Paper 186-30

A Regulatory Compliant Process for Developing SAS-Based Reports

Chuck Reap, SAS Institute Inc., Cary NC

ABSTRACT

This paper describes an efficient process that is tailored for the high-volume, batch-based reporting that is common in the pharmaceutical and other regulated industries. It includes topics such as code sharing, code promotion, code maintenance, check-in and check-out, audit trails, versioning, batch processing, output management, sign off, and quality control. The presentation walks the audience through the process using technology provided by SAS Drug Development 3.0.

INTRODUCTION

How many times have you said, or at least thought, “How do they expect me to finish these reports, when I have to go through all this red tape?”

I’ve heard tales of some companies taking as long as six months to a year to get a new standardized SAS-based report into production. It takes so long because their software development lifecycle takes that long. I know this is extreme, and it is complicated by the global nature of the reports, but I am sure that you have your own stories of delays and frustration at the endless paperwork, reviews, signoffs, and tools that get in the way more than help. The knowledge you deliver through your reports is just too important to be delayed by unnecessary red tape. There has to be a better way. And there is, in the form of the SAS Regulatory Compliant Process.

This paper presents a practical process from the perspective of the SAS programmer. It is my goal to prompt you to think about how you develop your SAS reports, and to help you make improvements to your current processes.

This paper assumes that you are familiar with the basic regulatory requirements of your industry and does not delve into these in detail. Rather, it is focused on those issues that have a direct impact on the SAS programmer who spends most of his/her day producing reports. Even here, not all SAS-based reporting is the same. Some reports are standardized for reuse and some are written for a single use.

While the SAS Regulatory Compliant Process is suitable for developing reusable reports, it is tailored for single-use reports.

Single use means that the program produces a single set of results based on a predefined set of input data. A single-use report is likely to be run more than once and its results may be used in multiple final documents. They do not require additional user input and are often run in batch and run as a group of reports. They are sometimes scheduled. With these reports the validation of the results is as much an issue as is the validation of the SAS program itself. The program’s requirements are likely to have been clearly defined. By far the most common form of code reuse is found in calling SAS macro libraries, and in copying an existing program to create a new report.

The SAS Regulatory Compliant Process is supported by, but does not require, SAS Drug Development 3.0.

BACKGROUND

As mentioned above, I will not go into a lot of details about the actual regulations; however, I have included a list of excellent resources on the subject in the References section of this document.

Programmers sometimes forget that the goals of a good software development process are to increase the overall productivity of the entire process and to improve the quality of the final results. The process is much more than just the writing SAS code. To evaluate the efficiency of a process, you must ask, “Is my organization efficiently producing report results and are the results we are producing accurate and reliable?”

I believe that most of the requirements of a regulatory compliant process are just good business practices. Even if you did not work in a regulated environment, you would still want to follow most of the practices encouraged by the regulations. While the SAS Regulatory Compliant Process is written primarily for the pharmaceutical industry, it is applicable in any industry, whether it is regulated or not.

The SAS Regulatory Compliant Process is roughly based on the General Principles of Software Validation; Final Guidance for Industry and FDA Staff (January 11, 2002; CDRH). In addition to its FDA pedigree, this guidance provides a straightforward software development life cycle that does a good job of capturing the basic waterfall flow that is common to all software development life cycles. But the FDA guidance presents a lot of concepts that are simply beyond the scope of the SAS Regulatory Compliant Process. For example, the FDA document discusses retirement, which is important, but it is not something we are concerned with here.

When you start talking about software development life cycles and processes, you quickly get into philosophical wars among those who favor the various flavors. The truth is that each software development life cycle has its pros and cons. For example, the Extreme Programming (a.k.a. XP) software development life cycle is best suited to smaller software development projects that do not have well defined requirements. The Rational Unified Process is best suited for very large and complex software development projects. Neither is directly suitable for our regulatory process, but they both have much to offer.

I do not claim that the Regulatory Compliant Process is the only way to meet your regulatory requirements. I strongly encourage you to create your own customized process—one that is tailored to best meet your own organization's culture, requirements, and resources.

REGULATORY TOOLBOX

There is an old saying that goes, "A poor carpenter blames his tools." But anyone who has ever done any real carpentry work will tell you that having the right set of high-quality tools will make your life much easier. You can implement the SAS Regulatory Compliant Process using manual processes. You do not have to have a fancy computer system to do it. As with a carpenter, the standard you are measured against will be the quality of your final results, not of your tools. But clearly, having the right set of high-quality tools is going to make it much easier for you to create those high-quality results.

The SAS Regulatory Compliant Process is built on top of a rich set of tools that are collectively known as the Regulatory Toolbox. The Regulatory Toolbox is the set of manual tasks, computer systems, and working practices that aid in the development of regulatory-compliant SAS reports.

The table below lists some of the more important tools you need to have in your Regulatory Toolbox. It also gives a description of how SAS Drug Development 3.0 provides those tools.

Tool	Description	Common Implementation	SAS Drug Development 3.0
Program Editor	An editor for creating and editing SAS program files. The use of an enhanced editor that aids in identifying syntax errors is strongly encouraged.	SAS Display Manager, SAS Enterprise Guide, Text Editors, third-party program editors.	The Process Editor provides automatic search of the SAS Log for errors, an enhanced editor, and direct access to the SAS Drug Development 3.0 repository. Plus, you can continue to use SAS Display Manager, SAS Enterprise Guide, Text Editors, and third-party program editors.
Batch Queue	Submits the SAS program to run in batch mode.	SAS batch mode, operating system batch queues.	Jobs can be run in batch, but batch queues are not required.
Scheduler	Schedules batch jobs to run at a specific time.	Operating system schedulers (i.e., AT and CHRON)	The Scheduler enables you to schedule jobs.
Explore Data and Metadata	Interactively explore SAS data and/or metadata. This is used to better understand the input data and to verify the output while developing the SAS program.	SAS Viewer, SAS Display Manager, SAS Enterprise Guide, submitting code.	The Data Explorer is used to explore data. The Data Definition Explorer is used to explore metadata.
Version Control	The ability to access and revert to the prior versions of the file.	Source management systems; some operating systems provide for version control (e.g., VMS), manual processes.	Version control is supported.
Check-in / Check-out	Identifies who is currently working on a file and prevents others from updating the file while it is checked out.	Manual processes or source management systems.	Files can be checked-in and checked-out.

