Learning from Experience Requires a Collaborative Team Effort: an Integrated-Software Solution for Automated and Reliable Distributed Reporting System.

Noga Meiry Lewin, The Emmes Company, Aimee Wahle, The Emmes Company; Amarnath Vijayarangan, Emmes Services Pvt Ltd; Abigail G. Matthews, The Emmes Company

ABSTRACT

Ten years ago, our reporting team developed a web report system to present the status of ongoing clinical trials for the National Drug Abuse Treatment Clinical Trials Network. Different programmers contributed to a growing number of electronic data capture (EDC) platforms and studies, each with its own requirements, with all reports uploading to a common website. A common setup program standardized the programming process. Nightly one program ran to generate all reports for all platforms and studies, followed by a summary program that uploaded reports to the website. Any errors were investigated and corrected, and the program reran.

As the number of active studies increased, the advantages of using one program to run everything were outweighed by generation time and debugging, and an error in one study prevented other studies from being updated.

With the addition of new EDC platforms, our challenge was to create a reliable, multiuser SAS system to produce web reports for many studies on multiple EDC platforms that adhere to the following, sometimes contradictory, requirements: reliable, standardized, flexible, and programmatically and computationally efficient.

We developed a new system that satisfies these requirements:

(1) while studies are programmed using standard processes, they run independently on different PCs so that failure in one does not affect others; (2) for each study, programs are assigned for submission using a batch file-creating excel spreadsheet that drives the process and serves all aspects of documentation; (3) each step only executes if the previous one succeeded; (4) the batch file runs all programs separately; (5) an initialization program checks that data was updated, and resets and updates the environment's permanent libraries, for the reports that follow, increasing efficiency; and (6) finally, a program checks all required outputs, creates a consolidated PDF report upon success, uploads reports to the website directory for posting, and sends a status email to relevant study teams. If a problem is detected in any stage, only the culprit program is corrected and resubmitted, followed by the web upload program. A global summary program across all studies and platforms sends an overall status update daily to key staff.

INTRODUCTION

The National Drug Abuse Treatment Clinical Trials Network has contracted Emmes to serve as the Data and Statistics Center (DSC) over ten years ago. Included in the scope of work is to design and implement a daily Trial Progress Reporting System which had in its core a set of standard reports. The trials consisted of a small number of studies that had many common traits, and most of the reports were uniform across studies. The sponsor had a very clear set of requirements and specifications and all studies adhered to them.

Based on this framework, our reporting team developed and implemented a web reporting system with the following specifications:

- different programmers contributed their reports for studies, with all reports uploading to a common website;
- common setup program and templates standardized the programming process and relieved programmers from repeating common code;
- an overall report program generated all reports for all platforms and studies, and runs nightly; and
- a summary JAVA program follows nightly that uploads the reports to the website upon successful completion of the overall report program.
- If there was an error found with the overall report program, a senior programmer investigates the error, corrects it, and then reruns the overall report and JAVA programs.

The CTN portfolio of trials has evolved over time and thus the needs of the web reporting system changed. While we started with a small number of similar studies, the number of concurrent studies tripled, and the studies were very different from one another and substantial customization was needed to meet the needs of each investigative team. As the web reporting system attempted to adapt to the new paradigm, the advantages of using one program to run everything were outweighed by the growing number of studies and increased heterogeneity among them. The issues we encountered were as follows:

- adding a new study to the web reporting system demanded increasingly more time;
- the run time was too long;
- debugging of one long, complicated program became more cumbersome and having to split the log further complicated debugging; and
- an error in one report in one study prevented all study reports from being updated since they were all part of one overall report program.

