SAS System and SAS Program Validation Techniques

Sy Truong, Meta-Xceed, Inc., San Jose, CA

ABSTRACT

This course will teach methodologies of performing SAS system and SAS program validation including new installations and updating to a new version of SAS on a production server including SAS macros and programs. Some of the topics covered include: backward compatibility, SAS systems versus multi-use SAS macros, project specific and stand alone SAS programs and standard programs for portability. Validation of a SAS system most commonly occurs during an upgrade from an older version of SAS or moving to a new platform. The examples used in this course include migrating to SAS 9.2 and moving from a legacy operating system to the windows platform. In either case, 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 focus in on individual components. This allows you to obtain the scope and interconnectedness of each component so that your validation efforts are balanced and thorough. Once the system architecture is clearly understood, the requirements and functional specifications of each component are documented accordingly. These functional specifications then drive the validation testing.

INTRODUCTION

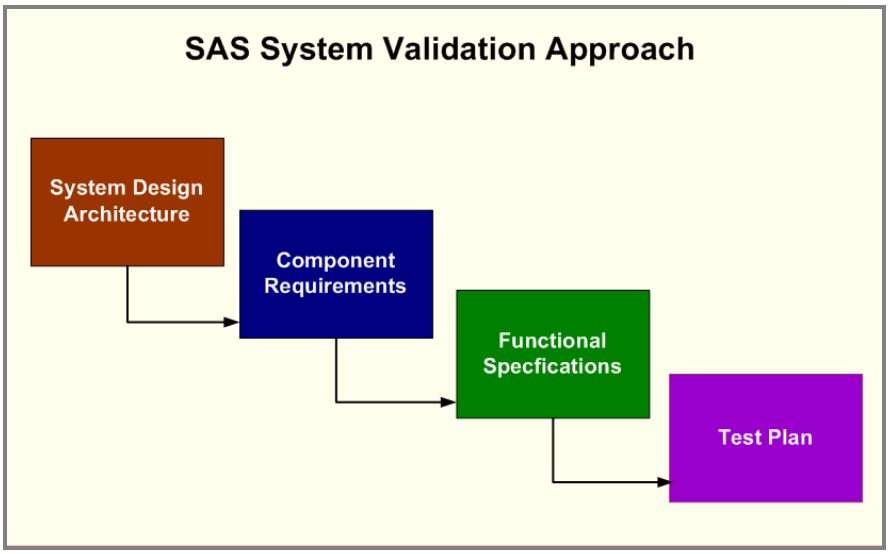
Validating a new version of SAS on a production server or a series of standard macros can be daunting task. The SAS System ships with user friendly installation qualification tools can assist in this process. This tool coupled with other tools and a well defined process makes it a little easier. Besides qualifying the installation and implementation, there are other tasks and components of the system that need to be validated or verified. Some of these components include:

1. Backward compatibility issues with older versions of datasets and format catalogs.
2. Validating multi use macros and standardized code templates.
3. Verifying stand alone or project specific programming and output.
4. Effects on Standard Operating Procedures (SOP) and programming practices.

The interconnectedness of the SAS computing environment does require considerable efforts in validating any addition or changes to the current environment. If the validation is executed successfully however, it can ensure that the new system has greater traceability between output, programs and source data. The performance qualification also sheds light on ways of optimizing the work and dataflow of your entire computing environment. The many benefits of performing validation of the SAS System will outweigh the effort and associated costs. In addition, validated systems are required within a regulated environment so it beneficial internally and when interfacing with an external collaborators or regulatory agencies.

It is important to follow validation steps in a systematic and orderly fashion since they are interdependent. Documentation of each step for instance, in the validation process is also essential in capturing and proving that the validation effort was done properly. Along with detail documentation, the capture the traceability of each validation task is essential for the reviewer. For each test case that is performed, there is an associated functional specification which then is connected to the requirements for a 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 therefore make the difference between a successful validation audit and a complete failure.

One of the main goals of the validation effort is to ensure that the installation and implementation of the SAS system and tools function as intended by the vendor (SAS Institute) or the author of the macro and your organization. The validation will ensure this success of this process. An example work flow for a validation process is shown below in this diagram.



