Paper PA48

Stretching Data Training Methods: A Case Study in Expanding SDTM Skills

Richard Addy, Rho, Inc.

ABSTRACT

With CDISC moving towards becoming an explicit standard for new drug submissions, it is important to expand the number of people who can implement those standards efficiently and proficiently. However, the CDISC models are complex, and increasing expertise with them across a large group of people is a non-trivial task.

This paper describes a case study focusing on increasing the number of people responsible for creating the SDTM portion of the submissions package (data set specifications, annotated CRF, metadata, and define file). In moving these tasks from a small dedicated group who handled all submission-related activities to a larger pool of programmers, we encountered several challenges: ensuring quality and compliance across studies; developing necessary skills (often, non-programmatic skills); and managing a steep learning curve (even for programmers with previous SDTM experience).

We developed several strategies to address these concerns, including developing training focused on familiarizing new folks with where they need to go to look for details , a mentor system to help prevent people from getting stuck, focusing extra attention on domains that consistently caused problems, and creating flexible and robust internal tools to assist in the creation of the submission.

INTRODUCTION

As the use of CDISC standards in regulatory submissions has increased, there has been a growing need for more people who can implement this work. When we began to prepare submissions using the CDSIC study data tabulation model (SDTM), it was sufficient to focus on developing expertise in a small team of people who oversaw all the submissions work. As SDTM became more prevalent (and is on the verge of becoming required), there was a need to expand the number of experts – there was more work to do, and the more people proficient in the submissions process, the higher the overall quality of the package became.

This paper describes how we moved from a small core of people who specialized in SDTM submissions to a more general expectation that any statistical programmer should be conversant enough with submissions standards in order to implement them. Also discussed are some of the hurdles we knew we were going to have to overcome – the submissions process has a steep learning curve, for example – as well as some unexpected pitfalls we encountered along the way.

THE STARTING POINT

When we began working with CDISC standards, the process was new and not widely adopted. The SDTM conversions we did were generally done as legacy projects – after a study had been locked. As a result, our submissions group was primarily composed of statistical programmers and statisticians. The advantage was that the submissions group quickly developed expertise with SDTM standards and it was easy to create efficiencies.

However, as time went on and CDISC standards became more widely adopted, it became clear that this was not a sustainable approach; there simply was too much work to do. It also became clear that given the direction regulatory submissions was taking, proficiency with at SDTM standards was quickly becoming a core skill for statistical programmers – and it was a broad skill.

Therefore, we recognized that we needed to move to a more general approach – one where every statistical programmer would become proficient in the SDTM aspect of a submission. Many already had some exposure – particularly in constructing SDTM data sets, but we needed them to become fully engaged in other aspects, such as creating study metadata, building the define file, and annotating the case report form.

METHODS

One issue that we knew would be a problem was the learning curve – between the details of the tabulation model itself, the metadata associated with the tabulation, the annotation of the case report form, and the define.xml file, there are a lot of moving parts in a submissions package, and it can be overwhelming for someone to try to learn everything at once.

Despite this, we needed to reinforce the expectation that to create a submissions package, a programmer new to the SDTM submission process needed to approach the study as an organic whole, not a series of isolated steps. New programmers needed to understand an entire study, not just focus on one section at a time. They also needed to understand the interdependencies of the individual segments of the submissions package – how changing one element impacted the others.

GENERAL TRAINING

We did not want to throw programmers new to the SDTM process into the deep end – that's a recipe for failure. So, our first step to mitigate this was to provide a general introductory training to all programmers. This was aimed at giving them a high-level view of the standards developed by CDISC they would need to know (CDASH, SDTM, ADaM, ODM), communicate the general process of putting together a submissions package, and the overall expectations of what would be required. The training was delivered to a large number of programmers and other interested parties, and involved an instructor presenting the overview, followed by a question and answer session. This broad approach had the advantage of being quick to implement, and could be given to a large number of people at once. On the downside, it was a high-level overview, and by necessity, without many details. Programmers who attended the introduction, but did not begin submissions work soon thereafter tended to lose details in a short span of time.

ADDITIONAL, FOCUSED TRAINING

We then created a series of smaller training sessions focusing on smaller portions of the submissions process:

- Understanding the SDTM model and submissions process in more detail
- Creating programming definitions
- Completing the metadata for a study
- Annotating the case report form
- Writing the reviewer's guide
- Reviewing and editing the define.xml file
- Checking the submissions package for consistency and compliance

These smaller sessions were intended to provide a greater level of detail about a specific submissions aspect, provide information about best practices, and to serve as a point of reference. The format was similar to the general CDISC training, but presented to smaller groups. Again, we found that if a programmer did not begin work on a submissions package soon after the training, information did not tend to be retained.