Our goal was to create a reliable, multiuser, multi-platform study SAS® reporting system that adheres to the following, sometimes contradictory, requirements:

1. *Standardization*: the process, data, formats and a large subset of reports should be standardized to the maximum extent possible.
2. *Flexibility*: each study requires customized reports, and the contributors have different programming skills and styles. Therefore, the web reporting system should be as flexible as possible.
3. *Programming efficiency*: save programming resources by using macros for standard reports as often as possible.
4. *Independent implementation*: running failure for one study should not impact other studies.
5. *Implementation efficiency*: save running time and turn-around time. Programs for each study should run quickly and if there is an issue, time to debug and posting of the complete set of study-specific reports to the website should be minimal.
6. *Independent posting*: for each study, upload a complete set of reports only after all its reports run successfully.
7. *Streamlining process of adding studies*: programming and testing time, as well as completion time, should not increase as new reports and studies are added to the system.

8. *Documentation*: there are many platforms, studies, and programs (one for each report generally). Therefore, a reliable and user-friendly documentation system is key, with self-documentation preferable.
9. *Summary of results for entire process*: because the solution is a distributed system, a single status summary report is necessary, particularly for individuals who require information on the status of all studies and platforms.

THE SOLUTION

1. STRUCTURE

The study data flow consists of data update, reporting and web upload. Therefore, the reporting system will first verify that data update was successful, and only then will the reports run. To insure the integrity of the reports, uploading reports to the website will only occur after verifying that all reports for the study ran successfully. Thus, all reports on the website are based on data that were updated at the same time.

For each study, three types of programs run in sequence but independently using a batch file. Because reports are independent of each other, running them in batch prevents one report from stopping the other reports as in the case of an include program. The three types of programs are described below.

1. Initialization program:
   - This is the key to having the best of both worlds:
     - It only runs once before all programs.
     - It sets up the environment, to save running time and programming resources.
     - All programs have access to the environment even though they run independently, but they do not need to run the time-consuming setup.
   - It does the following:
     - Verifies that the EDC data has been updated.
     - Initializes and creates permanent (on disk, as opposed to 'work' directory) format catalog, and specifications datasets and additional requirements to be available to all report programs.
     - Sets these libraries to read-only.
2. Report programs: the study-specific report programs run after, using the environment established by the initialization program. Each program creates a separate log and only runs upon updated data and successful initialization. The code below is an example of a standard program: it calls a setup program which points to the common environment, and then executes a standard macro. The report programmer just needs to change the first four lines.

```
/*************************************************************
Program: age_00XX_prod.sas
Author: John Doe
Date::  December 2019
Description:: create age report
SAS Version:: sas9.4m4
Gemini ticket:
Input Data:: enroll, sitename, dem
OUTPUT:: &prot._age.pdf
Revision::
*************************************************************/

ods listing close;
%LET PROT=00XX;
%LET PLATFORM=YYY;
%let prod_dev=_prod;/*suffix for prod or dev programs. required.*/
```

```
%let prod_dev_path=prod;
/*do not modify the next three lines*/
%let root=G:\xxx\TPR\yyy;
%let prot_path=&root\&prod_dev_path\&PLATFORM\&Prot;
%INCLUDE
"&prot_path\programs\&platform._setup_&PROT.&prod_dev..sas"/SOURCE2;
/*call your macro*/
     %M_age&prod_dev(PROT=&PROT)
```

