PharmaSUG 2016 - Paper PO21

SDTM Automation with Standard CRF Pages

Taylor Markway, SCRI Development Innovations, Carrboro, NC

ABSTRACT

Much has been written about automatically creating outputs for the Study Data Tabulation Model (SDTM). This paper provides brief descriptions of common approaches in practice and details an approach to automatically create SDTM outputs when case report form (CRF) pages are uniquely identifiable.

The process uses a fixed SAS® program structure that allows programs to be automatically constructed and deconstructed. This approach depends on three assumptions: 1) all CRF pages are identifiable, 2) all CRF pages that have the same set of identifiers are unique, and 3) all CRF pages are consistently represented in the electronic database. At SCRI, the statistical programming team and the data management team worked to meet these assumptions while designing the standard database.

After meeting these three assumptions we can automatically create SAS SDTM programs for each domain. These programs that are easy to read due to their systematic structures. Considering that the programming team is still working directly with SAS, minimal training is needed to implement them. Deviations from the standard CRF or sponsor-specific SDTM mapping are implemented directly in SAS programs without complicated macro calls or outside mapping tools.

INTRODUCTION

This paper describes some SDTM automation techniques and how our approach fits into the existing paradigm. Our approach is detailed, and the results of the initial live cases are reported. This paper assumes the readers are familiar with SDTM.

Our automatically generated SDTM programs take advantage of a metadata-driven approach. However, this paper will not detail the metadata-driven processes of these SDTM programs, as this topic has been covered in previous papers, including “Using SAS® Macros to simplify preparation of SDTM data, Annotated CRFs and Define.xml in a metadata driven environment” by Niels Both (2009).

SDTM AUTOMATION

The creation of the SDTM data sets falls under a type of computing called Extract, Transform, Load or ETL. We first extract the clinical data from the case report forms, central labs and other vendors. Next, we transform it to the SDTM standard. Then we load it to a final location to be used for analysis and reporting.

SDTM programs are an ETL tool. Every time we write an SDTM program that we have written before, we are rebuilding that tool much like a carpenter who builds a new hammer each time he builds a desk. This is wasting time and we need to automate.

After surveying the landscape we found that there are many technologies and approaches available for SDTM automation. Black-box mappers usually have a graphical user interface (GUI) and do not require the user to input code. Metadata sources such as the CDISC ODM are typically used to automate the mapping in these systems.

At the other end of the spectrum, we have whole SDTM programs that are recycled. On this side of the spectrum, all user inputs are in SAS code and each SDTM domain starts as the same version copied from the standards and is updated for each particular study.

Many programmers informally practice this second type of automation when they copy SDTM programs from one study to the next.

Figure 1 shows a graphic lay out of the landscape. This is, of course, an oversimplification, but it shows what our method is and is not supposed to be. The black-box method is typically automated the most and also creates restrictions and may not even use SAS. The domain level is the most flexible, yet it usually requires the most manual interventions.

So our module approach falls between these two extremes.