ARCHITECHING COMPONENTS

The first step in your validation effort is to understand what it is that you are working with. The SAS System, as delivered to you in a series of CDs, is a system which contains modules such as Base, Stat, Graph and other components of SAS. This however only makes up part of the system that you are implementing in your organization. The SAS software fits into a computing environment that interacts with other software and hardware. If you were to take into account all the associated hardware and software that SAS interacts with, this is what is considered the "SAS System" from a validation perspective. It is therefore important for you to take the right steps to identifying and documenting all these components.

STEP 1 – HARDWEARE COMPUTING ENVIRONMENT

Identify all the hardware components of your computing environment. For example:

Hardware Component	Name
SAS Application Server	SASAPPSRV
SAS File Server	SASFILESRV
Client Desktops	CLIENTDSK

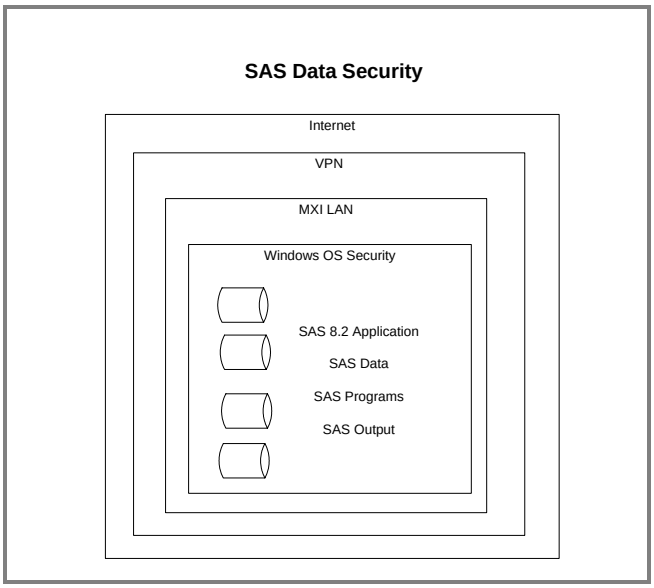
There could be many physical machines used or just a few. It is important to know where SAS is stored and which machine stores other files that relate to SAS.

STEP 2 – SECURITY ACCESS

Document security model implemented for your SAS environment.

Server Type	Security Levels
SAS Application Server	Windows Login
SAS File Server	Windows Login

There could be a software layer or physical layer to your security model.



It is common to have a combination of software and hardware contributing to your security model. By documenting it, you will see where there are holes or areas that may need improvements.

STEP 3 – SOFTWARE COMPONENTS

Identify all the software components that are installed on your hardware.

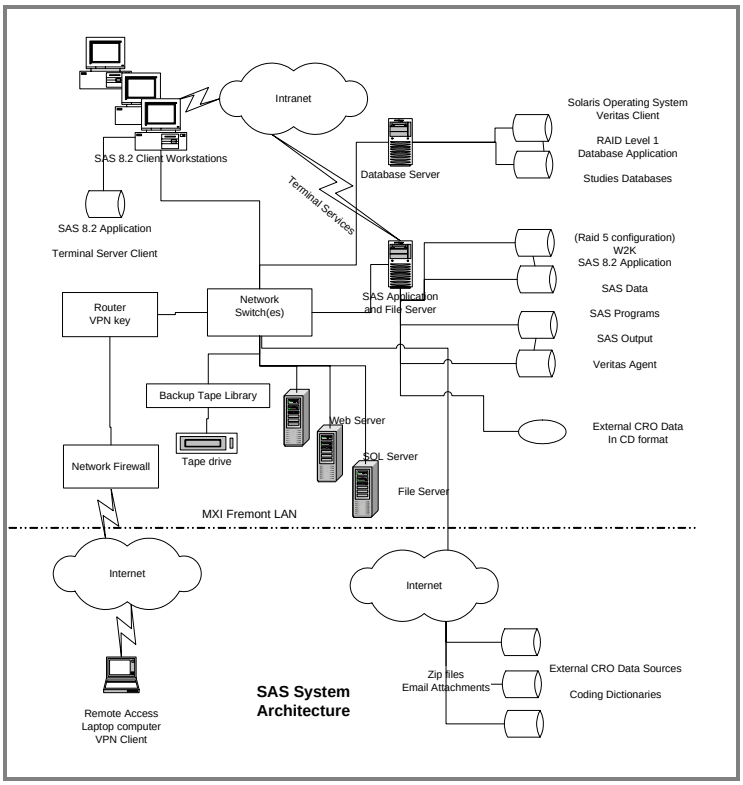
SAS Application Server

- SAS 9.1.3 Enterprise Bundle
- Microsoft Windows XP
- SAS System Viewer
- SyValidate version 3.0

