Paper 203-2011

Achieving Efficiencies using SAS® Drug Development

Aik Hoe Seah, Eli Lilly and Company

ABSTRACT

More pharmaceutical and biotechnology companies are using SAS® Drug Development to handle clinical data and perform statistical analyses. As a result, many users who are transitioning from Base SAS® find out that they have to change many habits to use SAS Drug Development effectively. This presentation discusses some pros and cons of working with SAS Drug Development. It also provides some alternative models in SAS Drug Development that might improve performance and efficiency.

Keywords:

SDD, SAS Drug Development, WebDAV, Job Editor, Scheduler, Process Editor

INTRODUCTION

SAS® Drug Development (SDD) is a web-based industry-compliant system which businesses can use as a secure central repository of data and analysis, inclusive of SAS analytic development and execution functionalities, more commonly known to users as Base SAS (SAS 9.1 or 9.2). With its well-controlled regulatory-compliant environment, SDD is more frequently used to support clinical trials development.

For long-time users of Base SAS, transitioning to SDD might not seem at first to involve a steep learning curve, as the look and feel of SDD and its Graphical User Interface (GUI) is made simple; with its expandable-and-collapsible left panel and centre panel that allows viewing of files and subfolder details. However, as one spends more time in SDD, one will realize that there are many little differences when comparing SDD and Base SAS usage. The next section discusses some obvious and not-so-obvious observable differences found while using SDD and how having a production code in Base SAS may not equate to having a production code in SDD.

MAJOR DIFFERENCES

Operating System

Depending on the operating system where SDD is implemented, a common user of Base SAS on a Windows environment might suddenly find that their existing code developed in Base SAS will not initially run when uploaded into SDD. This is because SDD has multiple servers (UNIX, Oracle®, and MS® Windows®), although SAS codes are run on the UNIX server. As such, there has to be more care taken when inserting input and output paths in your SAS program, as paths in UNIX environments are caps-sensitive.

SAS Version

For SDD 3.4, the equivalent Base SAS version is 9.1 and from SDD 3.5 onwards, the equivalent Base SAS version is 9.2. This naturally means there are SAS procedures which are not made available if you are using SDD 3.4. Some examples are new features in the REPORT procedure (e.g. being able to reset page numbers between BY groups), being able to create plots using the FREQ and UNIVARIATE procedures via ODS styles, and being able to use the MCMC procedure. If you use Base SAS 9.2 to develop your code and eventually move it to SDD 3.4, there will be issues that appear when running your program, as you are trying to call procedures or features that are not yet included in the particular version of SDD.

Macro Parameter List

In Base SAS, we have the Editor, the area in which we start writing or building our program. In SDD, this is called the Process Editor. At first glance, it is easy to see that the large top right area of the Process Editor is where we write our program. On the lower right area just below the text box is where you will find the macro parameter list. In SDD, the macro parameter list is very important as the list is where users can set paths for their data source and the paths where their analysis will be output, besides parameters for user-defined macros. If the macro parameter list is not filled correctly, the program which you developed in Base SAS will not run successfully and the log will be filled with ERRORs and WARNINGs.

Graph Output

We are able to view graphs in Base SAS through the Graph window. However, we are unable to do so easily in SDD. One way to view graphs within SDD is to run your graph procedures (GPLOT, GCHART, GREPLAY, etc) with ODS output to a specified location in SDD and open the plots as part of a RTF or PDF document, just to name a few common output destinations.

MINOR DIFFERENCES

The Import / Export procedure

If you are a Base SAS user who relies on the Import or Export procedures to input your data (which is in XLS or CSV format) into the SAS session and after manipulation, output the resulting data into another file format like XLS for someone else to further process the data, you may suddenly find that your program now does not run successfully in SDD. This is because the Import and Export procedures usually found since SAS version 8, is a functionality which is not available in SDD. CSV files can be read by using the INFILE statement in the usual DATA step. However, XLS files cannot be read this way. Further information on how to tackle this issue can be found in the paper "Importing Complicated Excel® Files into SAS Drug Development" by Murphy and Steffens.