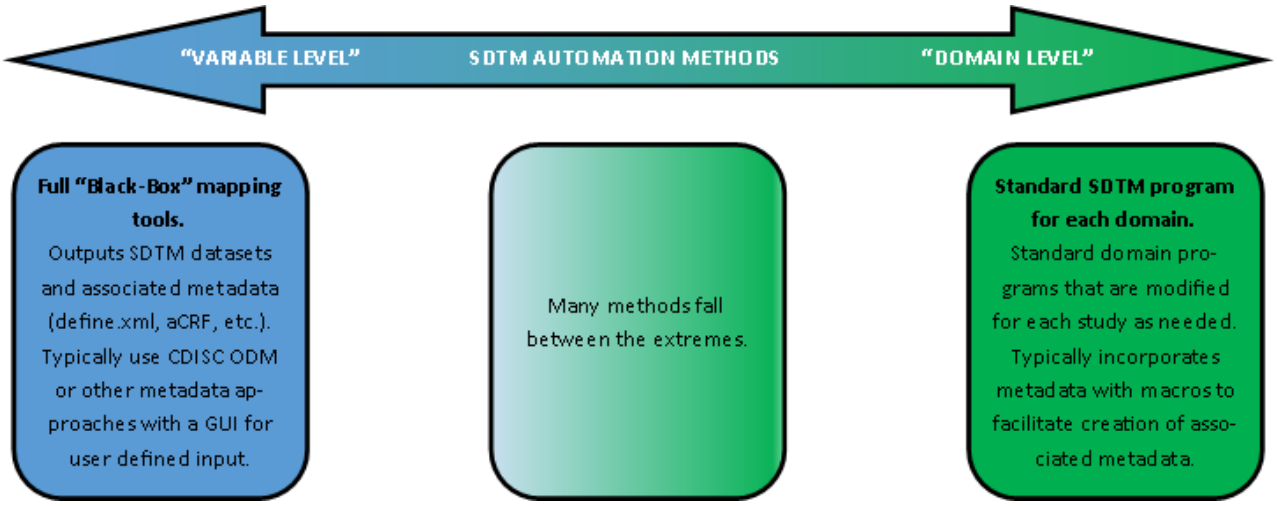

Figure 1. SDTM Automation Landscape

THE MODULE APPROACH

We call our approach "the module approach" because at its core are SAS code fragments called modules. Each module is associated with a CRF page and an SDTM domain. Each SDTM domain also has a SAS program shell associated with it, and it is called a domain program. By putting the module and domain programs together, we can create SDTM programs for any combinations of the standard CRF pages.

Think of this method as programming each CRF page to an SDTM domain in isolation. This is different than the usual approach where we think of programming an SDTM domain from all the CRF pages that map to that domain.

This is a subtle distinction, but it allows us to separate the study-specific details from the CRF and the SDTM programs.

All organization are going to face their own set of requirements when implementing new standards. To meet our goals for SDTM automation, we required minimum standards for our automation system:

1. Must be user friendly.
2. Must have a transparent output.
3. Must be easily maintained and updated.
4. Must work with any combination of standard CRF pages.

At SCRI, the statistical programmers who create analyses and reporting outputs also create the SDTM data sets. Since our users would be professional SAS programmers, any tool that limits the functionality of SAS would be a restriction rather than a boon. Transparency means that no black-box solutions or complicated macros can be used to create SDTM output.

In terms of maintenance, any solution that does not expand, grow, or adapt to changes is dead on arrival. In contrast, we require a solution that can be improved. It would also need to work for any combination of our standard CRF pages. No partial solutions are acceptable. We must have something that creates SDTM output without requiring user input when CRF pages are 100% standard.

For this reason, the module approach creates SAS SDTM programs that can be fully edited in SAS and validated in the typical double programming fashion that many sponsors expect. However, the module approach provides greater efficiencies by incorporating alternatives to the traditional independent double programming that will be discussed later.

MODULE METHOD DETAILS

The Module method will be explained in three parts. First, I will go over the standard CRF structure. Second, I will cover the standard structure of the SDTM programs. Third, I will show how the standard programs are built and finalized when there are deviations from the standard CRF pages.

The module method of automation has two main parts; module programs associated with each CRF page and the domain programs that contain the code that will always be part of an SDTM domain. When the module and domain programs are combined, the final product is the SDTM program.

STANDARD CRF

For our module approach to work, we need to confirm three assumptions 1) all CRF pages are identifiable, 2) all CRF pages have unique identifiers, and 3) all CRF pages are consistently represented in the electronic database.

At SCRI, we have used a template identifier and version date to uniquely identify CRF pages. Figure 2 is a simple example of an early clinical trial with the CRF template ENRL-01 and Version Date: 1774-10-25. The template and version date are stored as electronic metadata, and the information will be read and used to automatically combine the module programs into the domain programs to create the final SDTM programs.