This step is essential in understanding all the software used on your system. This step helps you perform the impact analysis of how SAS affects other components within your environment.

STEP 4 – SYSTEM ARCHITECTURE

Document the system architecture. This ties all the hardware and software together.



This gives you a bird's eye view of all your components and how they relate to each other. Only include those components that relate to the SAS system. This includes input to the SAS system such as source data bases and output reports or data that is produced by SAS. The above examples steps through the formal process of the SAS system which is rather complex and has potential for high risk. From a validation's perspective, a SAS macro or series of SAS program is less risk and would requires a abbreviated series of steps as compared to the formal described.

SYSTEM REQUIREMENTS

It is important to obtain all this software but you have to know what you want to do with it before it can be useful. If you are the end user, then you would pose the question to yourself.

What do I need SAS for to help me do my job?

See if you can answer this question with one concise sentence. This short answer will be the theme that you will use in capturing your requirements. If you are not the end user, then set up interviews with the end users and ask them the same questions. During this requirement collection stage, distinguish between what the user is requiring and what the software vendor (SAS Institute) is requiring. For example, the system must run on Windows XP Professional rather than Windows XP Home. This may be something that would be nice for the users, but it is more of a vendor requirement. This distinction will help you better organize your requirements.

STEP 1 – VENDORS' REQUIREMENTS

Identify all the vendor's (SAS Institute) minimum requirements. This has to do with the minimum memory or hard disk requirements for your particular hardware and operating system configuration. This can be different between your SAS server versus data and client machines.

STEP 2 – INTERNAL ORGANIZATION REQUIREMENTS

Document your organization's requirements pertaining to security and computing environments. Your organization may have backup facilities or there are password and data access stipulations. These requirements are usually established by your organization and can be different from the vendor's requirements or the specific user's requirements.

STEP 3 – USER REQUIREMENTS

Document the users' wishes and requirements. An example would be that SAS generate reports and perform specified statistical analysis. Some user requirements may relate to step 1 and step 2. For example, the user may want the system to be fast and available 24x7. Sometimes the requirements do crossover. It is not essential that you capture every little requirement but that the most important and essential features are captured.

The requirement steps can be laborious and not very exciting. However, it will give you great insight in to what are the core business needs of your organization as it relates to the use of SAS. This understanding is essential in focusing your validation efforts to ensure the highest degree of your system's effectiveness and integrity.

FUNCTIONAL SPECIFICATIONS

Now that you know what it is that you need SAS to do, the next step is to understand how that is accomplished. For example, if one of your requirements is for SAS to generate reports in PDF format, the functional specification corresponding to this requirement would detail the use of PROC REPORT and associated ODS options to produce the required report. The SAS System is very extensive and delivers most of user's requirements. However, there are instances where users have requirements that go beyond what is available from SAS. Here are a couple of sample requirements that may not be delivered by Base SAS.

1. **SAS Program Change Control** – SAS programs that are edited and updated need to be versioned to form a complete audit trail.
2. **Report Table of Contents** – All SAS listings and table output need to be documented and organized in a table of contents.

In this example, the functional specification will detail the use of tools supplied by external vendors such as MXI's SyValidate and Trialex tools. In this example, the corresponding functional specification would be:

1. SyValidate is used to make a backup copy of each SAS program each time it was submitted. This includes recording the user name and time in which it was captured. Optionally, the tool will capture a user note explaining the changes. Any SAS program can be rolled back to an older version. Reports can also be generated to detail a complete audit trail for a comprehensive change control of SAS programs
2. The Trialex tool is used to organize all report titles and associated footnotes in a central database. A table of contents can be automatically generated in RTF, HTML and PDF format including hyperlinks to each specified report.

