been installed properly. The purpose of the Operational Qualification (OQ) is to ensure that SAS operates properly on the server. The SAS OQ tool checks that SAS operates correctly and provides documents that show the SAS software operates correctly (Truong, Smoak 2010).

A checksum can be defined as follows:

> "A checksum is a count of the number of bits in a transmission unit that is included with the unit so that the receiver can check to see whether the same number of bits arrived. If the counts match, then one can assume that the complete transmission was received"
>
> (http://www.mnhs.org/preserve/records/electronicrecords/erglossary.html 26Jul2010).

In this case, each file in the IQ/OQ has a unique string (checksum) associated with it. When the SAS IQ/OQ tool is applied, it compares the current checksum with the known checksum and if the two checksums match, then the file passes validation (IQ/OQ) (Truong, Smoak 2010).

In addition, the SAS IQ/OQ should follow company standards. Protocols for the IQ/OQ should be written, reviewed and approved before executing the test scripts provided by the SAS Institute. The protocols should contain step-by-step instructions (including screenshots) so that the person executing the test scripts and the persons reviewing and approving the executed steps in the protocol are absolutely clear about the steps necessary to show that the SAS system is validated (IQ/OQ). The person executing the protocol should print-out the completed protocol, the test scripts, the test script log files and the test script output for review and approval. These documents should also be available for inspection in case of an audit by the FDA or other regulatory agencies (Truong, Smoak 2010).

PERFORMANCE QUALIFICATION

The Performance Qualification (PQ) for SAS ensures that the system will function properly in a real world environment with the appropriately sized input and output datasets that are processed along with the CPU intensive processing required for running SAS statistical procedures. This may vary widely depending on the types of analysis that are performed upon specific data. The goal is not to measure the exact performance of each and every example you have, but rather to identify the largest data and the most complex analysis simulating a stress test of the system. This would push the system to the maximum computing capacity while measuring if it still generates the desired result in a reasonable amount of time. Because the PQ is by definition unique to each environment, it is challenging to come up with a fix set of tests that can be applied to all systems. This is the reason why SAS has provided a set of tools as part of the installation for performing IQ/OQ but does not have anything to assist the user with the PQ. This section will describe a process and macro programs established specifically for the purpose of testing the Performance Qualification of SAS during installation (Truong, Smoak 2010).

The parameters of the test have to be clearly defined in order for the results to be meaningful. The following lists some required key conditions of the Performance Qualification testing for the installation of SAS 9.2 (Truong, Smoak 2010).

1. **Test Scripts** – A group of SAS programs has been identified to be representative of a production environment. This contains large input datasets and generate large output datasets or reports. This also contains the most complex and largest data to create a stress test.
2. **Performance Metrics –** The FULLSTIMER option within SAS is applied so the elapsed time and CPU usage is documented in the log for each execution for accurate measurement.
3. **Acceptance Criteria –** The test must include a successful single run of all the programs. The elapsed time for the single run representing one user is recorded as a baseline and the same set is executed simultaneously in separate threads simulating multiple users in a real environment. The average elapsed time is recorded for the multi-thread execution. A comparison is evaluated with a pre-determined condition for acceptance. For example, if the group of SAS programmers is small, the multi-thread can contain five simultaneous sessions simulating five uses. In this case, an acceptable criterion would be that multithread elapsed time does not exceed five times the single thread elapsed time.

The execution of the programs for PQ can be tricky and time consuming since in order to have an accurate execution, you need to identify all the users and have them log on at the same time to assist in performing the test. You can also simulate this by creating a macro which executes the test script in multiple SAS sessions as different threads. This macro is named %INSTALLPQ and functions as a driver program initiating the timed execution of each programs. Once the single user simulation is finished, multiple execution of the same program is performed in separate threads. The macro parameters are described here (Truong, Smoak 2010).

```
%installpq  ( script = Performance Test Scripts,
              session = Number of Sessions,
              output = Output Documentation);
```

Where	Is Type...	And Represents...
scripts	C (40 chars)	SAS Test scripts that contain large input data and complex statistical analysis testing the limits of I/O and CPU usage of the system.
session	Numeric	Number of sessions that the Performance Qualification will be performed. This is a number ranging from 2 to 100.
output	C (200 chars)	The name of the output report file. This can be in either HTML or PDF formats. If none is specified, the default will be INSTALLPQ.HTML stored in the current directory.