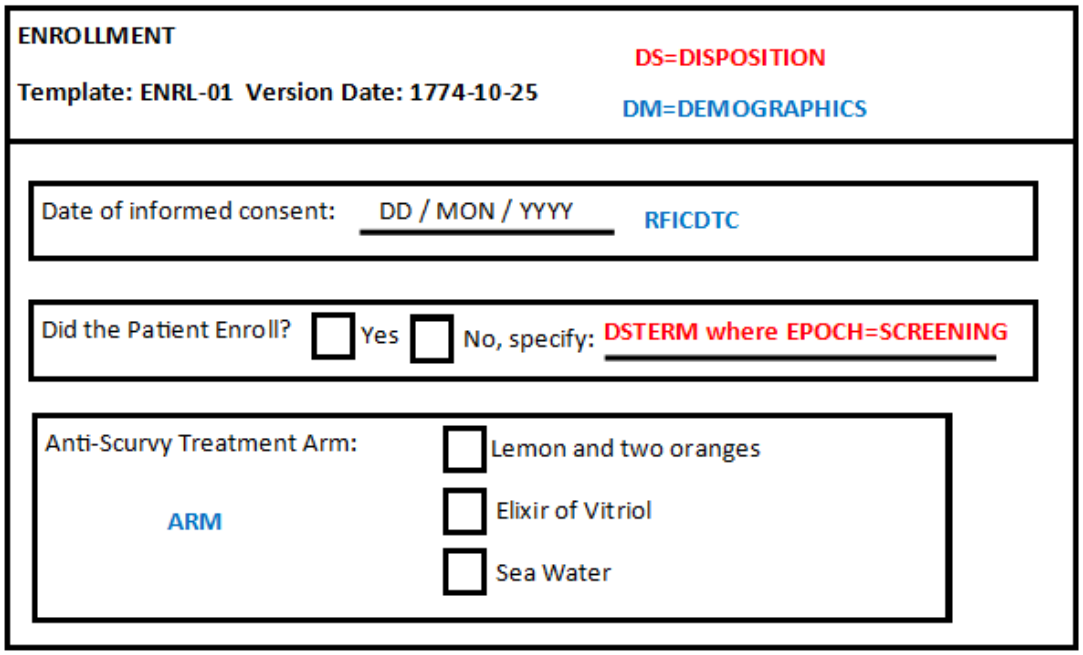

Figure 2. Sample CRF Page with Template and Version

SDTM PROGRAMS

To see how the module and domain programs work together, we will look at a SDTM program created using the Module method. All the programs are structured exactly the same way so that they can be programmatically constructed and deconstructed. Our approach creates an SDTM program that consists of five components:

1. Initialization of raw data
2. Mapping CRF pages to SDTM (the modules)
3. Combining all of the modules
4. Additional/Complex Derivations (the domain program)
5. Output

All five pieces of the SDTM program are automatically brought together for each study that uses the standards.

A standard macro is used to dynamically create components 1 and 3 by initializing the data and combining the modules. This ensures that the mapping of the CRF pages to the SDTM (component 2) will always have a consistent start and finish and allows the macro to copy any modules into the domain program.

Component 4 is the heart of the domain program as it contains the derivations that are required for the domain regardless of the CRF pages used. For example, derivations such as study day, baseline, and visit mapping take place in component 4.

Component 5 is part of the domain program and outputs the final data set. It applies the final data set's metadata.

All SDTM programs that use CRF data are organized this way.

Initialize the Data

After establishing a standard header, we initialize the data. This step is critical as it sets up the input for the modules in section 2. We use a "fill-in-the-blank" approach and create an empty SDTM data set shell that our module programs then fill. Other approaches could also be taken.

