Can Change Control be as Easy as a Right Mouse Click?

Sy Truong, Meta-Xceed, Inc., Milpitas, CA
Rory DeShano, M.A., MedImmune, Inc., Gaithersburg, MD

ABSTRACT

Change control functions as the fundamental process for quality control and is the foundation for validation. This is a standard procedure for software development but is lacking in the development of analysis when programming in SAS. The unstructured adhoc nature of exploration and discovery during the analyzing and reporting of data create challenges from the perspective of change control and validation. However, it is this data driven dynamic programming environment that requires the rigors of change control to ensure the integrity of the results. This paper explores processes and techniques used to manage change control of SAS programs and the output they produce within the process of development and validation. Some of the examples it will cover include:

- **Version Control –** Backward compatibility issues with older versions of SAS programs and output
- **Understanding Differences –** Comparing differences between versions of programs and output
- **Noted Changes –** Capture metadata or information describing the meaning and context to the changes
- **Validation Process –** Managing changes from requirements through the testing of all programs

This paper will demonstrate the challenges faced with managing change control of SAS programs in a regulated industry. In the recent history of computing, the use of the mouse and its right mouse click has made it easier to access context information upon objects within a user friendly graphical interface. This approach described in this paper will simplify and automate the process of managing change control of SAS programs.

VERSION CONTROL

Version control plays an essential role in any SAS programming environment. Not only does it assist the user in keeping track of the life cycle of the program, but it can also supply a quick reference point for any additional modifications the user may have during updates to existing programs. However, the life cycle of a SAS program and associated output documents are only meaningful when proper change control information is captured. This information or metadata includes but is not limited to the following: programmers name, date, time and any relevant comments defining the program characteristics and/or any modifications to these characteristics. This poses some fundamental questions:

1. Is there a solution that allows the user to version their programs?
2. Without having to constantly be checking them in and out?
3. Does it provide adequate comparative differences from one version to the next?
4. Does it allow users to roll old versions forward?
5. Does it have the ability to add notes quickly and efficiently and generate swift reports on the life cycle of any given program?

The following example steps describe an approach that addresses some of these questions.

STEP 1: **VERSION SAS PROGRAMS –** Through the use of Sy/Validate™ software, the user can right mouse click on a SAS program with a file extension of (.sas) from within Windows Explorer. A pop-up menu is presented with selections to perform backups or review the history. This is similar to submitting a SAS program in batch mode so it adheres to the workflow of the user.

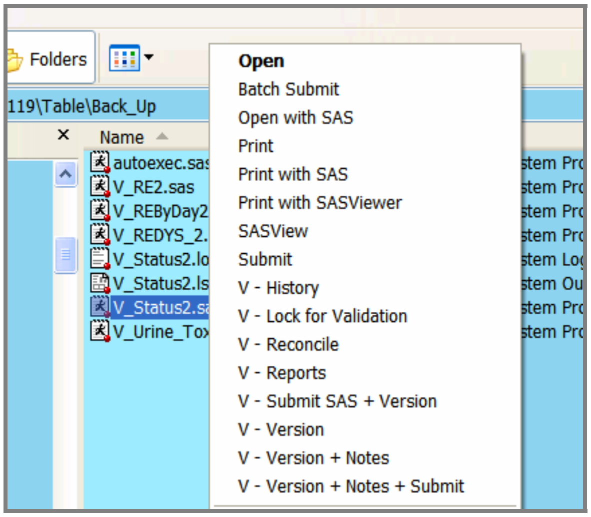


figure 1

This is demonstrated through the "V" menus which clearly distinguishes these functions from other items on the menu such as the “Batch Submit”. Since it is part of the same menu, it is familiar to the user and integrates well into the user’s work process.

STEP 2: **VERSION SAS OUTPUT–** Similar to performing backups to SAS programs, a version of the SAS output can also be accomplished through the right mouse click. The distinction between output files and SAS programs is that for the files such as Word documents, the file extension is either (.rtf or .doc), or Excel (.xls), or PDF (.pdf).