Tool	Description	Common Implementation	SAS Drug Development 3.0
Audit Trail	The record of actions that have been taken against the file.	Source management systems, log files, paper logs, manual processes.	A complete audit trail is maintained for all files.
Coding Standards	Define organizational standards for the formatting, styles, and basic techniques to provide consistency among programmers and to promote good programming practices.	Standard operating procedures, guidelines, style guides, and checklists are enforced by visual inspections.	Standard operating procedures, guidelines, style guides, and checklists are enforced by visual inspections.
Peer Review	A second programmer or scientist reviews the code. This is sometimes called code inspection.	Standard operating procedures, guidelines, style guides, and checklists are enforced by visual inspections.	Standard operating procedures, guidelines, style guides, and checklists are enforced by visual inspections.
Developer Test SAS Macro Library	SAS Macros that aid in developer testing by checking for error conditions as the code is executed. Developer testing is sometimes called unit testing.	Programmers write their own checks or call a set of standard SAS Macro library.	Programmers write their own checks or call a set of standard SAS Macro library.
Sign-offs	Giving official approval that a work product is complete and accurate.	Physical signatures on paper, electronic log files.	Files can be signed electronically.
System Level Security	Protecting the entire system or parts of a system from unauthorized access.	System-level logins, file system permissions.	System-level login is required; file system permissions and access to some features can be controlled using permissions.
File Level Security	Protecting an individual file or folder from unauthorized access.	File system permissions.	File system permissions are supported.
SAS Data Strategy	Defines where the SAS data will be stored and how the data will be updated.	Naming conventions and folder structures are often used to organize the files. The SAS data are stored on network drives, networked servers, or a SAS server. SAS Views are sometimes used to access data stored in database management systems. Sometimes a data warehousing tool or ETL tool is used.	Naming conventions and folder structures are often used to organize the files. A regulatory compliant repository is provided, but SAS Drug Development 3.0 has robust capabilities for accessing data stored in other systems. These capabilities can be expanded to support proprietary systems.
SAS Code and Results Management Strategy	Defines where and how the SAS code and the output they create will be stored, organized, tracked, and managed.	Naming conventions and folder structures are often used to organize the files. The files are often stored on network drives, networked servers, a SAS server, or a code management system.	Naming conventions and folder structures are often used to organize the files. A regulatory compliant repository is provided with SAS Drug Development 3.0.

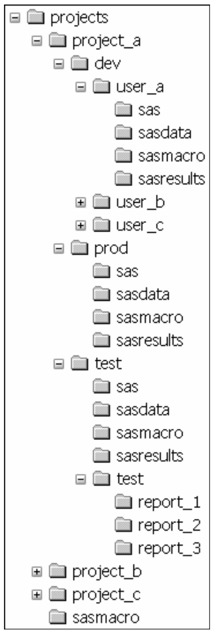
SAS CODE AND RESULTS MANAGEMENT STRATEGY

It is essential to have a well thought out plan for how you will manage your SAS programs, SAS logs, and SAS output. Particularly in larger organizations, you need to consider how and where your code will be stored, how you will manage the state of that code, and how you will share code between multiple projects and programmers. Along with this you will need to define where the SAS logs will be saved. A strategy for managing the SAS output (results) must also be defined. Where will SAS output be written and in what format, how will the results be approved for use, and even how will they be accessed?

DEV-TEST-PROD

The SAS Regulatory Compliant Process uses a DEV-TEST-PROD-based code and results management strategy. This means that it defines distinct environments for development, testing, and production. The SAS program is then physically copied from DEV to TEST to PROD as the code is promoted (i.e., copied) through the process. Having distinct environments gives the programmer much greater freedom in how they configure their own development

environment, yet it maintains a centralized and controlled test bed, and ensures that development and testing have no impact on the production environment.



```
projects
    project_a
        dev
            user_a
                sas
                sasdata
                sasmacro
                sasresults
            user_b
            user_c
        prod
            sas
            sasdata
            sasmacro
            sasresults
        test
            sas
            sasdata
            sasmacro
            sasresults
            test
                report_1
                report_2
                report_3
    project_b
    project_c
    sasmacro
```

Figure 1: Folder Structure

The DEV-TEST-PROD system relies on having three parallel folder hierarchies, as show in Figure 1, Folder Structure. Under the project level is a folder for each of the three development levels. Each of the development levels has identically named folders for storing the results, the SAS code, the SAS data sets, and the project's SAS Macros. The folder **projects/sasmacro** is used to store SAS Macros that are global to all of the projects.

There are a few exceptions to the parallel structures. The first is that the TEST environment adds the test folder. This contains a folder for each report. While the exact contents of this folder will depend on the needs of each report, it holds items such as the Test Plan, test programs, and test data that are specific to each report. These test items may be copied to and used in the DEV environment, but they are never promoted to the PROD environment.

The second exception to the parallel structures is that the DEV environment is pushed down a level to insert a folder for each developer. This enables a developer to customize their DEV environment without impacting any of the other developers.

The development levels do not have to be on the same system; although I recommend that the TEST environment be on the same system as the PROD environment, or at least on a system that is configured as closely as possible to the PROD system.

By allowing the DEV environment to be on a different system, you are given the flexibility to use whichever development tool(s) you need. This may be your favorite program editor, PC-SAS, Display Manager, or SAS Enterprise Guide. You maintain your regulatory compliance by maintaining your official TEST and PROD environments on a centralized, validated, and controlled server. Note that splitting the development levels on to different systems can require additional coding procedures. In the interest of brevity, I assume that all three of the development levels are on the same system.