One option is to convert all your CSV/XLS files to SAS7BDAT or TXT using the Advanced Loader. However, if a user would like to preserve the documentation for raw data such that nothing is changed for the original CSV file, SDD users can use the INFILE statement to input your data. Here is an example on how to set the statement up:

```
data test;
        infile &datapath.
        dlm=',' dsd termstr=crlf lrecl=4096 truncover firstobs=2;
        input study $ subject treatment :$35. part $ sequence $ period :$15. day dose analyte :$15. @;
run;
```

As a reminder, you need to have the *&datapath* macro variable without quotes around it, in order for the DATA step to run successfully. The option ***termstr=crlf*** is important if the CSV file was created in a Windows environment, to denote the end of line criteria. If this is not specified, users will have trouble accessing the last variable in a dataset, as a hidden Carriage Return symbol is included in the cells of the last variable. TERMSTR= controls the end of line/record delimiters in PC and UNIX formatted files. This option enables the sharing of UNIX and PC formatted files between the two hosts, and since SDD runs programs in a UNIX environment, it is suggested to include the option in any INFILE statement calling CSV files.

TERMSTR=	
CRLF	CRLF Carriage Return Line Feed. This parameter is used to read PC format files
NL	Newline. This parameter is used to read UNIX format files. NL is the default.

PROS OF SDD

Secure Environment

SDD provides a secure environment for user access as each user has his/her own unique identification and a strong password that utilizes small letters, caps letters, numbers and symbols. This discourages any hackers trying to break into your account, as a significant amount of time is required for the permutations needed to break a strong password.

SDD also has fine grain permission settings capabilities for each file and folder. If implemented properly, different groups of people utilizing SDD for different purposes will not have access to files by "stumbling" on them. As a matter of fact, they will not even be able to know they exist in the particular folder!

Central Repository

With a central repository, users can be assured that there is only a single version of data. This allows users to save the time otherwise spent trying to find out if the data on hand is the most updated version for analysis.

Users are also be able to mine data more efficiently when all data can be found in a single repository. Data can then be easily analyzed across trials to support better future planning or new findings.

Auditing and traceability

Files in SDD have full audit trails when a new version of a file is updated, ensuring existing programs are able to point to whichever version of data they need to use. This ability is particularly helpful for clinical data, as programmers can then run interim analyses more easily when there are trials that run for a significant period of time.

Clinical research results also need clearly documented integrity. Significant time can be spent on manual processes and even more time on defending the work done, if a regulatory agency does not believe the results to be credible. This may even delay product approval.

SAS Functionality

SDD is not just a big hard drive to store your data; it also allows users to perform analyses using its SAS functionality, which means the whole analytics process is fully executed within the system. From uploading of data, to data processing, to analyzing and, finally, preparing the results output, SDD is able to keep logs of the full process just like a normal SAS log. This way, there will not be any missing linkages in data integrity, where programmers have to bring the data or results out from the system and process it further with other tools.

SDD USAGE HABITS

Broadly speaking, there are two different categories of SDD users. The first type is users who do not use its SAS functionality. They are those users who are mostly at the beginning or at the end of the analytics process. This includes users who upload the raw data (which is at the beginning of the process) or those users who write the report using the results generated by the analysis (who stand at the end of the process). These users only need to know how to access the folders in which they need to upload the data or download the results needed.

The second type SDD user is the one who uses SDD more frequently, filling in the full analytics process from data processing to a form of standard data set, and running the analysis using the data to generate results in a nicely formatted table, figure or listing.

Naturally, the level of complexity for the first type user would not be too difficult as they only require basic training in SDD; to be able to drill down to the folder they need access and use the Basic/Advanced Loader functionality. However, the second type user tends to spend a significant amount of time within SDD, developing code, testing and debugging before putting the code in production. Here, we will focus on the second type, those who requires heavier usage of SDD.

MODEL IN INCREASING EFFICIENCY

Basic Scenario – SDD hosted in single location

As SDD only requires Java Runtime Environment to be installed and an Internet connection, a user will be able to develop his code from any computer globally with these two minimal technical requirements. In the most common scenario where the user only has access to SDD as the primary analytics tool, the user has to develop, test, and debug the code using only SDD. For this user, a good Internet connection is very important. SAS allows its customers to have a hosted solution (where SAS will have servers dedicated for their customer) or to have customer-based solution (where the customer will have to provide their own IT infrastructure and resources).