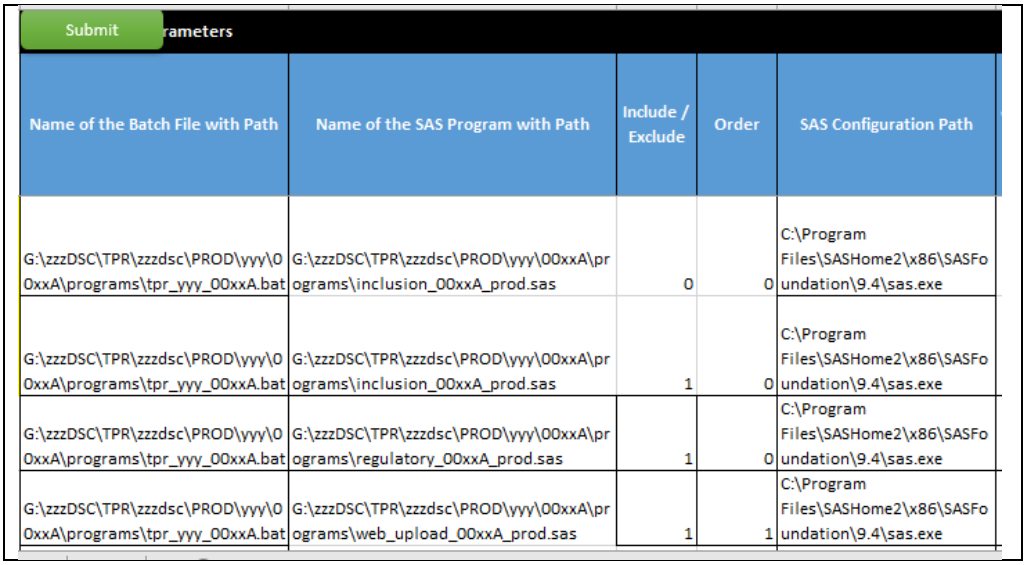
3. Web upload program: this program runs last. It summarizes the distributed system results and does the following:
   - determines the success and completion of the reports
   - collates the individual PDF reports to two different PDF consolidated reports according to specified orders – one for standard reports and one for study-specific reports
   - uploads the individual and consolidated reports to a directory for the website
   - sends a status email of success or failure with detailed information on which report failed to the programming team.

   Note that if there is a failure, a programmer follows up by checking the log of the failed report and fixing the program. They then run this report only and re-run the web upload program.

2. USE OF EXCEL TO CREATE THE BATCH FILE AND SERVE AS CENTRAL DRIVER OF THE PROCESS

An Excel spreadsheet is used to create the batch file described above. This approach is ideal for documentation and tracking purposes given the large number of programs involved. It also facilitates the addition or removal of certain reports as a study progresses and controls the order of reports in the consolidated PDFs. The approach implemented in Excel is described below.

1. Programmers include their SAS programs by filling in three types of columns into an excel spreadsheet:
   a. Columns that serve as input to the batch file (Figure 1):
      - Program name and path
      - SAS executable path
      - Batch filename and path
      - Include/exclude column that specifies whether to include the program in the batch file which determines if the program is run. This way, programs can easily be turned off and on without removing them from the file.
      - Their running order. Usually this will be 0, since the reports are not dependent on each other. The web upload program is assigned the order of 1 to run last.