In this example, the programs were executed in batch so the spawning of the simultaneous threads were accomplished through SAS X commands. This is a command within a SAS program that executes a command to the operating system. For Windows, it is like starting a SAS program in a BAT file or at a DOS prompt. The X command would start a separate SAS session executing the multi-user threads. If your programs share the same input dataset or output, running them at the same time can cause file contention as multiple programs try to update the same dataset. You can get around this problem by creating a separate set of programs and data for each thread. You can name the programs as the single user case and then followed by a number. So if the single thread program is named: aesum.sas, the multi-thread would be: aesum1.sas, aesum2.sas, aesum3.sas, etc…. The %installpq logic would need to execute the correct programs to prevent data contention (Truong, Smoak 2010).

The log files of the test script contain the results from the FULLSTIMER option. An example for the total elapsed time will appear at the bottom of the log as follows:

```
NOTE: PROCEDURE DISPLAY used (Total process time):
      real time           7.81 seconds
      cpu time            2.03 seconds
```

The %installpq macro would contain logic to parse all the resulting logs; apply a PROC MEANS on the time and present this in the final report containing the single thread time as compared to the averaged multithread time. This will be used in determining if it has met the acceptance criteria (Truong, Smoak 2010).

VALIDATING SAS MACROS

All validation efforts are applied according to the risk of the software being tested. SAS macros are intended to be used many times and thus incur more risk than a standalone (single-use) SAS program. Macros are not quite as pervasive as the SAS system itself which is described above. However, since SAS macros can be used on multiple projects, they deserve a good amount of testing to demonstrate that they are validated. The following steps describe validation processes taken to ensure that a SAS macro is adequately validated.

STEP 1 – Macro Documentation

The SAS system requires formal requirements and functional specification documentation. SAS macros are not as high risk as the SAS System but still require good documentation. This will be useful for users when it comes to applying the macro. From a validation standpoint, the documentation is also useful for knowing all the areas to be tested. The first step in the macro documentation will list the macro name and parameters in much the same way that the %INSTALLPQ macro is shown above in the "Performance Qualification" section. In addition to this, the documentation should also explain in plain English what the macro does and provide examples of macro calls. These two sections of the documentation are shown in an example below, the macro %REPLIBRARY which is part of a suite of tools named Trialex Toolkit designed for managing standard reporting library.

Details

This macro defines a system level dataset used to manage the reports library. This will keep track of which reports dataset is considered standard and can be used to be created as a template when a new study is defined. It has the ability to manage standards through a hierarchy of levels so that a set of report attributes defined at a higher level in the hierarchy will take president and override attributes applied in other reports at a lower level. Some of the prerequisites needed to be previously defined is listed here:

- The library name "GLOBLIB" must be defined to the location containing the Trialex Reports Library dataset.
- The LEV1 parameter must be defined since there must be at least one level.
- The LEV2-LEV5 parameters can be optionally defined but must have values continuously. This means for example that you cannot skip a level and define level 3 but have a missing level 2.
- If an existing library already exists in the specified "GLOBLIB", it will make a backup of the old library and create a new library.

Example

```
*** Define the library for global reports library dataset ***;
libname globlib "c:\path\to\report";
%replibrary (lev1 = Global);
%replibrary (lev1 = Global, lev2 = Project Level);
%replibrary (lev1 = Global, lev2 = Project, lev3 = Study);
```

The "Details" macro documentation section will explain what the macro does. In addition, it will document error conditions or unexpected situations that may occur when applying the macros. Each one of these conditions will be tested in the test script section below.