Your group may have a macro or other utilities that are analogous to these tools. In this example, the functional specification provides a little more detail on how a particular requirement is accomplished. If you are the end user, the documentation of the functional specification will answer the question whether the requirement is met. If you are not the end user, you would need to present the functional specification to the user and ensure that it fulfils the original requested requirements before proceeding to the next step.

TEST PLAN PROTOCOL

The last major step in your validation process is creating the test plan and executing the tests. This will confirm that each functional specification that you have identified truly does work. There are many formats in which you can organize your test plan. The following are suggested steps that need to be documented.

STEP 1 – VALIDATION ROLES

Identify the role and who will be responsible for performing testing related tasks.

Qualification and Testing Activity	Responsibility Assignment
Protocol Generation	Validation Specialist or Contractor
Protocol Approval	System Owner Validation Quality Assurance
Protocol Execution	System Owner or Contractor or System Administrator
Protocol Execution Review	Validation

Qualification and Testing Activity	Responsibility Assignment
Validation Protocol Deviation Review and Approval	Validation Quality Assurance
Qualification Final Report Generation	Validation Specialist or Contractor
Qualification Final Report Approval	System Owner Validation Quality Assurance

It is not required that specific names be detailed at this stage, but rather the job title or description will suffice. This is more flexible in that the same person can do more than one job or there may be reassignments once the testing starts.

STEP 2 – IMPLEMENTATION PLAN

This step involves documenting the implementation plan. This is where you specify the category of tasks that need to be performed in order for the testing to be completed successfully. An example of some of the steps includes:

1. **Validation Objective** - The objective of this qualification effort is to qualify the installation and functionality of the SAS system.
2. **Activities**
    - **a.** Installation Qualification Testing
        - **i.** Verification of the hardware and software components
        - **ii.** Verification that the servers are properly secured
    - **b.** Performance Qualification Test
        - **i.** Verification procedures are put in place for the operation of the system****
        - **ii.** Backup and recovery procedures are verified
    - **c.** Acceptance Criteria
        - **i.** Demonstrate and document that system requirements have been satisfied
        - **ii.** The system is installed according to the approved qualification protocols
3. **Change Control**
    - **a.** Version management and configuration management
    - **b.** Revalidation procedures and associated triggers for revalidation

These are just highlighted areas that are to be considered a complete list. There is no exact one template that will work for all conditions but these steps are a good start as suggested guidelines.

STEP 3 – TESTING TYPES

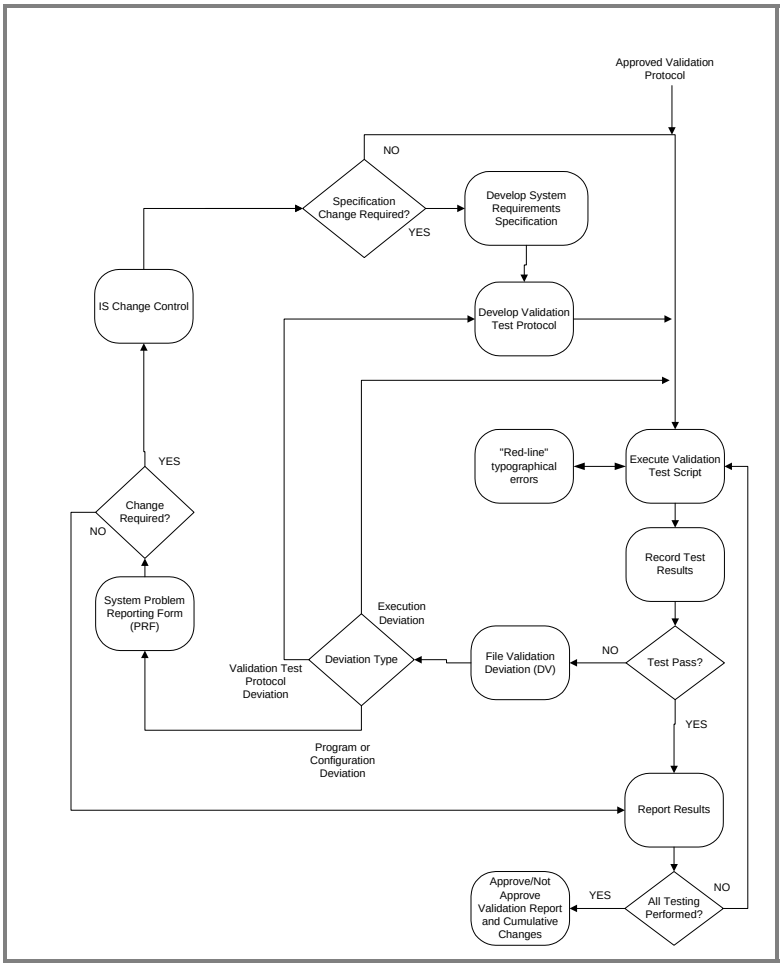
This step involves defining the types of testing that need to be done. There are several different types of qualification testing. You would need to perform all of these to be thorough.