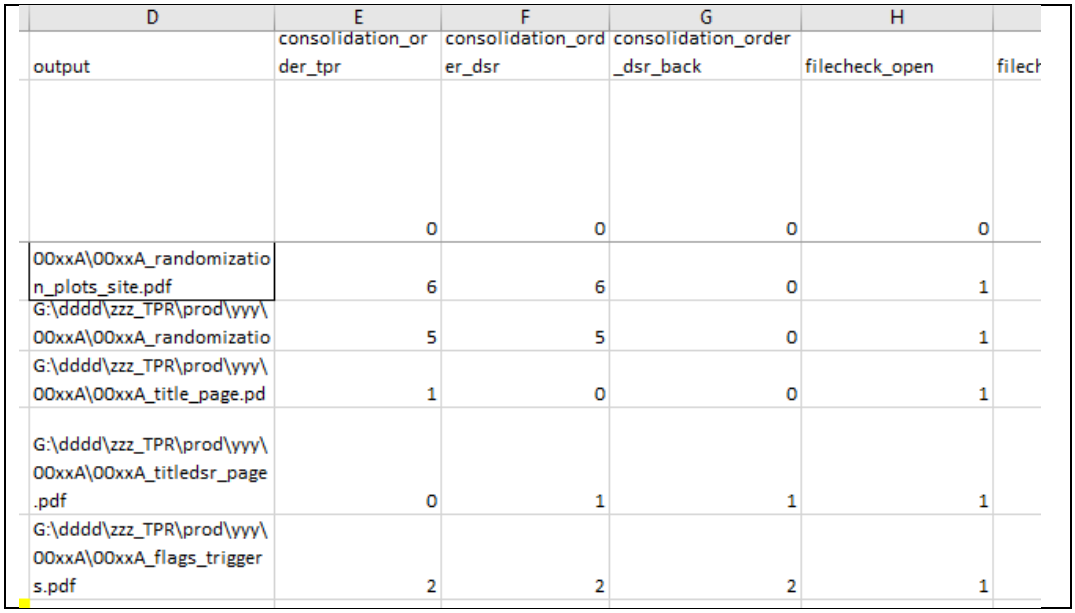


Submit rameters

Name of the Batch File with Path	Name of the SAS Program with Path	Include / Exclude	Order	SAS Configuration Path
G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\tpr_yyy_00xxA.bat	G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\inclusion_00xxA_prod.sas	0	0	C:\Program Files\SASHome2\x86\SASFoundation\9.4\sas.exe
G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\tpr_yyy_00xxA.bat	G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\inclusion_00xxA_prod.sas	1	0	C:\Program Files\SASHome2\x86\SASFoundation\9.4\sas.exe
G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\tpr_yyy_00xxA.bat	G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\regulatory_00xxA_prod.sas	1	0	C:\Program Files\SASHome2\x86\SASFoundation\9.4\sas.exe
G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\tpr_yyy_00xxA.bat	G:\zzzDSC\TPR\zzzdsc\PROD\yyy\00xxA\programs\web_upload_00xxA_prod.sas	1	1	C:\Program Files\SASHome2\x86\SASFoundation\9.4\sas.exe

Figure 1. Columns in the Excel sheet that serve as input to the bat file

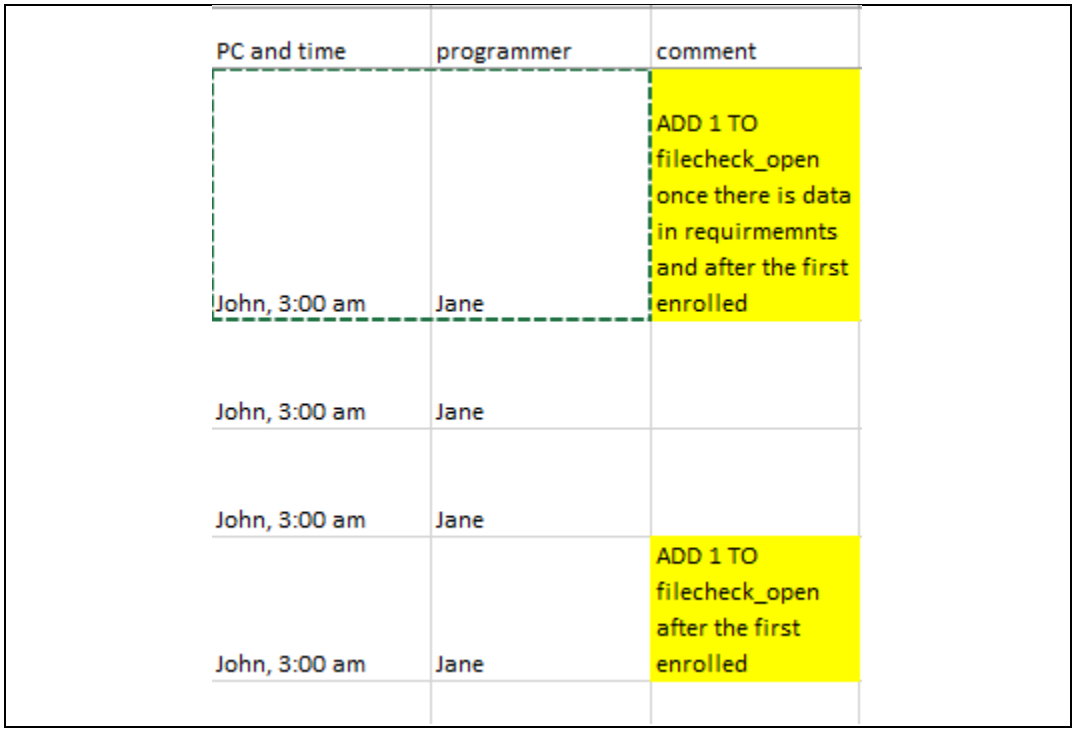
b. Columns that serve as input to the summary program (Figure 2):
   - Output filename
   - Required output for success of web upload program (filecheck_open)
   - Whether it needs to be added to the consolidated reports
   - Order of the reports in the consolidated reports