The examples are intended to show the user the basic syntax on how to call the macros with its parameters. It is not intended as an exhaustive set of all the different ways users can use the macro, but give common examples as to how the macro is to be used.

STEP 2 – Macro Test Protocol

Prior to performing the testing, a protocol is written to provide the members involved in testing with clear instructions on how to the macro testing. Each area can be short and concise. The protocol should include the following:

1. **Objective and Scope –** This will clearly state the goal of the testing such as to ensure the macro functions without errors and define the scope relating to the SAS system or other programs.
2. **Installation Instructions –** This may contain special instructions on the location of the macro and how SASAUTOS or other macro paths are configured to access this macro.
3. **Test Scripts –** This will list out each test script which is described in the next section. It will also include instructions on how to perform each test script and how to review the results.

The protocol details the expected outcome and how to distinguish between a successful validation test and one that is considered a deviation which would require reconciliation. The goal of the test protocol is to not leave any ambiguity as to how the assigned tester should perform the test scripts.

STEP 3 – Macro Test Scripts

The test scripts include two components. The first component include instructions on how to perform the testing and the second is the actual SAS program with macro calls which the tester can use as their test scripts during the performance. The main goal and purpose of the test script is to ensure that the macro is functioning as expected. Moreover, there are two general principles which the test script author should try to verify including the following:

1. **Invalid Conditions –** The invalid condition is partially listed in the documentation but there are others invalid conditions which the test script should try to include. Some example invalid conditions include:
    a. Required parameters should have missing parameters tested.
    b. Character parameters will include very long text exceeding expectations.
    c. Special characters including quotations should be tested.
    d. Out of range numeric values should be included.
    e. Prerequisite such as expected data is not available.
2. **Valid Condition –** The test should also include verifying when all parameters are valid that expected results are produced by the macro.

It may not be possible test all the possible conditions that the user comes up with related data that exist in the real world. However, the test script ensures that the most common errors are captured and expected results are produced from the test script.

The formality of the documentation and the level of details of the test scripts described in the three steps above may vary depending on existing SOPs or risks that each the macro poses. However, if the above steps were taken at a minimum, it will ensure that your SAS macros will function consistently and fulfill any regulatory audit review.

SAS MACROS INTERFACES

The most common way that users access SAS macros is through a SAS program which is processed using foundation or Base SAS. SAS macros are now being used in different ways with different interfaces allowing it to be used by different users who may not be SAS programmers. This is good from the standpoint of having the macro reach a new audience, but it does result in new validation challenges. Some new ways of using SAS macros include:

1. **Stored Process –** Macros can be executed in a stored process on a BI metadata server through EG.
2. **Web Report Studio –** It can be adapted into a report from Web Report Studio
3. **Mobile App –** It can be adapted to be executed on an iPhone App such as BI Flash.
4. **Web HTML5 –** It can have an HTML5 web interface running through SAS/IntrNet or CGI based web servers.