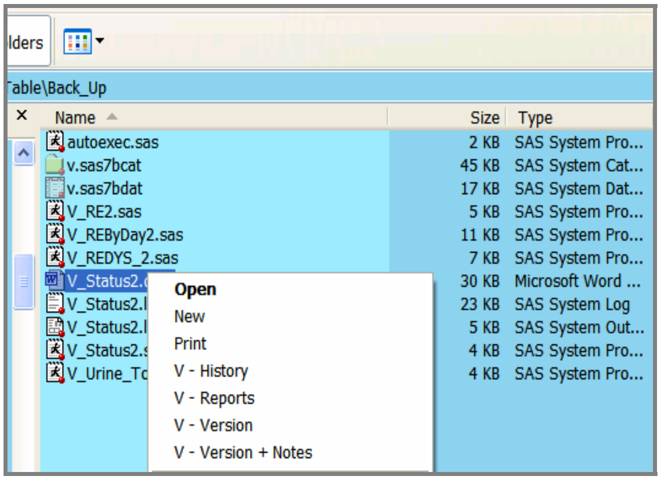
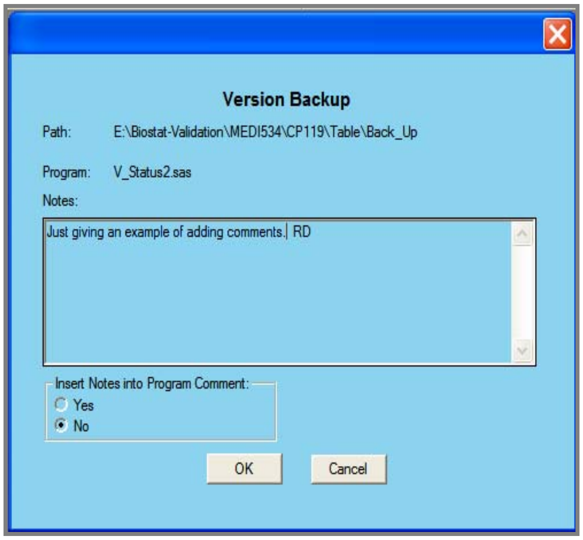


figure 2

This example is similar to the SAS programs but with just slightly different options through the “V“ menu items.

STEP 3: **ADD COMMENTS TO VERSIONS –** The file being versioned can retain additional meaning if comments are added explaining the reason for the change. This can be accomplished by adding notes which act as additional metadata so that the history has contextual meaning for each version.



Version Backup

Path: E:\Biostat-Validation\MEDI534\CP119\Table\Back_Up

Program: V_Status2.sas

Notes:

Just giving an example of adding comments. RD

Insert Notes into Program Comment:
○ Yes
● No

figure 3

In this example, the text added to the "notes" has the option to also automatically insert the same note to the program header. The notes are therefore analogous to the same information that users add to the program header explaining the significance of the update. This therefore provides greater meaning from a historical perspective. This approach adds value when it comes to reviewing and deciding which old version is the correct version to be restored in the event of corruption.

The three steps demonstrated above illustrate a solution and process which integrate into the work flow of a SAS programmer. SAS programming by nature is more exploratory due to its analytical and data oriented use. The users are therefore not used to the check-in check-out process of other change control systems. This approach of right mouse click is similar to how SAS programmers would submit their programs so it fits into how users would interact with their programs.

UNDERSTANDING DIFFERENCES

The steps taken during versioning of the programs can help the user better understand the history of the program. Some of the information captured that contributes to the historical knowledge includes the following.

Key Advantage	Description
Notes	As mentioned in step 3 above, adding notes pertaining to the version adds key information that explains the significance of the backup. This can explain things such as important algorithms or key logic changes in the program or file.
Program Name	The program name is used to identify the file as compared to other programs being versioned.
User Name and Date Time	The user that is logged onto the system will have the user name captured along with the date time stamp of the version.
File Contents	The program code or the output file contents are captured and maintained to obtain a complete audit trail.

The metadata captured will provide key information towards understanding the complete picture of the history for the file. Besides understanding information pertaining to each version, the information about the differences among the versions is also important. This will help determine the correct version to roll back to in the event that the current version is corrupted. The following steps are taken to illustrate this process.