D	E	F	G	H	
output	consolidation_order_tpr	consolidation_order_dsr	consolidation_order_dsr_back	filecheck_open	filech
	0	0	0	0	
00xxA\00xxA_randomization_plots_site.pdf	6	6	0	1	
G:\dddd\zzz_TPR\prod\yyy\00xxA\00xxA_randomizatio	5	5	0	1	
G:\dddd\zzz_TPR\prod\yyy\00xxA\00xxA_title_page.pd	1	0	0	1	
G:\dddd\zzz_TPR\prod\yyy\00xxA\00xxA_titledsr_page.pdf	0	1	1	1	
G:\dddd\zzz_TPR\prod\yyy\00xxA\00xxA_flags_triggers.pdf	2	2	2	1	

Figure 2. Columns that serve as input to the summary program

c. Columns that serve only for tracking purpose (Figure 3):
   - Programmer name
   - Which team member's PC the program runs on and the scheduled time
   - Comments



PC and time	programmer	comment
John, 3:00 am	Jane	ADD 1 TO filecheck_open once there is data in requirmemnts and after the first enrolled
John, 3:00 am	Jane	
John, 3:00 am	Jane	
John, 3:00 am	Jane	ADD 1 TO filecheck_open after the first enrolled

Figure 3. Columns that serve only for tracking purposes

2. Finally, once they click the 'submit' button, an imbedded Visual Basic APP macro creates the batch file.

3. RUNNING REPORTING PROGRAMS USING THE BATCH FILE

Below is a description of how the batch file functions in the report generation process.

1. Within the batch file, call this sub to run each program. It uses the information in the Excel spreadsheet to create the parameters (enclosed between %%):

```
:sub
IF EXIST "%SASLOC%\%PROGRAM%" (
  "%SASPATH%" -CONFIG "C:\Program
Files\SASHome2\x86\SASFoundation\9.4\nls\en\sasv9.cfg"  -sysin
"%SASLOC%\%PROGRAM%" -print "%LSTLOC%\%PROGRAM%.lst" -log
"%LOGLOC%\%PROGRAM%.log" -ICON %MEM%
) ELSE (SET ERROR=1
  ECHO file %SASLOC%\%PROGRAM% does not exist

)
exit /b
```

2. Run the initialization program using this code in the batch file:

```
(
SET PROGRAM=%platform%_INITIALIZE_%SUFFIX%
REM INITIALIZE ENVIRONMENT. IF NOT SUCCESSFUL DO NOT CONTINUE (CHECK IF
INIT_Y WAS CREATED)
REM ***************************************************************
attrib -r %root%\initdata\*.* /s
del "%INIT_Y%"
call :sub
if NOT exist "%INIT_Y%" GOTO EXIT
```

3. Next, call the report-generating programs. The program names are extracted from the information in the Excel spreadsheet by the Visual Basic App.

```
REM ****************************************
set PROGRAM=RANDOMIZATION_plots_00xxA_prod.sas
REM ****************************************
call :sub

REM ****************************************
set PROGRAM=title_page_00xxA_prod.sas
REM ****************************************
call :sub
```

4. Finally, call the web upload program:

```
REM ****************************************
set PROGRAM=web_upload_00xx_prod.sas
REM ****************************************
call :sub
```

4. USE DAILY STATUS EMAILS FOR MONITORING

Two types of emails provide status reports to staff:

1. Daily study status email - Figure 4 is an example of a success email, and Figure 5 is an example of failure intended for programmers. They also provide information on reports found in the directory that were not required. You can see in the example how this helps with the debugging – the required report name should have been the name of the extra file.