In Figure 3, you can see the creation of the demographics shell via a macro (%mu_shell) and the use of a simple initialization macro (%ms_initial) to perform the repetitive tasks of ensuring that the data set assumptions for the module programs are correct. To ensure that all mappings are explicit in the final program, we renamed all of the raw data variables that have SDTM variable names with a leading underscore.

```
21*------------------------------------------------------------------------------*;
22* #1 initialize data *;
23*------------------------------------------------------------------------------*;
24*------- imports the shell data set structure  ---*;
25*----    and keeps appropriate variables that are in the shell ---------*;
26%mu_shell(domain=dm);
27
28*------- Initialize Standard Identification variables in CRF datasets  ---*;
29*------- Creates STUDYID, USUBJID, SUBJID, SITEID and DOMAIN -------------*;
30%ms_initial(domain=DM, shell=DMx, crf=raw.dm, out=dm,rename=SEX=_SEX ETHNIC=_ETHNIC);
31%ms_initial(domain=DM, shell=DMx, crf=raw.dth, out=dth,rename=);
32%ms_initial(domain=DM, shell=DMx, crf=raw.eos, out=eos,rename=);
```

Figure 3. Initializing the CRF Data

Mapping CRF Pages to SDTM

The modules, are the second part of any SDTM program and where the CRF variables are mapped to SDTM variables. All of the modules are included automatically and each module starts and ends with standard tags. In our case:

```
*---- START: Map <module name> to <SDTM domain> ----*;

<Module code>

*---- END: Map <module name> to <SDTM domain> -----*;
```

Though the modules are included automatically, they must still be written manually they are created. These tags make it easy to identify where the modules begin and end and make it possible to extract modules and save them in our standards library.

This allows a module to be written only when it is first used and facilitates the creation of sponsor-specific standards if changes from our standard SDTM mapping are requested.

```
72*----- START: Map DTH-01 to DM -----*;
73
74data dth_dm (keep=usubjid dthdtc);
75  set dth;
76  DTHDTC = tranwrd(dthdat,'/','-');
77run;
78*----- END: Map DTH-01 to DM -----*;
```

Figure 4. Death Details CRF Page Mapped to DM Domain

Figure 4 shows the mapping from the Death CRF page to the demographics (DM) domain. In the full SDTM program, there is a module for each CRF page mapped to the DM domain. Note that the input data set dth matches the raw data set named for the initialization and the output name is the raw name, underscore, domain name.

Combining the Modules

As shown, all of the modules have programmatically determined output names. Using these standard names, all of the individual modules are combined into one data set. We have a metadata file called the module index that contains a list of all the modules for every CRF page and indicates how they relate to the core domain either as a set or merged. This is automatically generated for each program based on the CRF pages used.

```
166*-------------------------------------------------------------------------------*;
167* #3 combine crf modules *;
168*-------------------------------------------------------------------------------*;
169
170*Create the sorting macro to sort all datasets in the group*;
171%macro _sort(ds=,bv=);
172proc sort data=&ds. out=&ds.;
173  by &bv.;
174run;
175%mend _sort;
176
177*----- Merge Datasets in Group 1 -----*;
178%_sort(ds=DM, bv=usubjid);
179%_sort(ds=DTH, bv=usubjid);
180%_sort(ds=EOS, bv=usubjid);
181%_sort(ds=EX, bv=usubjid);
182
183data dm;
184  merge dm_dm (in=in_dm) dth_dm (in=in_dth) eos_dm (in=in_eos) ex_dm (in=in_ex);
185  by usubjid;
186run;
```

Figure 5. Combining All CRF Modules

Figure 5 shows the merging of the module output to create a consistent domain output. We can set or merge any combination of our CRF pages.