Some organizations may opt to completely forego the use of a DEV environment. In these organizations, the development will be in the TEST environment. If you are working in such an environment, then you must be more careful how you configure the environment in order to avoid impacting other developers.

If you have a more complex definition of a project, which is often the case, then you may have additional folder(s) inserted in between the projects and project level; for example, **projects/compound/indication/protocol/project_a**.

The SAS program of record will be the latest version of the SAS program stored in the PROD environment, but the SAS program that is stored in the TEST environment is also kept. The history of the SAS program stored in the PROD environment records the versions of the code that are related to production runs of the report. The history of the SAS program stored in the TEST environment records in greater detail the changes to the design and code over time. When a file is promoted from the TEST to PROD, the current version of the SAS program in the TEST environment is copied to the PROD environment and becomes the current version of the PROD environment file. When a file is demoted from PROD to TEST, then the current version of the SAS program in the PROD environment is copied to the TEST environment and becomes the current version of the TEST environment file.

You should use your Regulatory Toolbox to check the files out of the PROD and TEST environments when you plan to update the files. This will lock the file to prevent others from updating the files while you are working with them. It will also let others know that you are updating the files. Conversely, when you have completed your work with a file, you should check it back into the TEST and PROD environments.

SAS MACRO VARIABLES

The SAS Regulatory Process uses several SAS Macro variables that make it easy to move code from one development level to another. Most of these are simply shortcuts to the folders they define. They are constructed in such a way that it is only necessary to change the value of &DEVLEVEL when the code is promoted within the process.

These SAS Macro variables are listed in the following table.

Name	Description
`&DEVLEVEL`	The development level of the SAS program. Valid values are `dev`, `test`, and `prod`.
`&PROJECTNAME`	The name of the project folder.
`&PROGRAMFILENAME`	The name of the SAS program (i.e., the name of the .SAS file with the .SAS extension).
`&PROGRAMNAME`	The name of the SAS program (i.e., the name of the .SAS file without the .SAS extension).
`&GLOBALPATH`	The path to the global (i.e., top-level) folder. For example, **/projects**.
`&GLOBALSASMACROPATH`	The path to the folder that contains the global macros. For example, **/projects/sasmacro**.
`&PROJECTPATH`	The path to the project folder. For example, **/projects/project_a**.
`&BASEPATH`	The path to the base (i.e., development-level) folder. For example, **/projects/project_a/dev/user_a**.
`&SASMACROPATH`	The path to the development-level macros. For example, **/projects/project_a/dev/user_a/sasmacro**.
`&SASPATH`	The path to the folder that contains the SAS code. For example, **/projects/project_a/dev/user_a/sas**.
`&SASRESULTSPATH`	The path to the folder that contains the SAS output, including the SAS logs. For example, **/projects/project_a/dev/user_a/sasresults**.
`&SASDATAPATH`	The path to the SAS data library. For DEV and TEST, this will be test data. For example, **/projects/project_a/dev/user_a/sasdata**.

The following code segment shows how the SAS Macros might be constructed:

```
/* An example of a SAS Macro that sets the SAS Macro Variables based on the  */
/* provided development level. This would be standard for your organization. */
%macro rcpsetup(level, project, program);
  %global devlevel projectname programfilename programname
          globalpath globalsasmacropath projectpath
          basepath sasmacropath saspath sasresultspath sasdatapath;
  %let devlevel=%str(&level);
  %let projectname=&project;
  %let programfilename=&program;
  %let programname=%substr(&programfilename, 1, %length(&programfilename)-4);
```

```
    %let globalpath=%str(c:/projects);
    %let globalsasmacropath=&globalpath./sasmacro;
    %let projectpath=&globalpath./project_a;
    %if (&level=dev) %then %let basepath=&projectpath./&devlevel./&sysuserid.;
    %else %let basepath=&projectpath./&devlevel.;
    %let sasmacropath=&basepath./sasmacro;
    %let saspath=&basepath./sas;
    %let sasresultspath=&basepath./sasresults;
    %let sasdatapath=&basepath./sasdata;
  %mend rcpsetup;
```

The following code segment shows how the SAS Macro %RCPSETUP might be used:

```
  proc printto log="&sasresultspath./&programname..log";
  run;
  libname proj_a "&sasdatapath";
  %include "&globalsasmacropath./globaltitles.sas";
  %include "&sasmacropath./projectfooters.sas";
  ods pdf file="&sasresultspath./&programname..pdf";
  proc print data=proj_a.demog;
  run;
```

In some development environments, such as SAS Drug Development 3.0, the program will automatically know the full path of its .SAS file. In these development environments, it will be possible to construct %RCPSETUP in such a way that it would automatically determine and set the &DEVLEVEL. This would make it so the code could be promoted or demoted within the DEV-TEST-PROD environments without having to make any edits to the SAS program at all.

By using such standardized structures, it also makes it possible to write programs that will perform much of the tedious and manual parts of the process for you; for example, programs that configure your development environment or programs that promote the SAS code for you.

SAS REGULATORY COMPLIANT PROCESS

The following table outlines the SAS Regulatory Compliant Process from a very high level.

#	Phase	Description
1.	Requirements	What are you going to write?
2.	Design	How are you going to write it?
3.	Code	Write it.
4.	Test	Is the code likely to produce accurate results?
5.	Production	Did the code actually produce accurate results?
6.	Maintenance	Fix something.

Figure 2, SAS Regulatory Compliant Process—Part 1, is a graphic representation of the flow of the SAS Regulatory Compliant Process from the Requirements phase through to the Test Phase to the point at which the code is ready for review.