1. **Installation Qualification (IQ)** – This ensures that the installation of SAS and related tools was done according to the vendor's expectation. It usually verifies if all the files and components have been installed and are operational.
2. **Operational Qualification (OQ)** – This ensures that the system is operating with all of the intended features. This includes all the functions detailed in the functional specifications.

3. **Performance Qualification (PQ)** – This ensures that the system performs within the size data and amount of users that would be used in the real world. This will simulate real use to see if the system performs according to expectations.

STEP 4 – TESTING PROCEDURES

Testing flow charts can really clarify how testing is to be conducted. This documents the order of the testing steps and how the test is to be handled in the event of deviations.



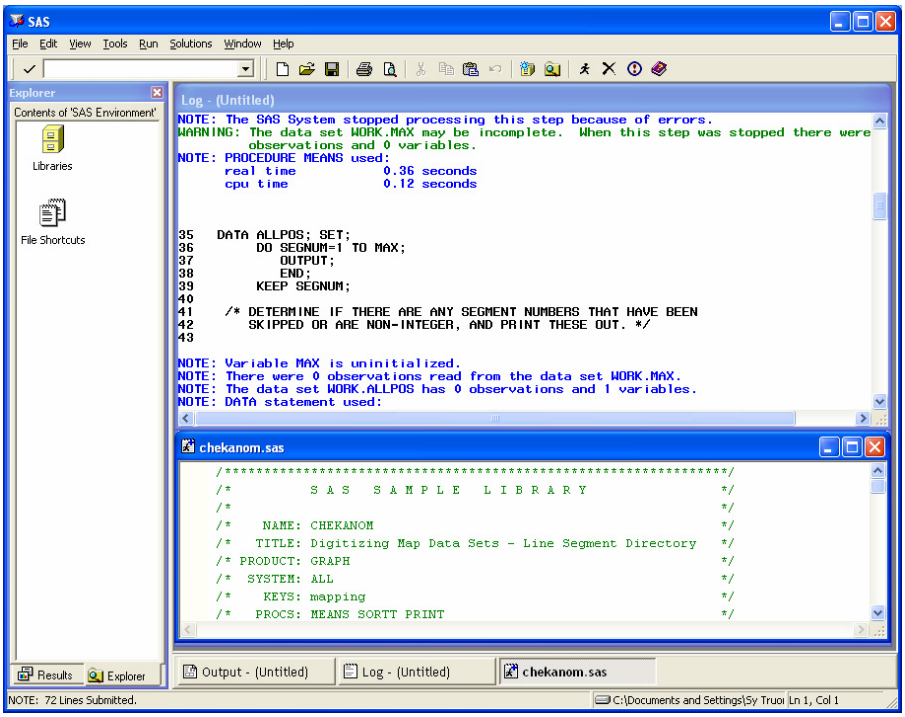
Use diagrams wherever possible to visually clarify how testing is to be conducted. This is more useful than long list of text procedures.