Additional/Complex Derivations

This fourth section is the heart of the domain program. Because the modules contain only the mapping of raw variables to SDTM variables this section contains any SDTM variables that are the result of derivations, for example, study day and the SDTM sequence number. In Figure 6 for the DM domain we have derivations for RFPENDTC, DTHFL, and AGE.

```
188*-------------------------------------------------------------------------------*;
189* #4 additional derivations *;
190*-------------------------------------------------------------------------------*;
191data dm;
192  set dm;
193
194  if dthdtc ne '' then RFPENDTC=dthdtc;
195  else if alivedat ne '' then RFPENDTC=tranwrd(alivedat,'/','-');
196  else if eosdat ne '' then RFPENDTC=tranwrd(eosdat,'/','-');
197  else RFPENDTC='';
198
199  if DTHDTC ne '' then DTHFL='Y';
200  else DTHFL='';
201
202  if length(brthdtc)=10 and length(rficdtc)=10 then
203    do;
204      AGE = floor(yrdif( input(brthdtc,is8601da.), input(rficdtc,is8601da.), 'AGE'));
205      AGEU='YEARS';
206    end;
207
208run;
```

Figure 6. Additional Derivation for DM

Output

The output section contains minimal coding and strictly creates the final output. In our case, this is both a SAS data set and a SAS version 5 transport file. Figure 7 displays this simple step. The macro variables (&domlabl and &keep) were created in the initialization step by the same macro that created the SDTM shell data set, %mu_shell.

```
210*-------------------------------------------------------------------------------*;
211* #5 finalize outputs *;
212*-------------------------------------------------------------------------------*;
213proc sort data=dm out=p_sdtm.dm (label="&domlabl" keep=&keep.);
214  by usubjid;
215run;
216
217*-- putting sdtm dataset into (xpt) transport file format -----*;
218%mtxnspt(dsn = p_sdtm.dm);
219
220%put >>>>>>>>>>>>>>> end of program   <<<<<<<<<<<<<<<<<<<<<<;
221%put >>>>>>>>>>>>>>> run by: &sysuserid.      <<<<<<<<<<<<<<<<<<<<<<<<;
222%put>>>>>>>>>>>>>>> run on: %sysfunc(date(),is8601da.)      <<<<<<<<<;
```

Figure 7. Final Output for DM Domain

COMBINE MODULES AND DOMAINS

The macro call to create our standard SDTM programs needs to specify the version number of the standard CRF used and the SDTM version. Figure 8 contains the actual macro call that ultimately creates over 25 SDTM domains.

```
26%mu_GetStandardSDTM(crfversion=5.0, sdtmversion=3.2);
```

Figure 8. Macro to Automatically Create Standard SDTM

For transparency sake, we split the creation into two separate pieces. This first macro call created a program containing the macro calls that actually created the SDTM programs. This was so manual corrections could be made if a CRF page had incorrect metadata in the EDC system.

Figure 9 contains the final macro calls used to create the SDTM programs. This is run once at the beginning of SDTM production.

Based on the CRF and the SDTM versions the macro will go to a directory and copy the indicated modules and paste them into the domain program. The macro will also create the initialization and combination parts of the program from the module input.

The input for the module parameter is <raw data set name>-<CRF template>. So the DM domain in figure 9 has a raw data set of EXF that corresponds to the CRF template EX-01.

```
%mu_WriteProgSDTM(
  domain=AE,
  modules=AE-AE-04,
  crfversion= 5.0,
  sdtmversion=3.2
);

%mu_WriteProgSDTM(
  domain=DM,
  modules=DM-DM-01 DTH-DTH-01 EOS-EOS-01 EXF-EX-01 EXA-EX-03 FU-FU-01 IE-IE-02,
  crfversion= 5.0,
  sdtmversion=3.2
);
```

Figure 9. Combing Modules to Domain Programs

UPDATING NON-STANDARD PAGES

When creating the study CRF we placed each page into one of three buckets:

1. EXACT: A match to the standard CRF page.
2. MODIFIED: Based on a standard CRF page modified in some way.
3. CUSTOM: A completely new CRF page that is not based on any existing standards.

This category is determined by the template name and the version date (see Figure 2 as an example).