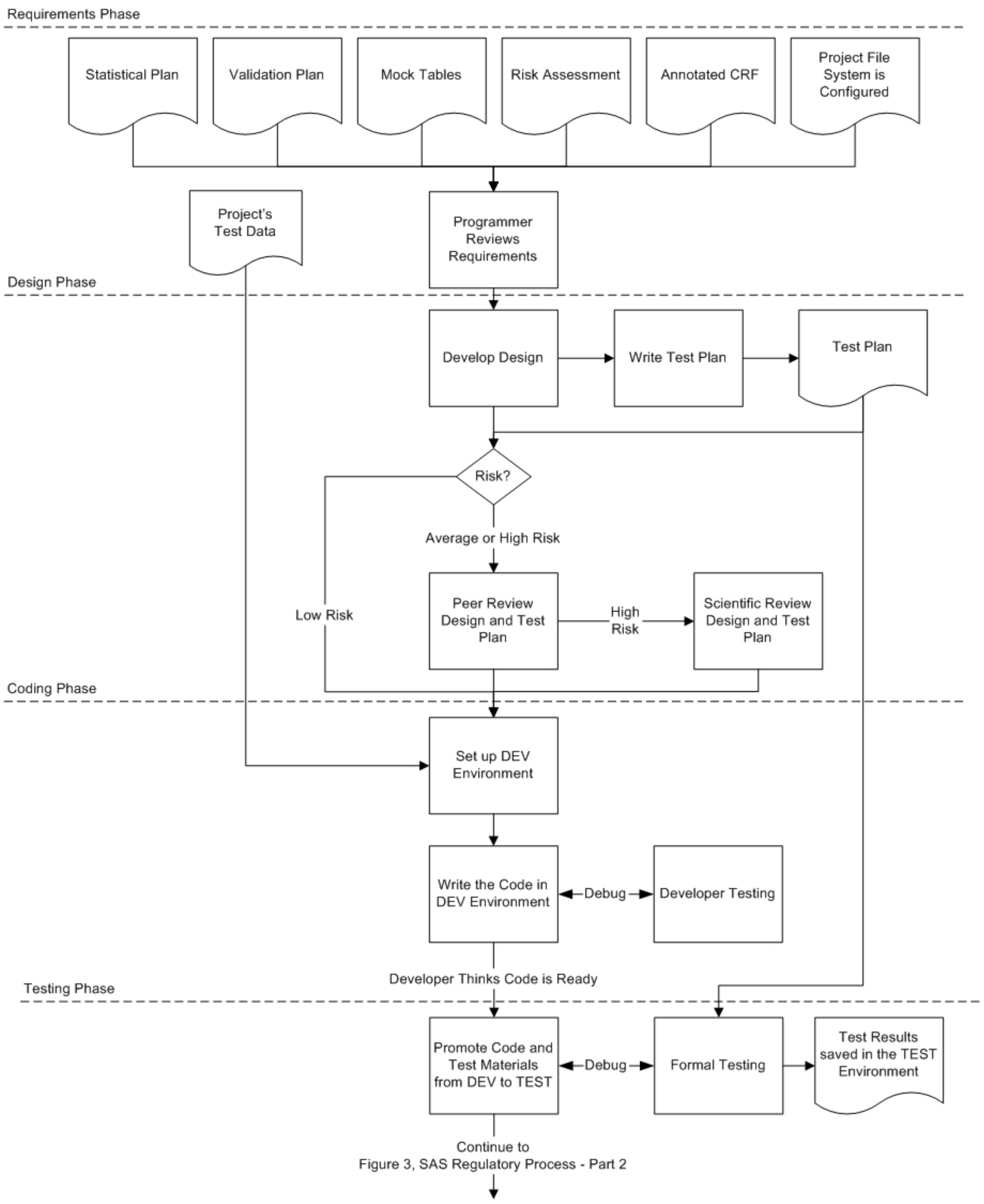

Requirements Phase
Statistical Plan
Validation Plan
Mock Tables
Risk Assessment
Annotated CRF
Project File System is Configured
Project's Test Data
Programmer Reviews Requirements
Design Phase
Develop Design
Write Test Plan
Test Plan
Risk?
Average or High Risk
Low Risk
Peer Review Design and Test Plan
High Risk
Scientific Review Design and Test Plan
Coding Phase
Set up DEV Environment
Write the Code in DEV Environment
Debug
Developer Testing
Developer Thinks Code is Ready
Testing Phase
Promote Code and Test Materials from DEV to TEST
Debug
Formal Testing
Test Results saved in the TEST Environment
Continue to Figure 3, SAS Regulatory Process - Part 2

Figure 2: SAS Regulatory Compliant Process—Part 1

Figure 3, SAS Regulatory Compliant Process—Part 2, continues the graphic representation of the flow of the SAS Regulatory Compliant Process in the Test Phase with the code ready for review and continuing through the rest of the process. The Maintenance Phase is not shown in either of the graphs, since it follows the same basic flow as the original development.