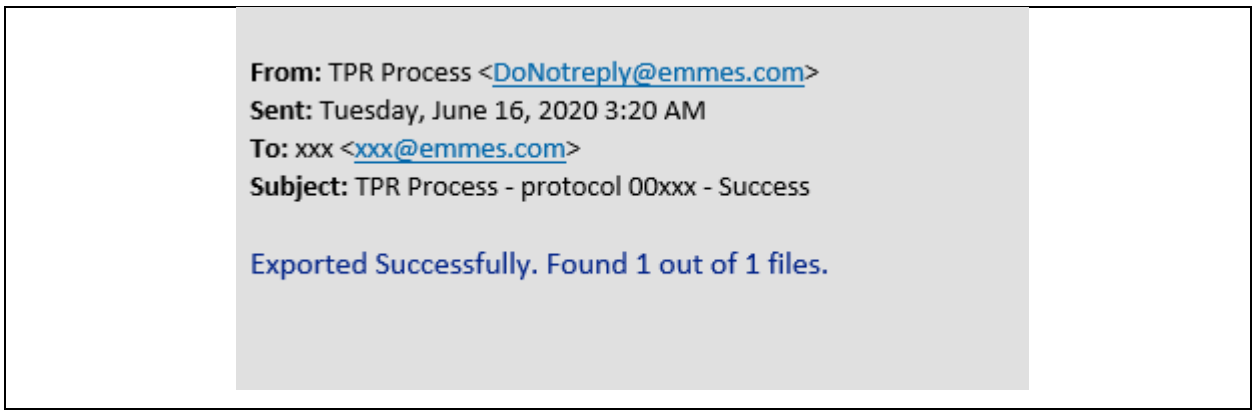


Figure 4. Study Status Report Email - Success

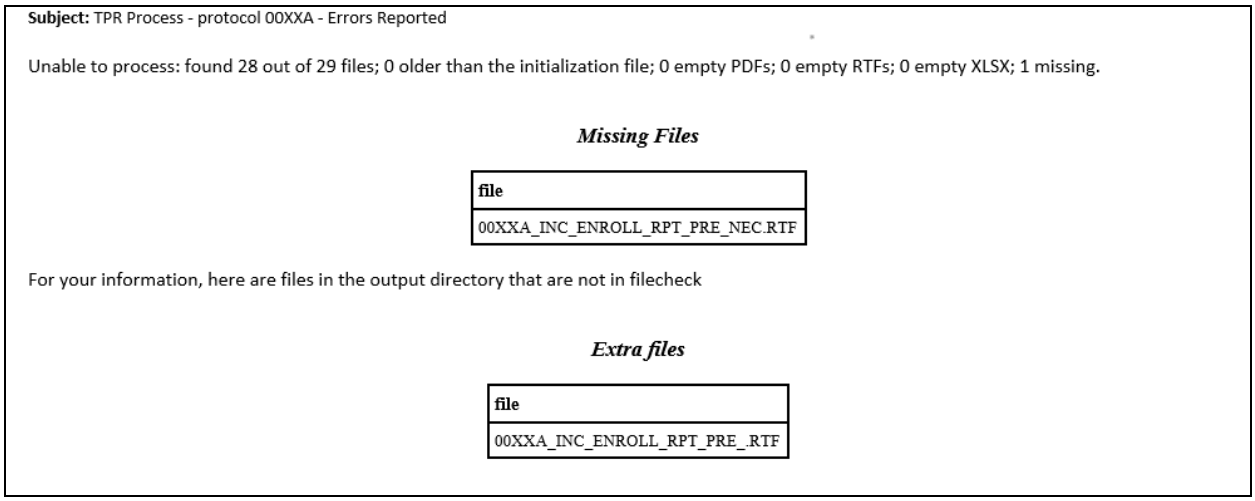


Figure 5. Study Status Report Email - Failure

2. Overall status email - an overview program scans all directories for platforms and studies and sends a status email to all DSC staff including management. Figure 6 and Figure 7 depict examples of the success and failure overall status emails. In Figure 7 for study "00XX" – a failure early in the data update stops reporting, reflecting the principle that each step executes only if the previous step completed successfully. (TED and CUP are data update processes, PRIV-UNPRIV are programs that manipulate the updated data, TPR – web reports process).