SAS PROGRAMS

The SAS system is comprised of many components that are delivered from SAS Institute. Programs or SAS macros that users develop are also an important component. Validating SAS programs presents some unique challenges especially when working within a regulated environment such as the pharmaceutical industry. SAS programmers in your team that develop these programs come from many different backgrounds that can range from biology to statistics. The majority are not from a computer science background. This is normally due to the fact that SAS programmers have expertise in the domain of the data in which they are analyzing. This is helpful for ensuring the outcome of the analyses but creates challenges by creating an unstructured programming environment. The work flow is driven by reports and therefore is usually done in an ad hoc manner. The analysts normally get mockups of the report which describe what they need to produce. SAS programmers often jump right into programming with little or no programming design considerations. SAS has adapted to this work flow as compared to other more structured high level languages. Other languages such as C or Java are stronger typed. This means that the variables and data tables have to be defined with proper variable type and length before they can be used. On the other hand, SAS

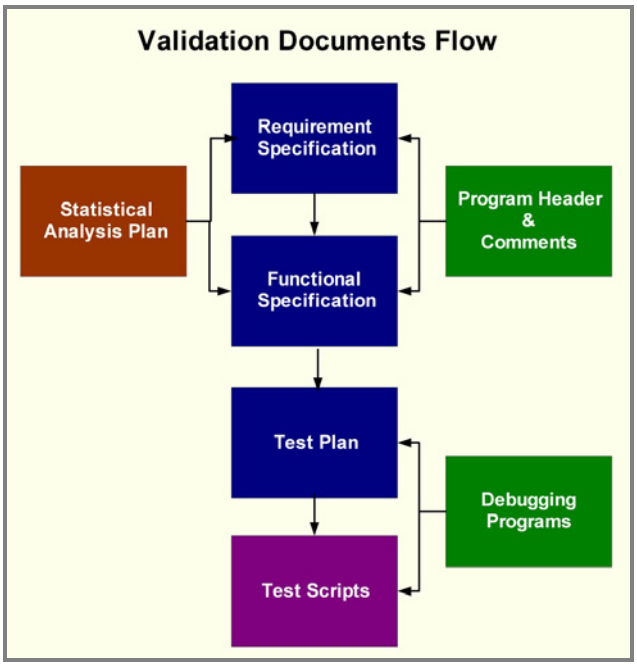
programs can dynamically create variables as you go along, lending itself to the ad hoc nature of the development process. This can be beneficial for creating exploratory analysis and conducting experiments with the data. However, it fosters software development that is riddled with maintenance challenges.

The tools used to develop SAS programs, such as display managers or text editors, are further examples of the ad hoc nature. Display manager gives some structure, but it is designed for exploration. Software development tools for other languages allow for the programmer to manage the source code as it relates to other programs and data. On the other hand, SAS programs are ASCII text files that any user can edit with a text editor of their choice. In a similar way, display manager stores programs on disk as files and does not create any other structure upon that.



VALIDATING PROGRAMS

Similar to validating the SAS System itself, SAS programs needs an organized methodology and approach for validation. One of the first steps in validating programs is to have a test plan. The test plan is derived from a list of functional specifications. The functional specifications are, in turn, driven by the list of requirements. This is an interrelated set of documents that drive the process.



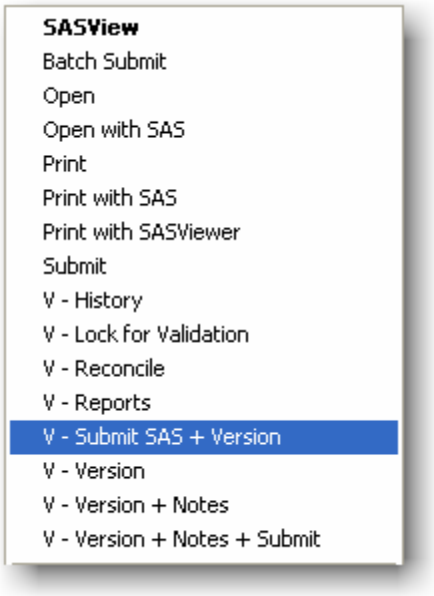
The level of detail of documentation depends on the complexity of the set of programs or macros being validated. The following levels distinguish the differing types of SAS programs.