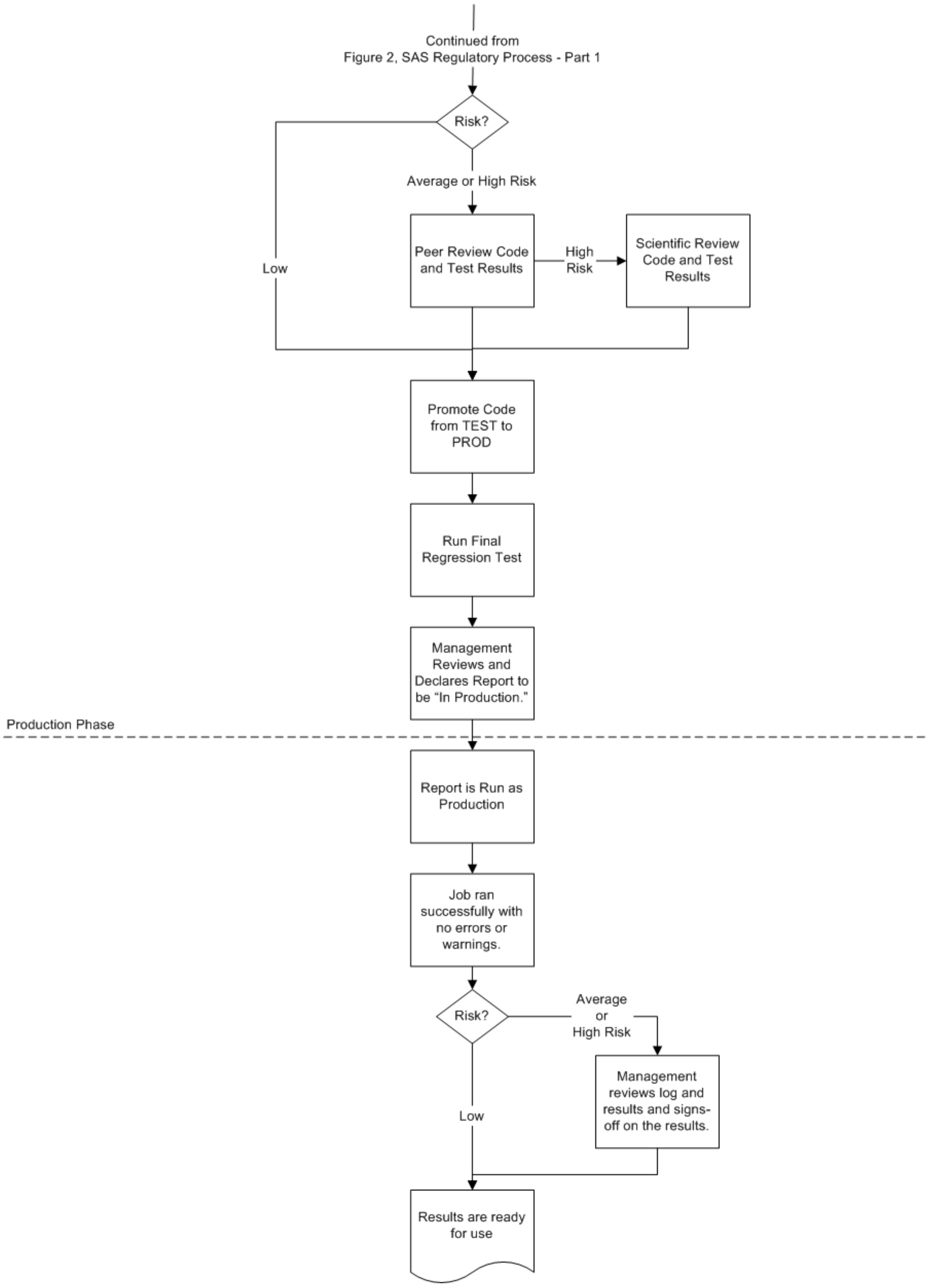

Continued from
Figure 2, SAS Regulatory Process - Part 1
Risk?
Low
Average or High Risk
Peer Review Code and Test Results
High Risk
Scientific Review Code and Test Results
Promote Code from TEST to PROD
Run Final Regression Test
Management Reviews and Declares Report to be "In Production."
Production Phase
Report is Run as Production
Job ran successfully with no errors or warnings.
Risk?
Average or High Risk
Management reviews log and results and signs-off on the results.
Low
Results are ready for use

Figure 3: SAS Regulatory Compliant Process—Part 2

RISK ASSESSMENT

The SAS Regulatory Process provides alternate flows based on the relative risk associated with each report. The higher the risk, the more rigorous the process becomes. The level of risk for each report is evaluated as low, average, or high. The large majority of your reports will be of average risk. Only a few exceptionally important reports will be rated as high risk.

The SAS Regulatory Compliant Process requires that even a low-risk report must be produced by using a rigorous regulatory compliant process to ensure that they produce accurate results. The risk assessment is used to justify whether or not *additional* steps need to be taken to mitigate a higher level of risk.

The risk assigned to a report is not necessarily the same as the value of the report. There are many additional factors that can be involved in determining risk. The risk assessment is ultimately a management decision and is relative to the risks associated with the other reports. It is based on the careful consideration of anything that might increase or decrease the risk to your organization if the report produces incorrect results. This may include the scientific importance of the results, the business importance of the results, the complexity of the code, the novelty of the algorithms, the skill of the programmer, the likelihood that the code will be reused, the risk relative to other reports, and the availability of resources.

This level of risk should be assigned by the research scientists, statisticians, and managers. These roles are defined below.

The management goal in assigning relative risk levels is to reduce the overall risk to the organization, and even more importantly, to ensure the safety of the consumer. If management can pull resources from the lower-risk reports and assign them to reducing the risk of the higher-risk reports, then they can reduce the overall risk to the organization. The natural inclination is to avoid making the decision by saying that all reports are high risk. While it is definitely true that all reports incur some risk, the risk is not the same for all reports, and some reports are going to incur a higher risk than others. The difference is that the higher the risk, the more effort you will need to invest in order to ensure that those results are accurate.

The risk assessment should be performed early in the project. Once the risk of a report has been identified, you will need to document any decision to treat it as a lower-level risk. The inverse is not true. You can always treat a lower-risk report as though it was a higher risk. You are even encouraged to do this, if you sense that some unforeseen circumstance justifies the additional process requirements.

ROLES

The following table defines the roles of the people involved in the process. These are generalizations and may not match precise job descriptions or titles. One individual may play more than one role in the process, especially in smaller organizations. In these situations, you should be conscious of the role you are playing at any given moment, so that you will think from that perspective. In larger organizations, it may be the case that some of these roles are split between multiple people.

Role	Description
SAS Programmer	Designs, writes, and executes reports using SAS software.
Peer Reviewer	A SAS Programmer who is not writing the report and is thus able to provide an independent inspection of the code.
Statistician	Responsible for defining the analytical requirements for the SAS-based reports. Provides statistical and scientific oversight for the SAS Programmer.
Research Scientist	Responsible for defining the overall research and scientific requirements for the SAS-based reports. He/she is the person who will ultimately consume the results of the report. This person will write the Clinical Study Report, Integrated Summary of Safety (ISS), Integrated Summary of Efficacy (ISE), and/or publication that will depend on the results produced by the reports.
Manager	Manages resources and is ultimately responsible for the reporting process, including the results. Managers can and often do delegate their authority and responsibility to others.