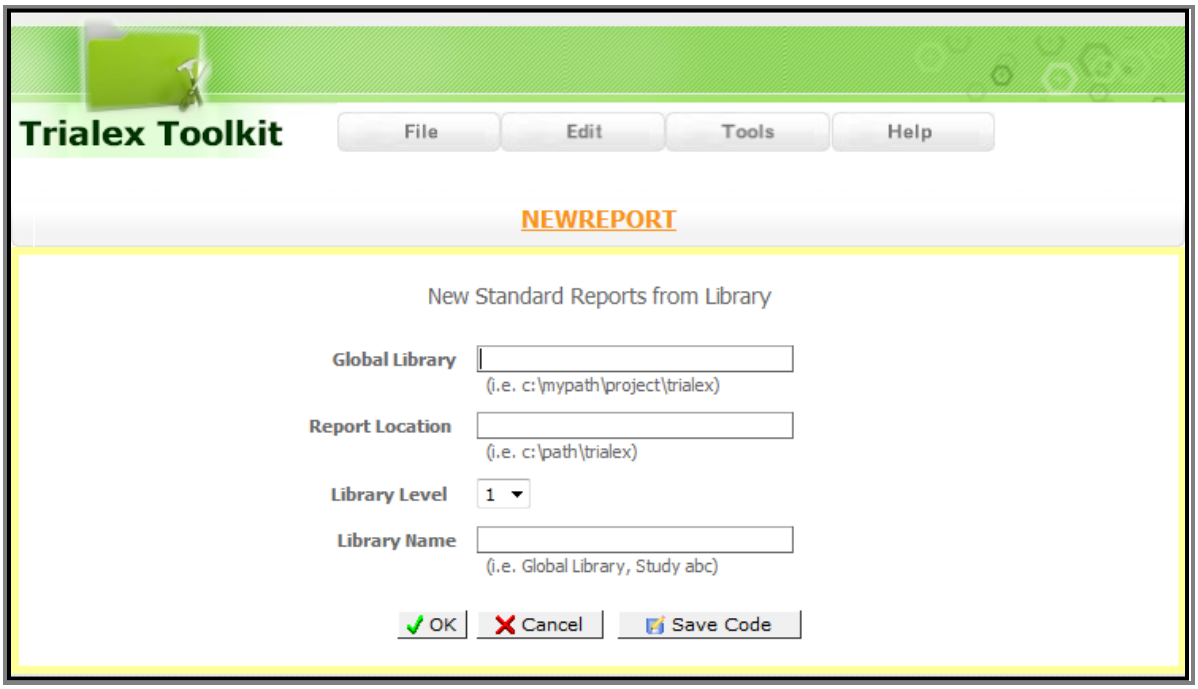
Each one of the above methods requires a different approach to validation. The core principles of evaluating valid and invalid conditions still apply. The following example illustrates the steps taken for validating a SAS macro named %NEWREPORT that has been converted to include a Web HTML5 interface.

The original %NEWREPORT macro has two parameters and two additional input options specifying where standard reports will be generated. The parameters are described here.

```
%newreport (level=level inside library,
            library=library name);
```

Where	Is Type...	And represents...
Globlib	LIBNAME	Library referencing the global library of standard report datasets.
Report	LIBNAME	Library referencing the reports dataset to be created.
Level	N	The level within the library that this new report will be derived from. This can be a numeric value between 1 and 5. If none is specified, the highest level defined in the library will be used.
Library	C	The name of the standard reports dataset defined in the library.

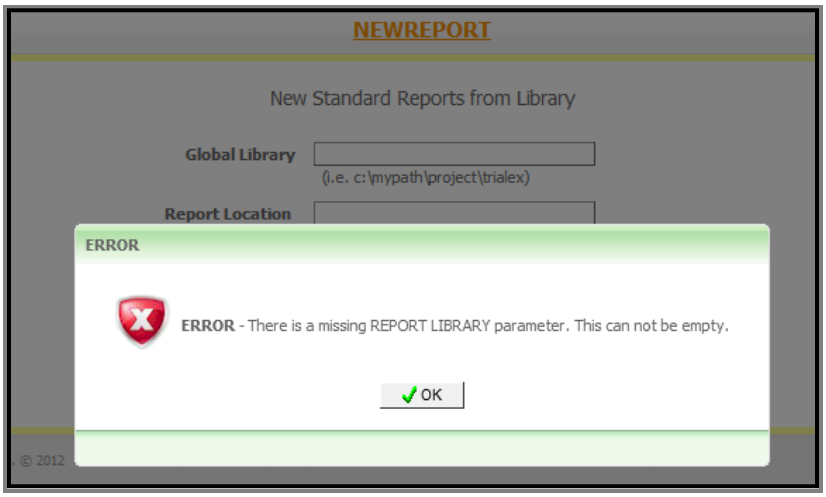
This macro will establish a standard library for reports to manage attributes of the report such as titles, footnotes. The global library where this study is managed is managed by the “Trialex Toolkit” which is a group of macros used to handle the hierarchy of standards such as titles at the project specific report. The difference between how the macro is used in a web interface is in how it selects the parameters from a web browser.