STEP 1: **VIEW THE HISTORY –** The user would right mouse click on the program in Windows Explorer and select the history option "V-History".

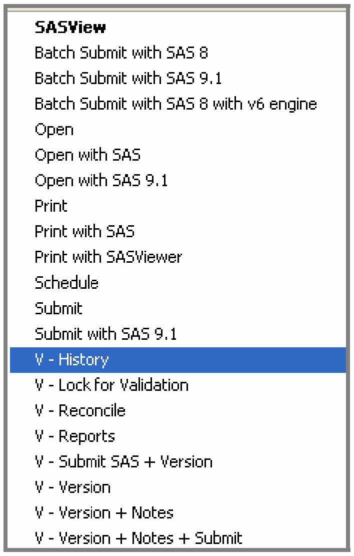
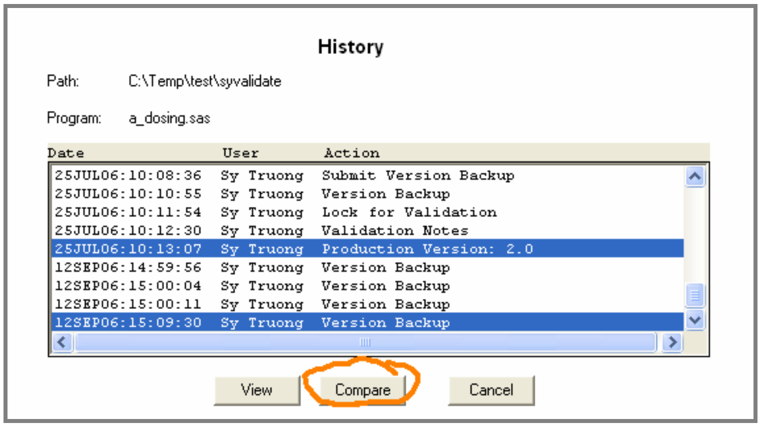


figure 4

STEP 2: **SELECT TWO VERSIONS –** In order to compare the differences, the user then selects two different versions of the program within the history. It is common that the latest version is then compared with one of the last production releases of a program.



Path: C:\Temp\test\syvalidate

Program: a_dosing.sas

```
Date                User        Action
25JUL06:10:08:36    Sy Truong   Submit Version Backup
25JUL06:10:10:55    Sy Truong   Version Backup
25JUL06:10:11:54    Sy Truong   Lock for Validation
25JUL06:10:12:30    Sy Truong   Validation Notes
25JUL06:10:13:07    Sy Truong   Production Version: 2.0
12SEP06:14:59:56    Sy Truong   Version Backup
12SEP06:15:00:04    Sy Truong   Version Backup
12SEP06:15:00:11    Sy Truong   Version Backup
12SEP06:15:09:30    Sy Truong   Version Backup
```

figure 5

STEP 3: **COMPARE –** A comparison is made by clicking on the compare option. This will then perform a comparison with differences highlighted. The left column within the comparison highlights within a vertical bar to indicate the area in the file that contains differences as shown in figure 6.

If you were to scroll down to the view, the actual differences between the two files will be highlighted in color to help you see the differences.