No matter if the solution is hosted or customer-based; a major factor that will affect web-based applications is connection speed and stability. Assuming the solution is hosted in Cary, North Carolina, where SAS Institute has its headquarters, a user for a company with global operations will soon find that its employees situated near Cary or within the US experience low network latency while users in China or Japan experience higher network latency. The theoretical peak Internet bandwidth is naturally fixed depending on the technology being used as well as cost

consideration to the local Internet service provider. The reasons for high Internet latency are propagation, transmission and processing delays. For a program to successfully run in SDD, information from the local computer needs to be sent to the server in Cary before the server runs the analysis and then sends the results back to the local computer for viewing. If this is being done very frequently (as it will be, with program errors encountered, debugging, and re-running the program), there is a significant time increase for a successful run when compared to Base SAS, as data in Base SAS are most commonly loaded to the current work session.

Model – SDD in combination with Base SAS

Assuming a user has access to both SDD and Base SAS, he/she will be able to use a combination of these to have faster program development. Xythos has collaborated with SAS to develop software that enables network drive mapping of SDD. Named "SAS Drug Development Desktop Connection (SDDDC)", users can map the SDD directory as a network drive and are then able to call library references using the "libname" command. This allows the user to copy data to the work session temporarily or have files copied to a local PC cache for processing. A point to note is SDDDC cannot be used in a Citrix or Windows Terminal Services environment. SDDDC uses Web-based Distributed Authoring and Versioning (WebDAV) technology which is functionality already existing within SDD. However, program traceability is limited when using the WebDAV functionality, as the SDD log does not trace the WebDAV files and it also cannot access specific versions of files.

Examples of WebDAV code:

```
filename ref sasxbamw "URL to webdav location" user="&userid" password="&pw" DIR FILEEXT;

Libname ref base "URL to webdav location" webdav user="&userid" password="&pw";
```

If a user has a secure environment (e.g. a local network drive) in which data can be copied from SDD, an alternative model to developing code could be used.

1. Copy data needed permanently from SDD to secure environment using WebDAV
2. Develop SDD-compliant code using Base SAS, accessing data from secure environment, until program is very near production-ready.
3. Proceed to move code from Base SAS to SDD, changing values in macro parameter list to point to the correct input/output locations
4. Run the SDD program (and delete data residing in secure environment)

By using this method, users will reduce the time spent developing code within SDD and at the same time, the program in production will continue to retain its data traceability in the log.

However, the user still has to spend some time in the beginning to copy the data from SDD and problems might emerge when there is an update in the data when program development is ongoing, causing results which are different between Base SAS and SDD.

As such, it would be good to follow a habit in developing programs in SDD by using a combination of the Job Editor and Scheduler.

Job Editor and Scheduler

Perhaps the least used tools in SDD for users are the Job Editor and Scheduler. For common SDD users, it is sufficient to use the "Code" -> "Test" -> "Save Results" or "Do Not Save Results" in the Process Editor for a job to run.