The traditional macro handles errors when it encounters an invalid parameter by generating an error message in the log such as:

```
ERROR: [newreport] was missing the required libname REPORT.
```

The web interface presents a similar display using HTML5 as shown here:



The logic for checking for the error condition may be the same, but the mechanics of communicating to the user is different between the web interface as compared to the message in the log. This also makes the validation process

different. The following steps describe how this macro is tested through a web interface as compared to the traditional macro call.

STEP 1 –Test Script Instructions

The test scripts that are part of the validation protocol can be very different between the interfaces. The script normally would require a re-write since the steps taken for the user in a web interface is very different compared to a macro call. Here is an example test script for the traditional macro call:

1. From within Windows Explorer, navigate to the folder containing the test program.
2. Right mouse click on the program V_NEWREPORT1.SAS and select batch submit.
3. Verify that the V_NEWREPORT1.LOG was generated after the SAS program completes running.
4. Open the log file in an editor such as NOTEPAD and verify that the following is displayed:

   ERROR: [newreport] was missing the required libname REPORT.

This same verification would be re-written to match the web version as shown here:

1. From within Internet Explorer web browser, log onto the website: http:;//myserver/trialex with the test user account: user=test, password=Test123
2. Navigate to the “New Report” tool through the pull down menu: Tools > New Report
3. Leave the entry values blank and click on the “OK” button.
4. Verify that a popup is displayed with the message:

   ERROR - There is a missing REPORT LIBRARY parameter. This cannot be empty.