The macro that generates the SDTM program treats the EXACT and MODIFIED the same. It simply copies the associated module program and pastes it into the domain program. If the standard CRF page was a good starting place for the CRF designer, it would be a good starting point for the SDTM programmer.

Custom CRF pages will have to be programmed from scratch as these are not based off of any existing pages.

When the domains and modules are combined into the SDTM programs reports are created that summarize the buckets for each page. These reports are reproduced in Tables 1 and 2. From this information, the programmer can now approximate the effort needed to create the SDTM programs for a study.

RESULTS IN PRACTICE

Our Module method was conceptualized in September and designed and built in October, followed by validation of all our standard CRF pages during November and December.

Since December, we have had the opportunity to test the method on three live studies with great success. It is now being used on all new studies.

Early phase oncology studies are SCRI's specialty, so the three test studies were all oncology and phase I or I/II. The percent of the CRF pages that followed the standards have been 49%, 73 and 75%. The second two studies were similar, so I will only provide details on the first two studies.

STUDY A – 49% STANDARD

The first study to use the standards had 47 CRF pages with 23 pages being an exact match to our standards. The programmer using the standards completed 20 SDTM domains in two days. This was the first time this programmer had used the standards after two one-hour training sessions.

STUDY B – 73% STANDARD

The second study tested had 45 unique CRF pages. Based on feedback from the first test case, we implemented a standard report to help guide the programming effort. When we run the macros against the CRF metadata, it categorizes the pages into three groups: Standard, Modified and Custom.

If a page falls into the standard category, then it is an exact match to the standard CRF page. The metadata for a modified page indicates the standard page it was created from, so we still use the standard module as a starting point, while still needing to update the SDTM program to account for the changes. Custom pages are so unique that they are not based on of any existing standard page, so these pages must be included in the SDTM programs from scratch.

Tables 1 and 2 below are standard reports that are generated after running the macro that creates the SDTM programs. Table 2 has been slightly modified to show where the custom CRF page was mapped. Table 2 also tells us that there are 12 SDTM domains that are 100% standard and will not require manual intervention. Then there are eight domains that require a manual intervention to make updates for modified or custom CRF pages.

Study B – CRF Details

Mapping Result	Number of Pages	Percentage
Modified Standard CRF Page	11	24.4
Standard CRF Page	33	73.3
Custom	1	2.2

Study B – SDTM Mapping Details

Domain	Mapping Result	Number of Pages	Percentage of Domain for the Mapping Result
AE	Modified	1	100
CE	Modified	1	100
CM	Standard	2	66
CM	Custom	1	33
DD	Standard	1	100
DM	Standard	4	66
DM	Modified	2	33
DS	Standard	2	50
DS	Modified	2	50
EC	Standard	1	100
EG	Unknown	1	100
EX	Standard	1	100
IE	Standard	1	100
LB	Standard	8	100
MH	Standard	3	100
PR	Standard	3	60
PR	Modified	2	40
QS	Standard	1	100
RS	Standard	1	100
SC	Standard	1	100

Domain	Mapping Result	Number of Pages	Percentage of Domain for the Mapping Result
SS	Modified	1	100
TR	Standard	3	100
TU	Standard	4	100
VS	Standard	1	100

NEXT STEPS AND VALIDATION

Some may dismiss the Module method as a clever trick at a CRO that specializes in early oncology studies. However, recall that three assumptions had to be met to make this approach work: 1) all CRF pages are identifiable, 2) all CRF pages that have the same set of identifiers are unique, and 3) all CRF pages are consistently represented in the electronic database.

If you are working with multiple data collection systems, many different indications, or any other confounding factors, you only need to meet these three assumptions in four steps:

1. Identify the CRF page and the SDTM domain it maps to.
2. Grab the domain program from wherever it is stored.
3. Grab the module program associated with the CRF page.
4. Paste the module program into the domain program.

That is really all we are doing here. Conceptually, it is a simple copy and paste.