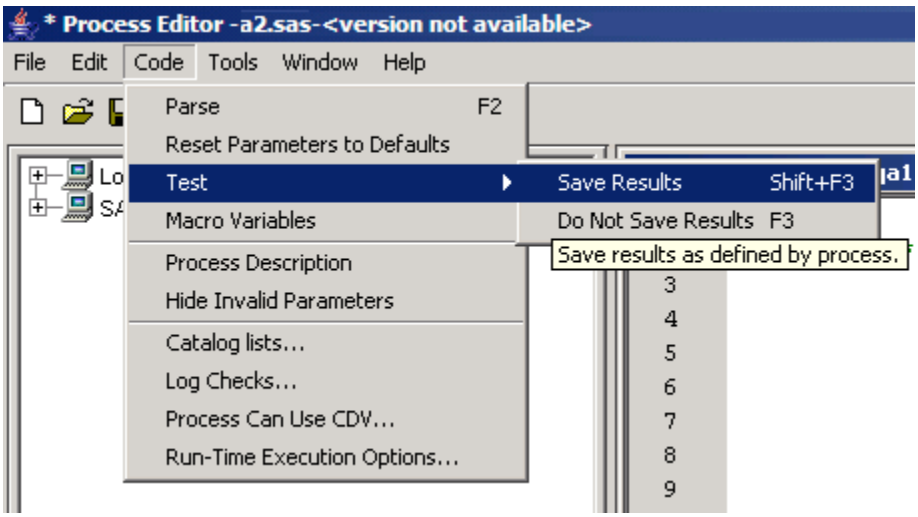

Figure 1: Running a job in Process Editor

However, for heavy SDD users, it is likely that there will be many situations where the program will not run successfully or that result in a program running for a longer period than expected. Here are a few reasons, amongst many, why this could happen:

1) The datasets being called by the program are significantly large

2) There are analyses requiring complex calculations

3) There are macros being called repeatedly within the program

In any of these cases, one should consider using the Job Editor and Scheduler tools combination so that the program will run within its expected time. The Job Editor tool allows users to save a few programs in a batch, and subsequently, run it immediately or schedule it to run at a future time using the Scheduler. The Scheduler will then send an email to confirm whether the program batch has been run successfully or with errors after the job is complete. One

advantage that the Scheduler has as compared to the Process Editor is the resources being used while running a program. If a user uses the Process Editor, SDD will have to use some resources from the user's computer in sending the updated program to the hosted location for the server to run, and subsequently retrieve the results. However, if the user uses the Scheduler, the server will run the program entirely without using any resources from the user's computer, which frees up resources for the user to start developing another SDD program.

In SDD, developing code is a multi-step process where users tend to type their program, and then proceed to run the code to check that the Data step has run the way the users intended to, and then proceed to code the next portion of the program. This creates a lengthy process in developing code within SDD, as Internet latency will cause a substantial delay in returning outputs for users to check every Data step. If a SDD user has access to Base SAS, then it will be faster to follow the steps I've mentioned earlier. However, if a SDD user does not have Base SAS, then there is a better way to develop code outlined as below:

1. Take a smaller set of data from the full data set that will minimally satisfy the conditions for the intended program to run.
2. Proceed to develop your code in SDD using the sample data, which will minimize run time and delay in Internet latency.
3. When the SDD program is finally ready, change the path calling the sample data to call the full data set and proceed to create a run job for the program by using the Job Editor
4. Schedule the job using the Scheduler to run the program with the full data set.

This way, the SDD user will be able to get their output slightly faster than having the Process Editor run through every observation in a large data set in each run.

A simple efficiency test was done to provide some additional information. Here, a program more than 100KB was run 30 times in each method (Process Editor or Scheduler) giving us the following results. CPU Time is defined as the time SDD takes to run the program within the SDD environment, while Real Time is defined as the time SDD takes in to provide the user the results from the point of time the user starts running the program.

Process Editor				Scheduler			
CPU Time (seconds)		Real Time (seconds)		CPU Time (seconds)		Real Time (seconds)	
Mean	Std Dev	Mean	Std Dev	Mean	Std Dev	Mean	Std Dev
20.97	3.86	80.43	11.79	22.93	4.35	314.33	37.07

From here we can see that the Scheduler performs just as fast as the Process Editor in running the code by CPU time, with a difference of almost 2 seconds longer for the Scheduler. However, when we compare the real times, the Scheduler takes at least 233.9 seconds longer (which is almost 4 minutes) in getting the output. This is because there is an agent that checks the Scheduler every 5 minutes to see if there are any jobs being scheduled, and the Job ends up running later than the original scheduled time.

With this, we can see that there are pros and cons while using the Process Editor and Scheduler, where the Process Editor is a good environment for users to develop their code (using smaller set of data) or to convert their Base SAS code to SDD code and using the Scheduler to run the code with the full data such that Process Editor resources can be freed up for other programs to use. The model is simplistic, but not without its drawback, which is approximately 4 minutes later for the code to run successfully.