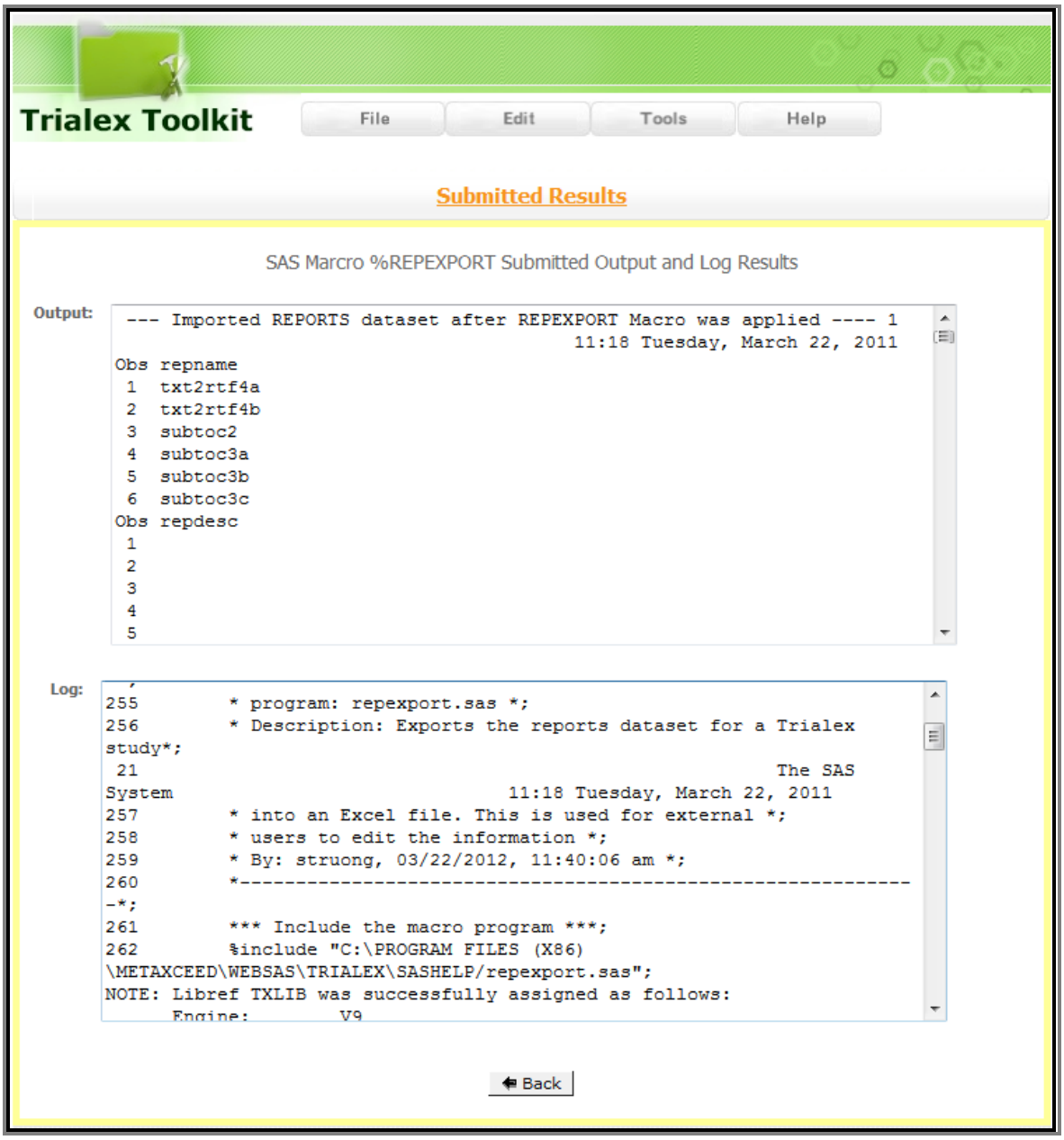
The purpose of the test is the same but the way the user interacts with the macro is entirely different. This means that any test script that was available for the macro would need to be re-written for the new web interface.

STEP 2 –Verifying Logs

A common step for the traditional macro testing is to open the log file and verify the ERROR message. The web interface also submits the same logic code and does have a similar log file created. The difference is that the location of the log file is specified by the process which the Broker CGI program is configured, so this part needs to be updated. Other than the location, verifying the underlying SAS log would be the same.

STEP 3 –Verifying Results

In a traditional macro, the result can be a LST file or an output SAS dataset. An interface such as the web would have the same result. The difference is how the web interface presents to the user the output file. In this example, the LST and LOG file is presented to the browser as shown below:



The traditional macro would have these files created at the location of where the test script program. This can also be the case for the web interface. The difference is how the user can access it and in the web interface, it is directly through the browser.

Validating SAS macros follows the same core principles of checking for valid and invalid values of its parameters and then review of the resulting output and log files. This remains the same with different interfaces to the macro. The way the test is conducted and the associated test scripts that accompany can vary greatly depending on how the different interfaces are implemented. The web interface in this example illustrates how user friendly a macro can be so it can be used by non technical SAS programmers. Because the interface is very different compared to how a programmer would execute the same macro, it also require a re-write of the test script and a new process of applying the validation test.

FILE MIGRATION

File migration is simply copying all files (SAS program, datasets, log files, list files and other associated output/files) from the old server to the new one and then verifying that the migration was successful. In theory this sounds like a simple task, but in actual practice it can be daunting for the following reasons. First, copying a large amount of data (hundreds of gigabytes) takes time. If anything goes wrong during the copying then all or part of the migration may have to be repeated. Second, the migration needs to be verified. This can be accomplished by generating a checksum for each file on the old server and comparing with a checksum for each file on the new server. The two checksums (for each file on the old and new servers) are compared and if the two checksums must agree before the migration is successful. This takes time since each file has to be compared (Truong, Smoak 2010).