REQUIREMENTS PHASE

The SAS Regulatory Compliant Process assumes that the requirements have already been defined as part of the overall clinical trials process. The requirements are most often derived from the protocol and are captured in the form of the Statistical Plan, Validation Plan, Mock Tables, etc. The risk level for the report of risk of the report should be defined prior to this step. It is further assumed that the TEST and PROD environments have already been configured for the project. It is not assumed that the DEV environment has been configured.

The first step in the process is for you, the SAS Programmer, to review the requirements to make sure you understand what it is you are about to do. This includes reviewing the risk assessment.

Once you understand the requirements for the report, you are ready to start by creating the SAS program file (i.e., the .SAS file.) At this point, the program should only contain the header file, which you write based on the requirements, and the code that is standard for all reports. The standard code might include the definition of the standard SAS Macros, OPTIONS statements, standard ODS definitions, etc.
Once you have created the SAS program, you should save it in the TEST environment's SAS folder. If you are using SAS Drug Development 3.0, it is at this point in the process that you would put the file under source control, and then check it out. Doing this enables your organization to identify which programmers are working on which reports. It also ensures that the history of the SAS program is maintained from the very beginning.
I recommend that you create a standard template SAS program for each project. Then for each new program, this template can be copied and edited. This is sometimes accomplished by copying an existing program and editing

that. But it may be a better practice to only copy the bits of code you need as you need them rather than copying the entire file. Even so, copying the entire file is acceptable, but you need to be aware that at this stage of the process, not much of the code will have been edited. I recommend that the header comment block be updated as soon as possible, and that an additional comment be added that states that the code was copied and has not yet been modified. If possible, you should delete all code in the SAS program that is not relevant to the newer report. The reason for this is that the SAS program also serves as the design document, and you will need to be clear as to what parts of the code are current for the new report and which are left over from the program you copied.

You are now ready to move into the Design Phase.

DESIGN PHASE

Design means deciding how you are going to meet the requirements. The design in the Regulatory Compliant Process is an outline of the steps you will need in your SAS program. These are documented in the SAS program as simple step-level comments. Each step-level comment should describe in very basic terms what that step is going to do. The design can also include pseudo code.

The code is the implementation of the design, so the SAS program becomes the design document. As the program is developed, the design is updated along with it. While the final SAS program that exists in the PROD environment will be the official final version of the program, the SAS program will continue to persist in the TEST environment. Because each version of the .SAS program is kept in the TEST environment and is under version control, this will automatically document the changes to the design over time. This is an example of how the Regulatory Toolbox can make your life much easier.

When you believe your design is complete, you should use your Regulatory Toolbox to check the SAS program into the TEST development environment. This is the first official draft of the design document.

It should be recognized that some high-risk reports may require a more formalized design document. This is especially true for SAS Macros that are intended for reuse. One of the reasons we are able to keep the design simple in the SAS Regulatory Compliant Process is that the designs of most SAS reports are straight-forward and are generally step-by-step in their logic. The complexity added by reusable SAS Macros could soon require a more complex design methodology. In these more complex designs, the design is a separate document, which itself must be under version control, but the process would remain essentially the same.

Along with the design, the Test Plan needs to be written. For low- and average-risk reports, the details of the Test Plan can be defined in a standard operating procedure or guideline, and may not need to be a separate document. If this is the case, then the standard operating procedures and guidelines would be expected to define the default set of tests that will be performed, and it would be sufficient to identify the risk of these reports. For high risk reports, an individualized Test Plan is required.

If you have a separate Test Plan document, it is stored in a folder that has the same name as the report's SAS program. This folder is stored under the project's test folder. The Test Plan should be under version control once the first draft has been completed. In addition to the Test Plan, all other formal test-related work products are also stored under this folder. The exact content depends on what is specified in the Test Plan. There may be no additional content, or it may include numerous test programs and test data sets. The results of any formal testing will also need to be stored in this folder. All items under the report's test folder should be under version control once formal testing begins.

The level of review that is required for the design and test plan depends on the risk of the report. The design and Test Plan for low-risk reports does not need to be reviewed by a Peer Reviewer. Average-risk reports need to be reviewed by a Peer Reviewer. The design for high-risk reports must be reviewed by both a Peer Reviewer and a

Statistician. The Statistician will review the statistical methods and verify that the design will meet the scientific requirements of the report.

It is considered good programming practice to write self-testing code. This is discussed more in the Code Section of this paper.

CODE PHASE

At this point we know what we are going to write, how we are going to write it, and how we plan to verify that we wrote it correctly. The next phase is to write the code.

The first step in writing the code is to set up the development environment. I recommend that you use your own DEV environment to develop the report and that the TEST environment be reserved for the final bits of debugging and testing. That said, as long as your report is not changing anything that is being used by others who may be working in the TEST environment, you can write your code in the TEST environment. If you are developing in the TEST environment, then I recommend that permissions be used to reduce the opportunity that programmers have to trash the TEST environment. For example, only certain programmers should be able to update the test data and SAS Macros.

If the SAS program is in the PROD environment, then you should check it out and copy it to the TEST environment. You should also check the SAS program out of the TEST environment. If you are using a DEV environment, then you will also copy the file from TEST to the DEV environment.

Your DEV environment is yours to configure as you see fit. Setting up the DEV environment will typically require copying the test data, SAS Macros, and other files required by the report you are writing. It may also require you to configure your SAS environment. The DEV environment can even be on different computer systems and use different software than the TEST and PROD environments. For example, it could be on a local PC running SAS Enterprise Guide. But you are responsible for making sure that the code will run correctly in the shared TEST and PROD environments.

Due to the dynamic nature of software development, there is no need to record or version the activities within the DEV environment. How long you maintain your DEV environment is up to you. There is no requirement to maintain it beyond its usefulness to you as the SAS Programmer.

In general, the exact details of the developer testing are left up to your discretion. Developer testing is sometimes called unit testing. For low- and average-risk reports, the developer testing does not have to be covered in the Test Plan, but it might be. It will certainly be addressed in the Test Plan for high-risk reports. Your organization’s standard operating procedures may define a minimum level of developer testing. When the Test Plan does require specific developer testing, then the appropriate results will need to be saved to the TEST environment under the report’s test folder.