EXPANSION

Since the programs are structured systematically, they can be taken apart. This allows us to keep adding new modules whenever they are encountered. If your custom CRF pages meet the criteria then they will be custom only once.

The addition and storage of new CRF pages carries over to sponsors who have specific SDTM mapping needs. Again, after updating the SDTM programs to fit the sponsor's needs, we can then extract the pieces (module program and domain programs) and store them in a sponsor-specific directory. The next time we work with the sponsor, we can use the modules we created earlier.

This process formalizes the copy-and-paste approach so many programmers are employing. By copying codes from a standard instead of from study to study we can easily copy the most appropriate code for a study and all it takes is our three assumptions. Hardly a herculean task.

A NOTE ON VALIDATION

Undoubtedly, we must validate outputs on every study, but what would be the appropriate level of validation when using the module method? To obtain the maximum benefit from the module method, we want to avoid re-validating something previously validated.

We validated the Module method as a two-step process. First, all the modules and domain programs underwent a code review. Next, we tested every module in the domain program performing a form and content check of the final SDTM data set. We did not validate every possible combination of module within each domain. This would have been too time-consuming and some combinations are never likely to occur.

This only made us certain that the method would work for any combination. Validation on individual studies still needs to occur. Presented here are two possible validation strategies to maximize the benefit of the Module method.

Semi-Independent Validation

Semi-Independent Validation would be similar to independent double programming. The key difference would be that both the production programmer and the validation programmer would be working from the combined module and domain programs. Other than having the same starting point, the programmers would remain independent.

This validation technique allows for the validation side to benefit from the Module method and avoids the redundancy of having to recreate a validation set of modules and domain programs.

Tracking Module Combinations

The tracking method would use full and independent double programming and track the validated module-domain combinations. Using this method, a combination can be validated and logged, and then the validation program can be stored in a central directory for reference and used on other studies.

This method requires the validation programmers to be fully aware that their programs would be used on future studies and plan accordingly with judicious use of study specific programming. This method is still the “gold-standard” of independent double programming, but it adds complexity in that the validation programs would not be systematically structured like those in the module method. This may cause study-specific assumptions to be included that could make them worse starting points than starting from scratch.

CONCLUSION

The SDTM automation process presented in this paper can be implemented by anyone, even if you don’t think you have time to devote to standards. Since the programs follow standard structures you can build up your standard library over time as you encounter new CRF pages. Nobody likes redoing an SDTM data set they’ve done before and for good reason: it’s a waste of resources. There are so many other things we can be doing with our time than mapping the latest exposure date to RFXENDTC for the umpteenth time.

The module method in this paper can be used to free up time and create savings that eventually can be passed on to the patients who hope to one day receive the drugs we study.

REFERENCES

- Ewing, D. 2010. “Automating SDTM File Creation: Metadata Files Speeding the Process.” *Proceedings of SAS Global Forum 2010 Conference*. Malvern, PA: Available at http://support.sas.com/resources/papers/proceedings10/181-2010.pdf
- Li, C., et. al. “SDTM Domain Mapping with Structured Programming Methodology.” *Proceedings of PharmaSUG 2012 Conference.* Ridgefield, CT. Available at http://www.lexjansen.com/pharmasug/2012/DS/PharmaSUG-2012-DS06.pdf
- Aerts, J., “SDTM-ETL”, website. March 18, 2016. Available at http://www.xml4pharma.com/SDTM-ETL/

ACKNOWLEDGMENTS

I would like to thank Jeffery Johnson, Rakesh Mucha and Zahir Karim for reaching out and creating the opportunity to be involved with SDTM automation at SCRI Development Innovations.

CONTACT INFORMATION

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

Name: Taylor Markway
Enterprise: SCRI Development Innovations
Address: 3322 West End Avenue, Suite 900
City, State ZIP: Nashville, TN 37203
Work Phone: (615) 329-7274
E-mail: Taylor.Markway@SCRI-Innovations.com