Subject: Status of CUP, TED, TPR - 16JUN2020 - Success

Status of CUP, TED, TPR for 16JUN2020:

	Data Update Status			
Platform	**CUP\TED Status**	**Priv-Unpriv Status**	**Protocol**	**TPR Status**
XXD	Success	Success	00XX	Success
XXC	Success	Success	00YY	Success
XXE	Success	Success	011X	Success
XXX	Success	Success	00XY	Success

Figure 6. Overall Status Report Email – Success

Subject: Status of CUP, TED, TPR - 03JUN2020 - Failure

Status of CUP, TED, TPR for 03JUN2020:

	Data Update Status			
Platform	**CUP\TED Status**	**Priv-Unpriv Status**	**Protocol**	**TPR Status**
XXB	Success	Failure	00XX	NA
XXC	Success	Success	00YY	Success
XXD	Success	Success	010XX	Failure
XXE	Success	Success	No protocols enrolling yet	NA

Figure 7. Overall Status Report Email – Failure in data update and reporting

5. TEAMWORK IS VITAL

For this system to work, it requires collaboration among different players with different skill sets, namely data managers, statisticians and SAS programmers with different levels of experience, and a Visual Basic programmer. Our work environment is supported by the following practices:

1. regular meetings to discuss updates and questions or issues;
2. ongoing training on how to take advantage of what the system offers (e.g., macro variables, formats);
3. clear roles and backup assignments to allow for continuous coverage;
4. ongoing documentation on how to use the system; and
5. open door approach: no question should go unanswered.

6. PERFORMANCE INDICATORS

Figure 8 is a top overview of the process. We were able to save time on each step:

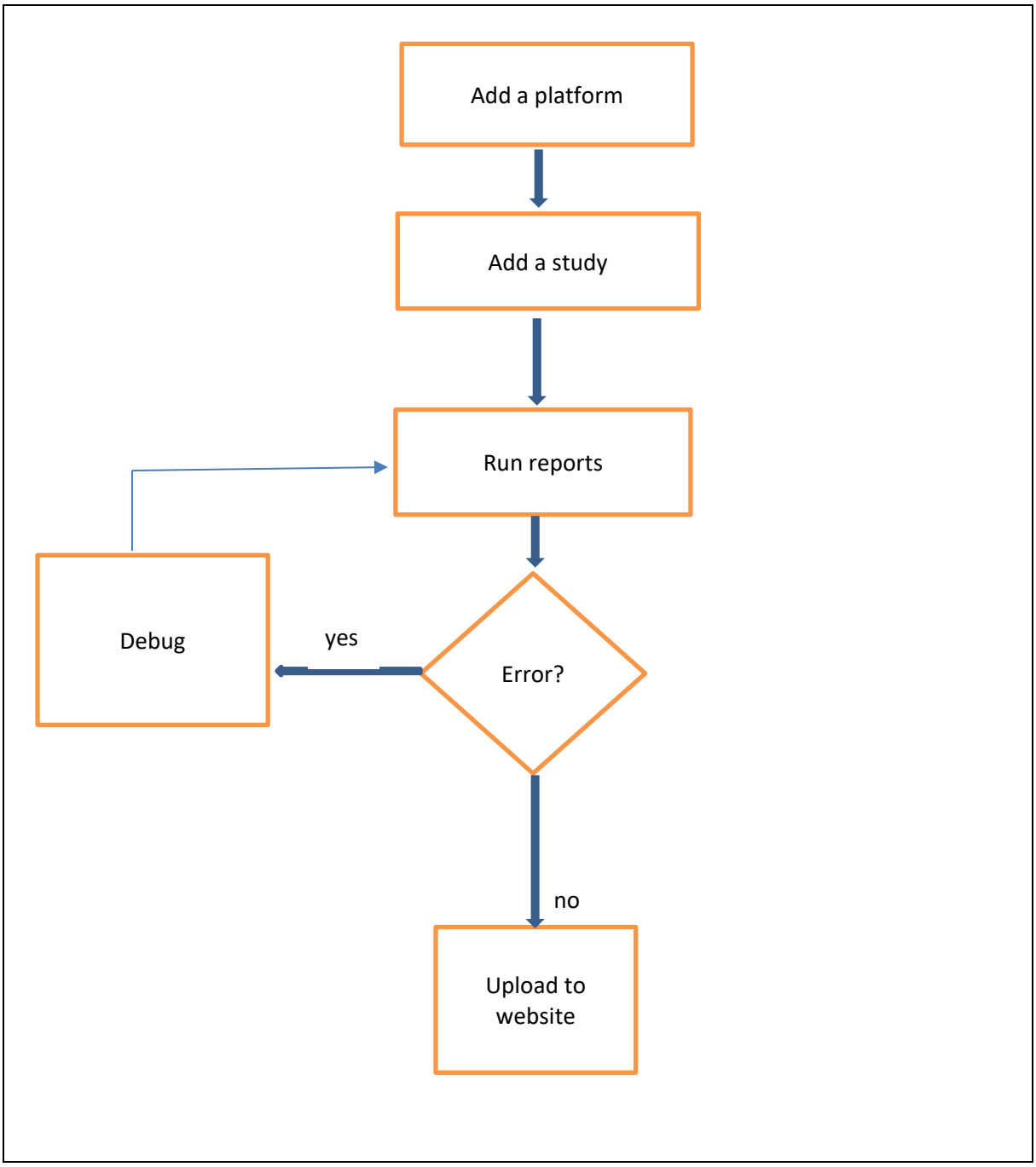


Figure 8: Top View of the Reporting System

The time it takes to add a study to the old system grew from ten hours when we seldom added studies ten years ago, to twenty hours, and just prior to revamping the web reporting process, it had grown to forty hours. After implementing the process described here, it takes four hours to add a new study to this system. It had taken an hour and a half to run and upload a complete set of all studies to the web in the previous system, while now, in many cases, it takes a full report set five to ten minutes to run and upload to the web.

CONCLUSION

The development of our web reporting system is a process of continuous evaluation. New methods build on the ones that exist and had proven themselves in order to preclude "re-inventing the wheel" and conserve resources. For example, while writing the paper we learned that we should develop performance measuring tools for accurate monitoring. Participation to the internal Emmes SAS User Group keeps us informed of methods used by other groups supporting different clients within the company, and some of which have been adopted and incorporated into the new process.

The current system has been functioning in production for six months, with another nine months of revamping prior to implementation. Development of programs is not hindered by the growing number of studies and various, complex reporting requirements. Standardization simplifies the programming and allows for easier central monitoring and aggregation.

ACKNOWLEDGMENTS

We would like to thank our colleagues at the Emmes Company for their feedback and encouragement, and especially the web report team of the NIDA CTN Data and Statistics Center. This work has been partially funded by the National Institute on Drug Abuse (contract 75N95019D00013).

RECOMMENDED READING

- *Base SAS® Procedures Guide*
- *Carpenter's Complete Guide to the SAS REPORT Procedure (SAS Press) Pap/Cdr Edition, by Art Carpenter*
- *Carpenter's Complete Guide to the SAS Macro Language, Third Edition 3rd Edition, by Art Carpenter*

CONTACT INFORMATION

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

Noga Meiry Lewin
The Emmes Company
nlewin@emmes.com

Aimee Wahle
The Emmes Company
awahle@emmes.com

Amarnath Vijayarangan
Emmes Services Pvt Ltd
avijayarangan@emmes.com

Abigail G. Matthews
The Emmes Company
amatthews@emmes.com