Whenever possible, you should write your code using an enhanced program editor that will assist you in finding syntax errors as you are typing in your program.

The code you write should be self-testing. Self-testing code will check for error conditions as it is run. When an error is detected by the self-testing code, the run will abort. This kind of testing is some times called unit testing. We highly recommended that you use standard SAS macros to perform most of these checks. Some common problems to check for are missing data sets, bad return codes from PROCs and DATA steps, incorrect data values, incorrect number of observations, and incorrect variable types.

It is up to you, the SAS Programmer, to determine when the code is ready to promote to the TEST environment. When you are ready, simply copy the .SAS program into the TEST environment, along with any additional files that may be needed, such as any required developer test materials.

TEST PHASE

The goal of this phase is it to verify that the code you have written correctly fulfills the requirements for the report by testing the report within the controlled TEST environment.

The SAS program should already be under version control within the TEST environment, as should any other test-related documents, such as the Test Plan, but you are granted some leeway as far as when you place files such as the SAS logs and SAS output under version control. But they too must be placed under version control before you can perform the formal testing. You should complete at least one successful test run of the report in the TEST environment with the related files under version control before moving to the Production Phase.

You should first perform developer testing within the TEST environment. When you have completed your developer testing within the TEST environment, you should run any formal testing as outlined in your Test Plan.

If you find any problems while running the report in the TEST environment, it is your decision as to whether you demote the code back to your DEV environment or simply debug the code in the TEST environment. This is likely to depend on the specifics of the issues you found and on the complexities of moving the program.

If the report is of average or high risk, then your code and the results of your testing must be reviewed by a Peer Reviewer. The Peer Reviewer verifies that the code is logically valid, followed standards, and that the Test Plan was completed and the tests produced the expected results.

The high-risk reports must undergo a scientific review by the Statistician, and may even require an independent creation of the same results as a cross-check. The Statistician will also review the code and the test results, but with the objective of verifying that the statistical and scientific goals behind the report have been fulfilled.
Low-risk reports do not need to undergo either a peer review or a scientific review.

Once the reviews have been successfully completed, the code can now be promoted to the PROD environment. The SAS program is copied from the TEST environment to the PROD environment. The SAS program that is stored in the TEST environment is retained. It is important to note that the reports are not yet considered production even though they are in the PROD environment.

All files stored in the PROD environment are always versioned.

As a final regression test, the report is now run in the PROD environment. This testing verifies that nothing was broken when the code was copied and that the code works as expected in the PROD environment.

If any problems are found, they must be fixed. For low and average risk, it is up to you to decide if you will fix the problems in the DEV environment or in the TEST environment. You should avoid making any changes to the SAS program in the PROD environment.

If no problems are found in the final systems test, then the Manager reviews the report, log, results, and all work products produced by the process. The Manager then signs the .SAS program to signify that the report is in production. These can be signed by manual processes, but whenever possible, electronic signatures are recommended.

The report is now considered to be in production. The report now moves to the Production Phase. The results of the final regression test can be consumed as production results. You do not need to rerun the report.

PRODUCTION PHASE

It is up to you and your organization to determine when and how a report is run, and by whom. A production run can be done once, it can be done via manual processes, or it can be scheduled for automatic execution.

Just because a report is validated as being in production does not mean that it will continue to produce accurate results. This is especially true when the data the report is being run against changes. Each and every run of a report must be reviewed to verify that it ran correctly. The level of review of a completed production run that is required depends on the risk of the report and the possible impact of the changes to the data.

For low- and average-risks reports, it may be sufficient to verify that the job ran without any errors and to give a cursory review of the results. Determining that the job ran without errors usually involves scanning the SAS log for errors, warnings, and other messages that may indicate an unexpected problem. This scanning can be automated as part of your PROD environment, as long as the automation will also automatically notify the responsible people that the report failed. The review of the results will verify that everything “looks right.” It may also include verifying known calculations, such as patient counts.

Your PROD environment should enable you to quickly determine which SAS program, SAS Macros, and set of input data were used to generate a particular set of results. This process includes the ability to determine which versions of these files were used, thus enabling you to unambiguously trace data from source through the results.

For high-risk reports, in addition to the log scanning, the Manager (or qualified designee) must also review the SAS log and results, and then sign the results themselves to indicate that they are ready for consumption. This review and sign-off does not need to occur for each and every execution of the report. But the results are considered to be draft until they are signed, which should severely limit the use of those results.

While beyond the scope of this paper, which focuses on producing a single report, your organization should compare the results across all of the reports. The goal of this cross-checking is to verify that all of the different reports are producing results that are consistent, thus validating the statistical methods used in the different reports, and ensuring that the calculations were performed with the correct set of data and algorithms. This cross-checking should be performed regardless of risk.

MAINTENANCE PHASE

In the SAS Regulatory Compliant Process, maintenance simply restarts the development cycle. The code is checked-out of production and returned to development. The requirements are the requested changes. The level of risk is determined based on the changes that are being made. The new risk level does not have to be the same as that of the original report; although the original risk level should be a factor in setting the new risk level. From this point on, the process is the same as that which you used to develop the code in the first place.

It is left up to you to determine if the changes are best made in the TEST or DEV environment.
If the original report is producing incorrect results, then you may need to lock any production results that may have been created in order to prevent them from being used. For example, in SAS Drug Development 3.0, you could temporarily change the permissions to deny access to the results until corrected results are available.

SAS DRUG DEVELOPMENT 3.0

The SAS Regulatory Compliant Process can be implemented using traditional SAS programming environments and manual, often paper-based processes. Alternatively, SAS Drug Development 3.0 has been designed to provide all of the tools you need to meet the compliant processes described in this paper, and to do so electronically. SAS Drug Development 3.0 provides check-in / check-out, versioning, audit trails, security controls, and electronic signatures, all within a controlled information management repository. Macro libraries can be developed and controlled, yet remain accessible to all calling programs. Development, production, and testing instances are easily maintained, and the system is fully accessible through Internet Explorer—providing a user-friendly way of developing, finalizing, and distributing research results. Importantly, all statistical results can be easily traced to the version of the code used to produce them, and to the version of the data that drove the results. Thus, the lineage of all programming results is never in question.