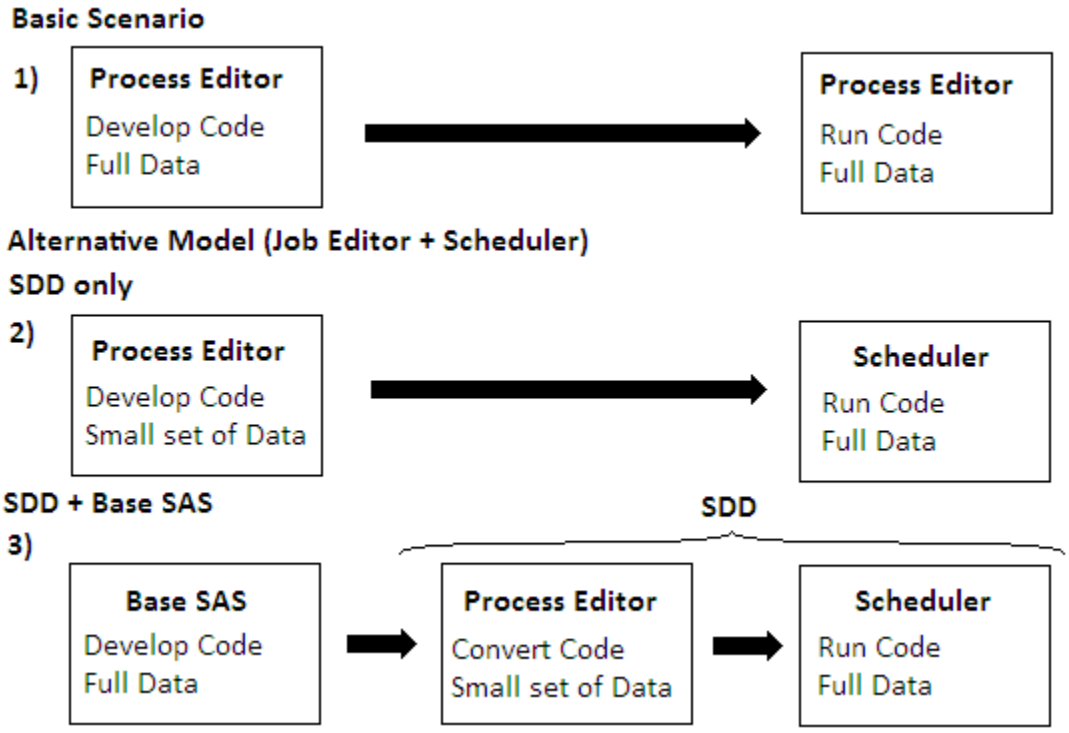

Figure 2: Breakdown in development models

ADDITIONAL TIPS AND TRICKS FOR SDD

Forcing a new SAS session in Process Editor

For SDD users of the Process Editor, it is common to submit code with the Process Editor open. When users submit code again, the Process Editor runs the submitted code based on the current SAS session instead of starting a new session. This may cause unintended results to users, especially if the logic flow from the previous data steps overwrites data in current data steps. There is a way to force a new SAS session in the Process Editor each time, and this can be done by selecting “Tools”, then “Set Pool”. On the “Set Pool” dialog box, there will be only two options, which are “Current” (default), and “Select Pool: Normal Usage”.

To start a new SAS session in the Process Editor, just select the “Select Pool: Normal Usage” radio button and click “OK”. This will allow the Process Editor to start a new session each time a user submits code through it. With this, SDD users will not need to close and restart their Process Editor if they need to run their analysis again after a small change. However, one drawback would be this option needs to be chosen each time a new Process Editor is open, as the setting always goes back to the default, which is “Current”.