Before doing the actual migration, it is a good idea to do a dry-run on a subset of the files. Doing a dry-run will give the person doing this task confidence that the actual migration will go smoothly (Truong, Smoak 2010).

TECHNICAL ISSUES

No matter how well planned the test protocols are defined, there will be technical glitches that are inevitable which needs to be added into the project timeline. The technical issues encountered in the SAS validation and third party software have some similarities and will be described together below. The migration had a separate set of issues that required a different set of solutions to resolve. There is no particular order or logic to the issues that occur, but an open minded and creative approach is needed to find an expedient solutions. Here are some examples of issues and their solutions (Truong, Smoak 2010).

SAS and Third Party Issues

1. **SAS/IQ/OQ Benchmarks –** The benchmark files that come with SAS is created at a particular time for the SAS build. A newer version of SAS based upon hot fixes and patches would lead to test cases failure. A review of the technical notes from SAS will confirm if the test is acceptable due these changes. In this case, the technical notes can be used to document the acceptance.
2. **Installation Order –** Many of the third party software are dependent on SAS, so some of the testing ran into errors when performed out of order. The protocol had to be clarified that the installation and testing needed to be done in a very specific order.
3. **Data Contention –** In initial PQ testing, there were collisions of files as multiple programs were trying to write to the same files. These were separated out into their own directories so there were no longer data collisions.

File Migration Issues

1. **Number of Files –** The sheer number of files created challenges in performing comparisons. A sampling will not catch all cases but it would be too time consuming to manually compare all files. SAS scripts were developed where possible so comparisons are automated and reports can summarize findings.
2. **File Name Special Characters** - Checksum was used to compare the files but an automated SAS script was used to perform the copy and comparisons. In some cases, the file names have special characters such as: ,|,<,>,?,*,&,/,% which crash the process. An update to programs using NBRQUOTE or SUPERQ helped on some cases and other cases were manually compared.
3. **File Permissions Special Characters –** A similar issue came about in a script that compared the user privileges among files. The same solution was applied in updating the script with NBRQUOTE or SUPERQ where possible and a manual comparison was done in other cases.

These are just examples since actual issues you will receive may differ. In the examples above, by developing scripts, it resolved one issue but it created others. In the end analysis, it was still more efficient to have developed the scripts to automate (Truong, Smoak 2010).

LESSONS LEARNED

There are multiple factors that create challenges when performing testing and validating the SAS system. The SAS software mixed with third party vendor software, combined with variations in the Performance Qualification test scripts, create an infinite number of possible issues that you can run into in performing a testing. It is important to have a clearly defined test plan and protocol, but you must also be flexible in the event that there are technical issues that require changes to the protocol. In these situations, it is more efficient to be adaptable to the situation. Some of the challenges require creative scripting solutions. The use of the operating system scripting languages may also be used, but the power of SAS is recommended as a method for creating scripts for automation. This can more easily be easily modified and adapted for other implementation that is on different operating system. Anticipate unexpected deviations and how to manage their resolutions. Effective communication between team members from different departments is essential. This can be accomplished by having clearly defined meetings with all key members. If

team members cannot make it in person; they can attend using Webex or similar screen sharing online meeting tools. A good version control of all validation documents storing MS Word, Excel and PDF formats is also recommended for the entire team. Many of these challenges and methods are not unique to validating SAS, but are issues involved in any complex technical endeavor that involves a team effort. If there are existing processes, tools and procedures that you already use to tackle complex projects with many moving parts, use what has worked before for your validation projects (Smoak, Truong 2011).

MANAGEMENT PERSPECTIVE

Managers of SAS programming are busy people and they often wear multiple hats. Dedicating the proper amount of time and resources to this type of task of validating a new server is important. Do not underestimate the amount of time and resources required (Smoak, Truong 2011).

Successful completion of the project requires interaction with people in different departments (e.g., IT and Quality) and their review and approval of the required documents. Having a weekly meeting with all the involved parties will help to keep the project moving.