OTHER ISSUES

STANDARD REPORTS AND SAS MACROS

It is certainly the case that developing standards in both data and reports means that you may not have to keep developing the same report over and over again. But it is seldom recognized that this comes with a cost. If a report is going to be used more than once, the risks associated with that report go up too. For example, a report that may have been a low risk as a report when it is used on once on a single project may be considered to be a high risk, when that same level of risk is multiplied by using it across many projects. This is true with SAS Macros too, since they are reused by multiple reports.

The benefits of standardization and reuse almost always outweigh the costs. But the costs are not insignificant. Standard reports and SAS Macros may include having to write more complex requirements, code, test data, error checking and handling; additional validation and testing; writing user documentation, and conducting user training and even internal sales and marketing.

Once a SAS Macro has been signed off as production, any reports that use that SAS Macro do not need to specifically revalidate the SAS Macro, but ultimately you, as the author of the report, are responsible for verifying that the results produced by your report are accurate.

SAS Drug Development 3.0 makes writing standard SAS programs easier by providing a way for the SAS program to prompt for input parameters when the program is executed. These input parameters can be defined ahead of time by creating a batch job using the Job Editor.

TEST DATA

Typically, the test data is stored in the **sasdata** folder in the project’s TEST environment. This folder should also include any documentation of the TEST data and/or programs used to create or update the test data. If the test data is specific and might conflict with the test data used with other reports, then they are stored under the report’s test folder under the TEST environment.

When the report’s risk is low or average, it is my opinion that only the TEST environment is of regulatory interest in terms of test data. Obviously, the PROD environment uses real data, and does not require test data. In general, the test data used in the DEV environment does not need to be kept or managed, since the formal testing will occur at TEST anyway.

For high-risk reports, any required developer test files should be copied to the report's test folder under the TEST environment when the SAS program is copied from the DEV environment.

The following table describes some common strategies for creating test data.

Strategy	Comments
Use or copy all of the production data in its final state	This is the easiest and best possible test data, because all of the potential data problems are known when writing the code. The problem is that this is seldom practical. It is not often that all of the data are available at the time the report is being written. If the data are particularly large, it may be impractical to develop against the entire database.
Use or copy a subset of the production data	This is easy, but runs the risk of not fully testing your code, because the missing data may contain unforeseen data problems. The data may be partial because the complete set of data are not yet available, or in order to reduce the volume of test data.
Use or copy data from a similar study	This is easy, but the quality of the test data depends on how similar the study data are. It also runs the risk that the data may contain unforeseen data problems. This is particularly useful when there is no real study data available when you write the report.
Create your own test data	This is the most difficult method, but it can be combined with any of the other methods. It has the advantage of being able to test conditions that may not exist in the data that is available for testing. If you expect your code to be reused, then you will need to create your own test data to test against a greater set of possible data problems than any of the other methods is likely to provide.

EXTRACTION, TRANSFORMATION, AND LOADING (ETL) VS. REPORTING

Extraction, Transformation, and Loading (ETL) is a common task for SAS programmers. This can be performed using the SAS Regulatory Compliant Process as it is presented in this document, but I believe that it would be best to tailor the SAS Regulatory Compliant Process to address the issues that are unique to the ETL process. For example, the ETL process will entail more requirements gathering, which must include determining which data transformations are required. Assigning a risk to an ETL process requires very careful analysis of how the data will be used. By default, all ETL processes are considered to be high risk.

Note that one of the goals of ETL processes is to be able to unambiguously trace data from source through the results. This implies that the ETL process will need many of the same kinds of tools that are included in the Regulatory Toolbox.

USING THE RESULTS

The production of the results is not the end of the story. Those results must be used in some way. Most often this means that the results will be included in a document. While documentation is beyond the scope of the SAS Regulatory Compliant Process, SAS Drug Development 3.0 does facilitate this task by providing a centralized, Web-based storage location; security; versioning; check-in / check-out, audit trails, and the ability to automatically notify others when a result has been updated.

CONCLUSION

Developing any computer program is a complex, difficult, and laborious task. SAS-based reporting is not an exception to this rule. It is necessary to follow a rigorous process to ensure that the results produced are accurate and reliable. Because of this requirement, the emphasis must be on making that process as efficient as possible.

The SAS Regulatory Compliant Process is a rigorous, efficient, and regulatory process that is tailored for producing a large number of reports. Not all reports have the same level of risk and therefore they do not require the same level of validation built into the process. The SAS Regulatory Compliant Process achieves much of its efficiency by having a complete set of regulatory compliant tools. These tools either support or automatically perform many of the tasks that are required by the SAS Regulatory Compliant Process.

REFERENCES

CR-CSV Working Party (2004). *Computerised Systems Validation in Clinical Research, A Practical Guide*, 2nd Edition, United Kingdom.

FDA, 21 CFR Part 11, Electronic Records; Electronic Signatures; Final Rule. Federal Register Vol. 62, No. 54, 13429, March 20, 1997.

FDA, General Principles of Software Validation; Final Guidance for Industry and FDA Staff Document, January 11, 2002.

Helton, Edward D., Halley, Patricia B., and Handelsman, David D. "SAS Solutions for Addressing 21 CFR Part 11 Compliance—The P21 Biomedical Knowledge Platform," *Proceedings of the Seventeenth Annual SAS Users Group International Conference*.

Nelson, Greg (2004). "SASUnit: Automated Testing for SAS," *PharmaSUG 2004 Conference Proceedings*.

CONTACT INFORMATION

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

Chuck Reap
SAS Institute
SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-0970
Fax: (919) 531-0700
Email: chuck.reap@sas.com
Web: http://www.sas.com