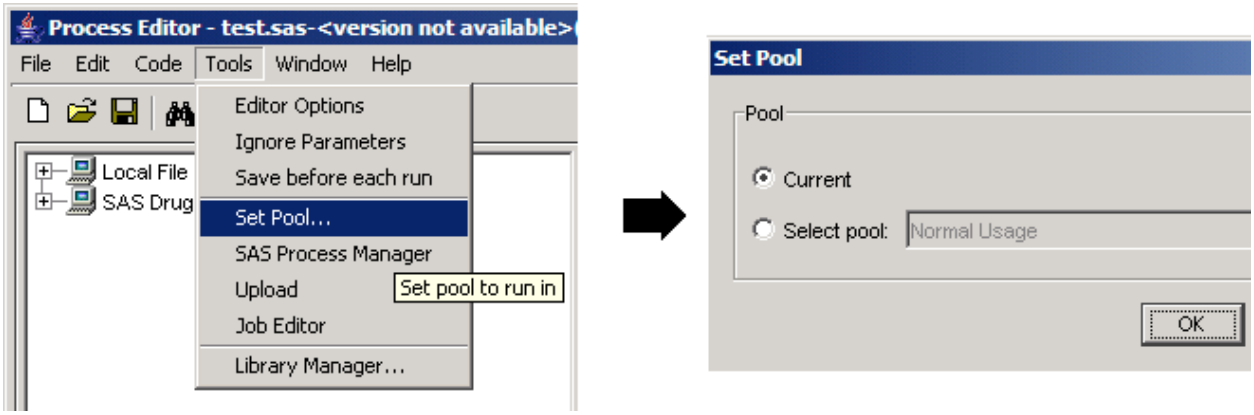

Figure 3: Forcing a new SAS session in PE

Selecting Datasets in the Process Editor Parameters List

For SDD users who need to point to the location of their data using the Process Editor Parameters List, here’s a technique that will save some effort for selecting data in a single folder. This trick assumes there are fewer datasets to be selected than to be not selected.

If a macro parameter in the Parameters List is selected as “Folder”, a user will be able to open up some path options. When a user clicks on the “Input” radio button for “Folder Type”, there will be an area where files can be selected for input. This is important because there will only be minimal datasets being called by a program, to reduce program run time. The default option

which is to “Get all files” will need to be toggled to “Get selected files”. Then, the user will have to spend some time un-checking the datasets not needed by the program.

The Process Editor automatically fills in all files within the selected folder (path), and if there are many datasets, the user will have to go through the trouble of un-checking many datasets. An alternative is to select the option “Get all files excluding selected” which will mean the files un-checked by the user will be the datasets being called in.

However, a user can also use the option of “Get selected files” to check the datasets needed without spending much time to un-check the rest. The user just needs to keep the selection of “Get all files” and close the “Customizing Folder” options. When the user opens up the options box a second time, the check marks on all datasets are cleared, and with this, the user only needs to check the datasets being used.

CONCLUSION

This paper documents some ideas on the level of knowledge needed for a Base SAS user to switch to SDD. Some tricks were highlighted and although they may not be the most important tricks around, they could still be used to save some time or tedious work for SDD users. This paper also discussed some pros and cons while using SDD, as well as some alternative models in using SDD with a Base SAS combination. SAS users from a Windows environment may find that there is much coding knowledge to fill while using SDD, given that codes running in SDD are in a UNIX environment and hopefully more tips and tricks will be shared by SDD users, to help the greater SDD community easily convert Base SAS codes that run to SDD codes that also run (or “to SDD-tize” it)

The future for analytics-on-the-go may indeed be exciting in the near future, as increasingly solutions like SDD can be reliably and securely accessed from the Web. With the development of SAS® Mobile in 2010, one can only foresee that SDD users might also one day be able to do the same level of work or retrieve outputs on-the-go, combining both technologies.

ACKNOWLEDGEMENTS

Michael Luther, Eli Lilly and Company, for your continued support and guidance.
Leong Sing Meng, SAS Institute, for the good comments, clarification and feedback.

REFERENCES

SAS OnlineDoc 9.1.3 for the Web, INFILE Statement, TERMSTR= option
http://support.sas.com/onlinedoc/913/docMainpage.jsp

Xythos-based SAS® Drug Development Desktop Connection (SDDDC) is not supported in Citrix environment
http://support.sas.com/kb/40/505.html

SAS® Drug Development 3.4 User's Guide

SDD Reporting Best Practices Oct 2009, SAS Institute Inc.

Murphy, Steffens (2009) *Importing Complicated Excel® Files into SAS Drug Development*
Retrieved from SESUG proceedings:
http://analytics.ncsu.edu/sesug/2009/AD009.Murphy.pdf

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

Aik Hoe Seah
Eli Lilly & Company
Lilly-NUS Centre for Clinical Pharmacology
Level 6 Clinical Research Centre (MD11)
National University of Singapore
10 Medical Drive
Singapore 117597
E-mail: seah_aik_hoe@lilly.com