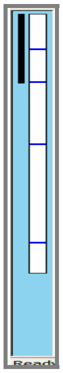
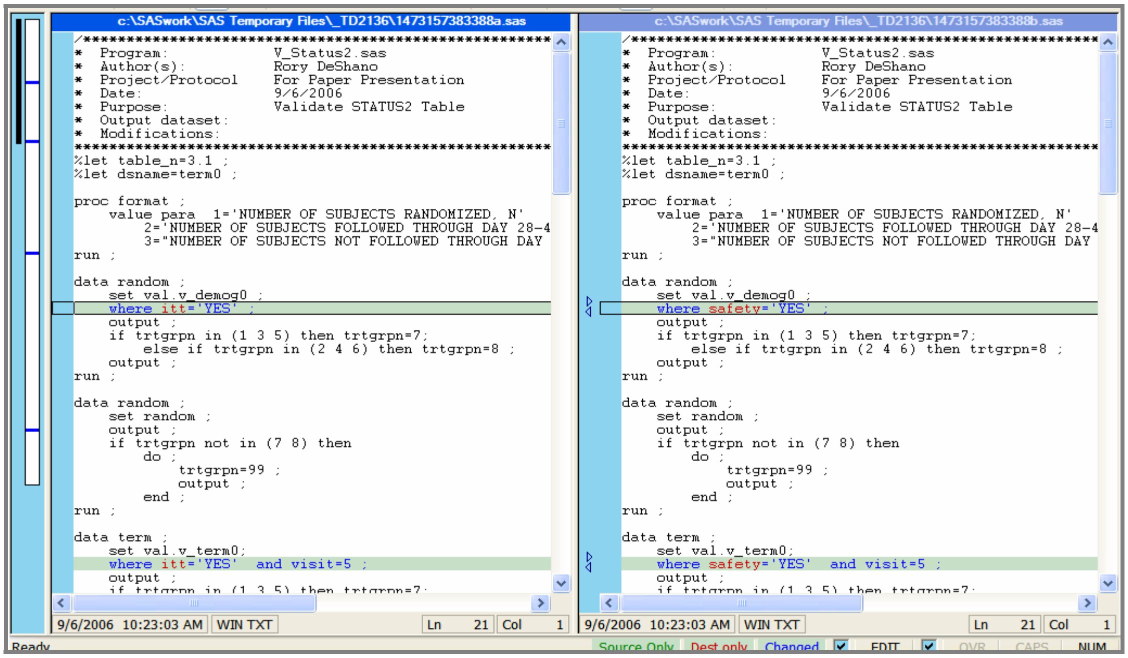


figure 6



```
/*****************************************************************
*  Program:                  V_Status2.sas
*  Author(s):                Rory DeShano
*  Project/Protocol          For Paper Presentation
*  Date:                     9/6/2006
*  Purpose:                  Validate STATUS2 Table
*  Output dataset:
*  Modifications:
******************************************************************
%let table_n=3.1 ;
%let dsname=term0 ;

proc format ;
    value para  1='NUMBER OF SUBJECTS RANDOMIZED, N'
        2='NUMBER OF SUBJECTS FOLLOWED THROUGH DAY 28-4
        3="NUMBER OF SUBJECTS NOT FOLLOWED THROUGH DAY
run ;

data random ;
    set val.v_demog0 ;
    where itt='YES' ;
    output ;
    if trtgrpn in (1 3 5) then trtgrpn=7;
        else if trtgrpn in (2 4 6) then trtgrpn=8 ;
    output ;
run ;

data random ;
    set random ;
    output ;
    if trtgrpn not in (7 8) then
        do ;
            trtgrpn=99 ;
            output ;
        end ;
run ;

data term ;
    set val.v_term0;
    where itt='YES'  and visit=5 ;
    output ;
    if trtgrpn in (1 3 5) then trtgrpn=7;
```

```
/*****************************************************************
*  Program:                  V_Status2.sas
*  Author(s):                Rory DeShano
*  Project/Protocol          For Paper Presentation
*  Date:                     9/6/2006
*  Purpose:                  Validate STATUS2 Table
*  Output dataset:
*  Modifications:
******************************************************************
%let table_n=3.1 ;
%let dsname=term0 ;

proc format ;
    value para  1='NUMBER OF SUBJECTS RANDOMIZED, N'
        2='NUMBER OF SUBJECTS FOLLOWED THROUGH DAY 28-4
        3="NUMBER OF SUBJECTS NOT FOLLOWED THROUGH DAY
run ;

data random ;
    set val.v_demog0 ;
    where safety='YES' ;
    output ;
    if trtgrpn in (1 3 5) then trtgrpn=7;
        else if trtgrpn in (2 4 6) then trtgrpn=8 ;
    output ;
run ;

data random ;
    set random ;
    output ;
    if trtgrpn not in (7 8) then
        do ;
            trtgrpn=99 ;
            output ;
        end ;
run ;

data term ;
    set val.v_term0;
    where safety='YES'  and visit=5 ;
    output ;
    if trtgrpn in (1 3 5) then trtgrpn=7;
```

figure 7

This color coded visual display makes it much easier to quickly identify differences between the program code. It therefore enables the user to easily identify the version that has the correct code in the event that one of the selected versions is corrupted.