Specific tasks which the manager may have to oversee include:

- Purchasing hardware and software
- Follow company validation standards
- Meeting with IT, Quality and other personnel on a regular basis
- Writing and execution of validation documents, such as:
    - Validation plan
    - User requirements and specifications
    - Change management
    - Traceability matrix
    - Test scripts
    - Validation Protocol (IQ/OQ/PQ)
    - Validation reports (IQ/OQ/PQ)
- File migration
- Start-up of the new SAS system

CONCLUSION

Validating the SAS system, migrating to a new SAS version and validating SAS macros is challenging due to the many dependent components relating to hardware, software and people. In such a complex system, if one component is broken, it can trigger a “butterfly effect” rippling throughout other components rendering the entire system nonfunctional. This concept is popularized by science fiction movies and chaos theory. Validating SAS is not as exciting as science fiction. Therefore, you have to keep a light hearted and open minded perspective in tackling the challenges and finding creative solutions. Performing the validation requires multidisciplinary set of skills that involves a collaborative effort from many team members ranging from IT, Quality, Project Managers, SAS Programmers and Statisticians. A balance of technical aptitude is required along with effective communications skills are needed to navigate the deviations that arises during validation. The examples in this paper illustrate methodologies and solutions for both the technical, managerial and communication approaches. Although this process requires resources and investment in time, the result is worthwhile resulting in a system that will function with integrity and meet regulatory standards and regulations.

REFERENCES

Smoak C, Truong S. “Managing the Validation and Migration from SAS® 9.13 to 9.2 on a New Server.” *Proceedings of the Pharmaceutical SAS Users Group*, May, 2011.

Truong S, Smoak C. Validating and Migrating SAS® 9.13 to 9.2. *Proceedings of the Western Users of SAS Software*, November 2010.

RECOMMENDED READING

Gilbert H, Light S. “Implementing SAS using Microsoft Windows Server and Remote Desktop.” *Proceedings of the Pharmaceutical SAS Users Group*, May 2006.

Helton ED, McNealy J, Halley P, Runde AS. "SAS® Validation in the Pharmaceutical Industry." *Proceedings of the Pharmaceutical SAS Users Group*, May 2003.

Kennedy GR. "Is it Harder for a Pharmaceutical Company to Move from SAS® V6.12 to V8.2 than it is to Qualify for the World Cup?" *Proceedings of the 27th SAS Users Group International*, April 2002.

Light s, Siegel A. "Compliance Readiness for the New Millennium: How does your SAS Environment Measure Up? Proceedings of the Pharmaceutical SAS Users Group, May 2000.

Mangold A. "Regulatory Compliant Validation and Deployment of SAS® Programs with the *Colibri* Server." *Proceedings of the Pharmaceutical Users Software Exchange*, October 2005.

Sporon-Fiedler G, Lassen M, Lundbeck H. "SAS® Coexistence with FDA 21 CFR Part 11, How Far Can We Get?" *Proceedings of the Pharmaceutical SAS Users Group*, May, 2002.

Truong S. "SAS System and SAS Program Validation Techniques." *Proceedings of the Western Users of SAS Software*, September 2009.

Woo W. "SAS® and VMWARE® to Create an Environment for Computer Systems Validation in a Pharmaceutical Company." *Proceedings of the 31st SAS Users Group International*, March 2006.

CONTACT INFORMATION

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

Sy Truong
MetaXceed, Inc. (MXI)
42978 Osgood Rd
Fremont, CA 94539-5627
tel: 510.979.9333
fax: 510.440.8301
E-mail: sy.truong@meta-x.com
Web: www.meta-x.com

Carey Smoak
Manager, SAS Programming
Roche Molecular Systems, Inc.
4300 Hacienda Drive
Pleasanton, CA 94588
Tel. 925-730-8033
Fax 925-225-0195
carey.smoak@roche.com