- **Exploratory** - These are random sets of programs developed by analysts and statisticians to test out a hypothesis. They are not included in the final analysis or part of a submission.
- **Stand Alone** - These are one of the programs developed to generate specific reports or analysis files. They may be driver programs that call other macros but these driver programs are not used multiple times.
- **Multi-Use** - These are usually macro code or standard code segments that are used multiple times in more than one analysis. They can be stored at a global library where multiple users can access them.

It may be worthwhile to track exploratory programs, but they are not usually included in formal validation documents. However, it is recommended that both stand alone and multi-use programs are included. The multi-use programs deserve more detail and formal documentation, while the stand alone programs can be abbreviated to fit the complexity of the specific program.

AUTOMATED VALIDATION TASKS

One of the most common ways that a SAS programmer interacts with programs is submitting them. This is a good time point to capture information pertaining to the SAS programs. From Windows Explorer, a user submits a program by right mouse clicking on the program and selecting "Batch Submit".

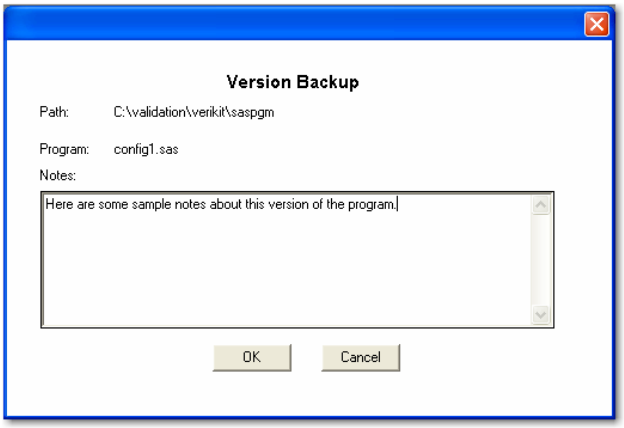


Additional information can be automatically captured along with submitting In this step through the use of the SyValidate tool. Among the information it captures include:

1. **Backup** - Make a backup copy of the code stored in the current SAS program.
2. **User Name** - Capture the current user name from the operating system.
3. **Date Time** - Capture the date time at the moment of the transaction.
4. **Action** - Identify what type of action is being performed. This is usually a well defined task defined in the validation process. Some examples include: version backup, locking for testing, validation testing, promoting to production.
5. **Status** - A status associated with the SAS program to identify if the validation testing has failed or passed.

After capturing and recording this information, the next step is to submit the program in the same way that the "Batch Submit" did before. In this example, the process captures many of the required tasks for maintaining an audit trail of the program without any additional effort from the user.

It is more realistic that only pivotal changes in the code would require a version backup. For smaller edits to the program, users would still use the "Batch Submit" selection. Once they decided that the code had changed significantly from the last time a version was captured, they would then choose the "V - Submit SAS + Version" menu item. On some of these code changes, a note describing the change is required to add meaning to the audit trail. In this case, another menu item "V - Version + Notes" is selected.

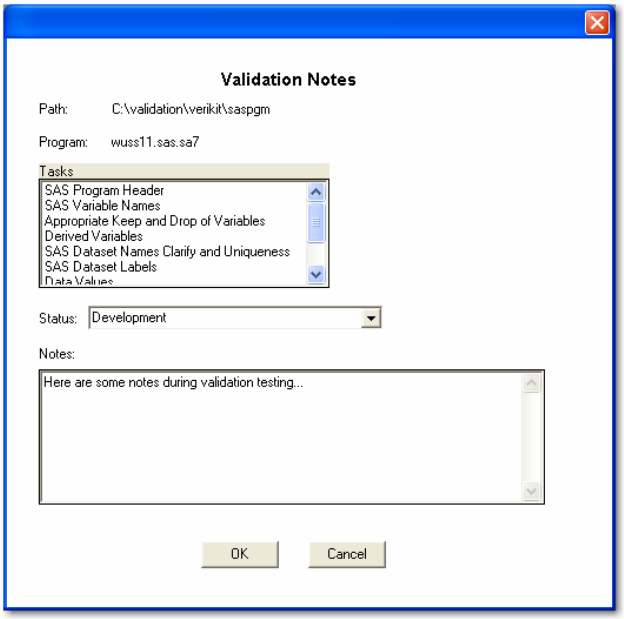


This step would capture all the information as the previous example including: backup program code, program name, user name, action, task, and date time. In addition, a short note can be entered describing the current code or logic change.