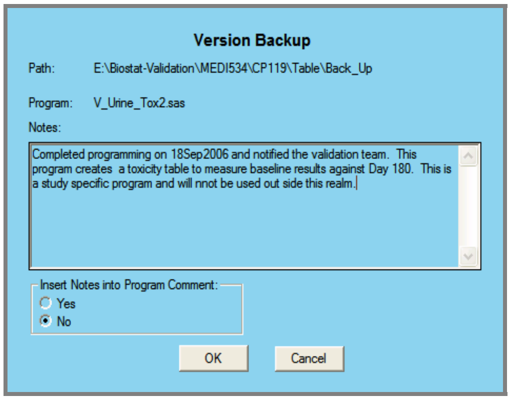
VALIDATION PROCESS

The validation process for SAS programs and associated output can be resource intensive and contain elaborate procedures. There are many considerations but it is important to have a process implemented which is closely integrated with the work flow of a typical SAS programmer. This will provide a process that is less painful and more efficient for users. One of the goals of validation is to develop quality programs which produce accurate output with integrity. Some of the components that go into an effective validation process include:

- **Risk Assessment** – An evaluation of the impact to the data from the programs including output and other associated resources.
- **Independent Review** – A separate programmer would then develop another script in attempt to verify the same output.
- **Visual Inspection** – An independent reviewer would do a complete or partial visual inspection of the output, program and log.

In addition to the above mentioned validation process, the user needs to document each step. This section demonstrates the process of validating a SAS program along with techniques of capturing and generating associated documentation. The following example illustrates the steps that a user would take within this process:

STEP 1: **INITIAL VERSION –** This process would begin with the versioning of the SAS program to track the program's life cycle. This can be automated through a right mouse click and selecting the option "V - Version + Notes". A comment or notes are captured as part of the documentation which explains the significance of this step.



Version Backup

Path: E:\Biostat-Validation\MEDI534\CP119\Table\Back_Up

Program: V_Urine_Tox2.sas

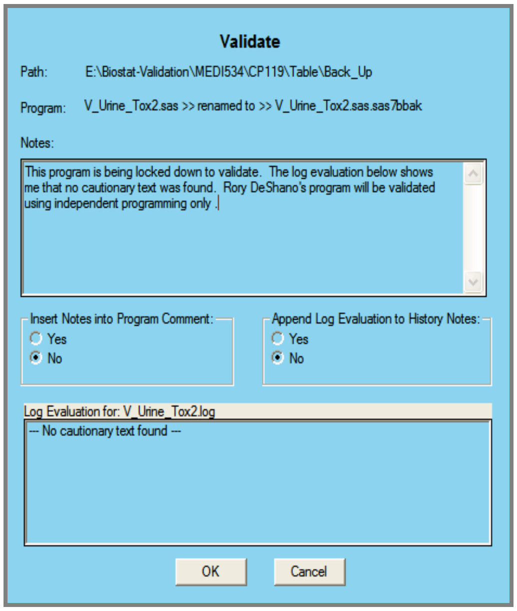
Notes:

Completed programming on 18Sep2006 and notified the validation team. This program creates a toxicity table to measure baseline results against Day 180. This is a study specific program and will nnot be used out side this realm.

Insert Notes into Program Comment:
○ Yes
● No

figure 8

STEP 2: **LOCKING FOR VALIDATION –** Once the program is ready for validation, the user would then lock down the code so that no additional changes can be made during the validation process. This process is automated through the "V – Lock for Validation" right click menu item. This captures a significant step in the change control process since it marks the beginning of the validation state. From this point on, there will be no more changes made to the program logic.



Validate

Path: E:\Biostat-Validation\MEDI534\CP119\Table\Back_Up

Program: V_Urine_Tox2.sas >> renamed to >> V_Urine_Tox2.sas.sas7bbak

Notes:

This program is being locked down to validate. The log evaluation below shows me that no cautionary text was found. Rory DeShano's program will be validated using independent programming only .