The features of creating a backup and capturing a descriptive note can be used during any type of SAS program development. However, in order to integrate this into the validation process, we need a mechanism that would lock the program for performing validation testing.

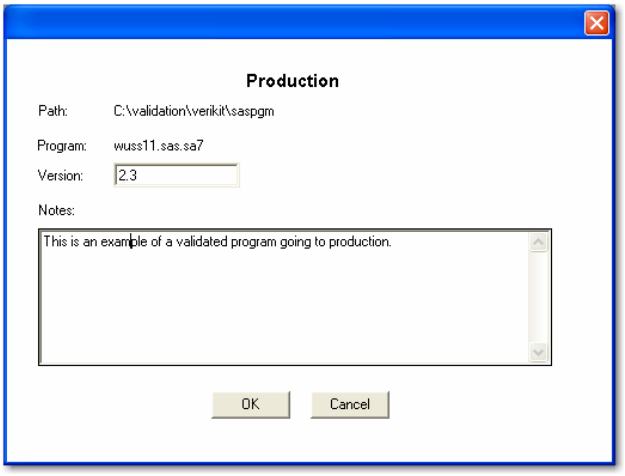
The process involves a verifier, who is a different person from the original author of the SAS program, to review the program and associated output and data. The verification may even include developing another independent SAS program to come up with the same results as a way of verifying. In this case, the verifying program can use the same versioning technique for a complete audit trail. During verification, however, it makes sense to lock the original code since we do not want to be verifying a moving target. When the user initiates the validation process by selecting the menu "V - Validation", a copy is made but it also changes the file extension which makes it clear that this is no longer a program to be edited.

Upon completion of verification, the verifier can record findings by right mouse clicking on the locked program and selecting "V - Notes".



This allows the verifier to record specifically which verification tasks were performed and if the testing was successful or not. A status is recorded to determine what is to be done. If it failed, then the original programmer has to fix the problem and the verifier goes through the loop again. If it passes, it can be promoted directly to production. At each step of the way, information is captured including a descriptive note which gives context to the task at hand.

Upon promotion to production, the programmer can choose to assign a version number. This can follow the decimal conventions such as version 1.0, 1.1, 1.2 etc…



If the verifier promoted a program directly from the verification process with the selection of "Verification Passed + Production", this will increment the version number by one integer value automatically. By performing the promotion in a separate step, the user can increment the version number to a custom value.

If during verification, problems were identified, the original programmer will need to unlock the program to perform the fix. This is available through the menu item "V - Unlock". SyValidate allows the user to record a note pertaining to the unlocking and then it renames the file back with the (.sas) file extension for further edits. Throughout the process, all this information is centrally recorded and reports can then be generated to document the validation process. Rather than manually typing these reports, SyValidate generates these reports for you so that the process of validating SAS programs can be documented with much more detail and is a less time consuming and laborious process.

CONCLUSION

Validation is a significant aspect of any SAS program development process. However, during the process of performing validation tasks, you may not appreciate what impact it is having. It may seem dry and laborious. In practice, however, you will find that when the test scripts are being created according to the test plan, you will discover issues pertaining to backward compatibility of SAS datasets and catalogs if you are upgrading. You will also find subtle errors and, in some cases, major bugs within your commonly used multi use macros. The performance qualification can also encompass the use of stand alone study specific code. In many instances, this reveals discrepancies that can lead to improvements to your system. The validation process can also solidify and help more clearly define the standard operating procedures and good programming practices of your group. If you can step out of the thinking of having to perform validation grudgingly and see it from a larger perspective, it can have a profound effect on your entire organization in how you and your users interact with SAS and its related data to fulfill your business objectives.

CONTACT INFORMATION

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

Sy Truong
MetaXceed, Inc. (MXI)
2185 Oakland Rd.
San Jose, CA 95131
Phone: 408-955-9333
Fax: 408-955-9307
E-mail: sy.truong@meta-x.com
Web: www.meta-x.com