Insert Notes into Program Comment:
○ Yes
● No

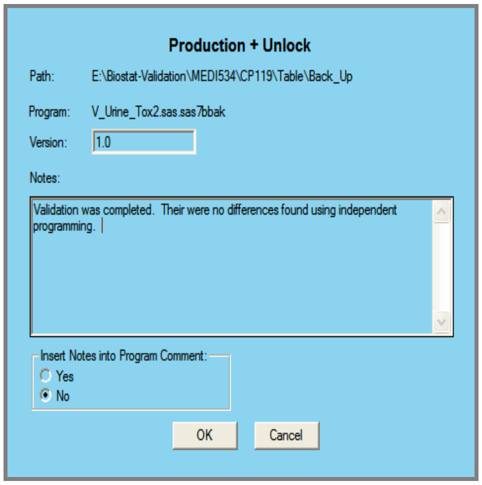
Append Log Evaluation to History Notes:
○ Yes
● No

Log Evaluation for: V_Urine_Tox2.log

```
--- No cautionary text found ---
```

figure 9

STEP 3: **DOCUMENT TESTING AND SET TO PRODUCTION –** Validating the output or logic of the SAS program will be applied in this step. In this instance, we use an independent SAS program to perform the test. This program will in effect document the findings pertaining to the original program being validated. Once all deviations are resolved through this validation effort, the user would establish a new state of "production" to the program being validated. This assignment can be applied using the "Production + Unlock" function. Note that this function can only be applied once a program has been locked down for validation. This prevents validating and testing against a moving target.



Production + Unlock

Path: E:\Biostat-Validation\MEDI534\CP119\Table\Back_Up

Program: V_Urine_Tox2.sas.sas7bbak

Version: 1.0

Notes:

Validation was completed. Their were no differences found using independent programming.

Insert Notes into Program Comment:
○ Yes
● No

figure 10

The above series of steps are integrated into the work flow of the SAS programmer since the user applies each step by right mouse clicking on the program within Windows Explorer to perform each task. These steps also automatically store the information centrally while capturing the user name and date time of the validation analyst. The information captured at this point is later used to generate documentation. This documentation is an essential component used by a reviewer or auditor to understand the efforts applied during the validation process.

REPORTS AND DOCUMENTATION

Reports can be generated to accompany the documentation used when communicating the development efforts to collaborators. These reports can serve multiple purposes since they are also useful documentation in the validation process, as seen in the previous steps. This section will illustrate some common reports and how they can add value to the process by documenting changes to the program.

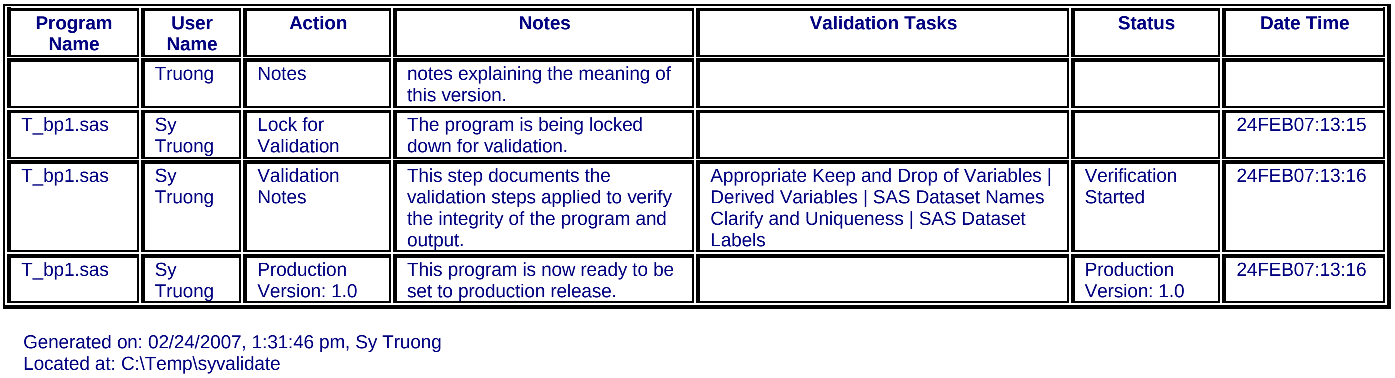
The user can acquire pertinent information through the reports that can be generated. Some of the most common reports include:

- **Current Status** – The user can obtain the latest information pertaining to the program. This will display information pertaining to the latest change control step applied to the specified program.
- **Complete History –** A history can allow the reviewer to see a complete view of all the steps applied to the program. This provides a method for tracing back in history to identify the step that may contribute to discrepancies. The traceability is a crucial element in the process of an audit.
- **Program View –** In addition to the metadata, such as notes pertaining to the program or the user name capturing who applied the changes, the actual program code can also be generated into the report. In this case, it is useful to have the latest version of the program captured when communicating to collaborators. For example, if the user needs to send all the latest production release SAS programs, this report can be generated and delivered to fulfill the request.

There may be other reports that can be generated but the above mentioned capture some of the most common needs. An example of the "complete history" report is illustrated here:

History of t_bp1.sas

Program Name	User Name	Action	Notes	Validation Tasks	Status	Date Time
T_bp1.sas	Sy Truong	Version Backup				22JAN07:08:38
T_bp1.sas	Sy	Version +	This is an example version with			24FEB07:13:15



Program Name	User Name	Action	Notes	Validation Tasks	Status	Date Time
	Truong	Notes	notes explaining the meaning of this version.			
T_bp1.sas	Sy Truong	Lock for Validation	The program is being locked down for validation.			24FEB07:13:15
T_bp1.sas	Sy Truong	Validation Notes	This step documents the validation steps applied to verify the integrity of the program and output.	Appropriate Keep and Drop of Variables | Derived Variables | SAS Dataset Names Clarify and Uniqueness | SAS Dataset Labels	Verification Started	24FEB07:13:16
T_bp1.sas	Sy Truong	Production Version: 1.0	This program is now ready to be set to production release.		Production Version: 1.0	24FEB07:13:16

Generated on: 02/24/2007, 1:31:46 pm, Sy Truong
Located at: C:\Temp\syvalidate

This step is accomplished automatically through the right mouse click and selecting the "V - Reports" item. The report selected is then automatically generated for the selected program. All the information that is captured during the change control process is stored in a SAS dataset with the following variables.

Variable	**Label**
pgmname	Program Name
usrname	User Name
action	Action
index_number	File Index Number
index_name	File Index Name
notes	Notes
tasks	Validation Tasks
status	Status
datetime	Date Time

The report generated above is a simple PROC REPORT of the history dataset with a WHERE clause capturing the specified program. These prepared reports allow users to quickly acquire valuable information pertaining to the changes of the SAS programs and associated output. This same dataset used to generate these common reports is also available to users. Since many of the users are SAS programmers, additional custom reports can be generated to answer questions or reveal pertinent information during the development cycle of the SAS program. The prepared reports add value by making it easy for users to obtain change control information, but the ability to subset and conditionally generate dynamic reports can add more value to the users by providing flexibility and power to the process.

CONCLUSION

A modern computing environment with a user friendly interface contains the ability to right mouse click on an object and obtain more information. The contextual information provided through this step is essential in understanding the development and validation of SAS programs. This paper presents techniques and methodologies which combine the user friendly steps of right mouse clicking on SAS programs or output while capturing the change control information. This metadata adds context and meaning to the history of the SAS program development cycle. From the perspective of a reviewer or auditor, if the user did not document it, it was not done. This paper therefore also emphasizes the importance of documenting every step through reports. This process of capturing and documenting the information, especially during validation, can be very time consuming and resource intensive. The suggested approach described in this paper provides methodologies and automation approaches to each step while integrating the tasks into a SAS programmer's environment. This therefore provides an environment with tools and automated processes which provide an efficient way of managing change control. Although there may be numerous steps for change control and validation, it is simplified through a right mouse click.

REFERENCES

ABOUT THE AUTHOR

Sy Truong is President of MXI (Meta-Xceed, Inc.) They may be contacted at:

Sy Truong
MXI, Meta-Xceed, Inc.
1751 McCarthy Blvd.
Milpitas, CA 95035
(408) 955-9333
sy.truong@meta-x.com

If you have any comments and questions regarding this paper, please contact:

Rory DeShano
deshanorj@yahoo.com