That tended to be a recurring theme – the trainings were useful as a method for providing an overview, but it was not until a programmer became actively engaged in a submissions package that it all starting to sink in. As long as the process was abstract, the programmers were able to hold on to the process as a whole, but it was not until they actually began to wrestle with an implementation that all the pieces began to fall into place.

TRANSITIONING PROGRAMMERS INTO SUBMISSIONS TASKS

We used three techniques to get programmers through this initial rough patch:

- Materials for previous trainings were available as a reference
- Checklists were developed to guide a new programmer through the process Completing the metadata for a study
- An experienced submissions team member was assigned to work with a programmer on their first submissions package

Having the previous training materials available was useful as a refresher; It allowed the new programmers to quickly remind themselves of the general process before diving into detailed project work. The SDTM implementation guide is a technical document, not an instructional one, and reviewing the overviews made it easier for the new programmers to approach it as such.

The checklists were helpful because there are so many parts of a submissions package, and they are usually interrelated. It's easy for a new programmer (and often, a not-so-new programmer) to miss a step, and the checklists served to keep them on task and avoid having to go back and pick up a step they missed. They also helped a new programmer manage their time; they knew what they done, and they knew what was still left to do.

Most importantly, new programmers were paired with an experienced team member, who was placed in a mentoring role – the new programmer would actually create the submissions package, but there was someone they could talk when they had questions, and there was someone who could review their work. By having the mentor as part of the project team, the new programmer had a dedicated resource when issues arose – they did not have to try to find help from people who were busy on their own projects. Of all our methods, this was the single most effective technique – this one-on-one interaction made a huge difference in how quickly a new programmer gained expertise in constructing a submissions package.

EXPANDING THE SCOPE

Many of the programmers came to this process with experience programming SDTM data sets, and one habit that often needed to be broken was looking at elements of the submissions package in isolation. Programmers that were used to picking up specifications for a single domain and implementing them had to realize that approach did not serve them well for the overall submissions process.

To address this, we had new programmers begin with domains that reach across SDTM data sets – the trial design domains. This forced them to deal with some of the thornier SDTM concepts - EPOCH and ELEMENT, especially, but working with VISIT and VISITNUM was also helpful. This encouraged them to begin with a broad view of the project. In addition, the checklists we developed guided new programmers to work on multiple elements of the submissions package at once – while creating specifications and metadata for a domain, for example, they were also instructed to work on annotating the case report form at the same time

We also found that the more a programmer was exposed to SDTM conversions of different types of studies, the better. Also, since there are many ways to construct a tabulation that are compliant (but some approaches may be more elegant than others), it was helpful to provide an area where programmers – both those new to the process and those with more experience – could regularly meet to discuss problems and techniques. This forum was self-directed and open to discussions of and SDTM-related topic.

LESSONS LEARNED

A programmer with previous experience programming SDTM data sets often had trouble when it came time to develop a complete submissions package. We found there were two causes for this: creating a submissions package is a substantially different process than just implementing data set specifications, and many of the tasks required for the submissions package are non-programmatic in nature.

In fact, we wanted programmers to move past the point of just implementing specifications. A programmer familiar with the foundations of SDTM and the particulars of the study being tabulated is much more likely to catch errors, omissions, and inaccuracies in specifications than one who is there just to create a program the way they're told. This was one of the reasons we wanted to spread expertise with the submissions package – we found that we ended up with a much higher quality product when people were not so focused on a small aspect of the work and had a more in-depth working knowledge of how everything works together.

However, the second aspect – a submissions package has a lot of non-programmatic tasks, and programmers like to program. Filling in origins, FDA definitions, CRF locations, controlled terminology lists, and assorted tasks often seemed tedious to programmers new to these tasks. Part of this is just the nature of the job, but we did find benefit in empathy. Reminding programmers of studies where they had been forced to deal with disorganized data structures served to help them recognize the benefit of their efforts, at least somewhat.

Past that, we also found that the type of programming someone had done previously played a part in how well they adjusted to new submissions tasks – SDTM requires a lot of data set programing, and people who were used to working that way generally had an easier time than people who were more focused on table programming.

We noticed a few issues came up consistently with programmers new to the submissions process:

- EPOCH and ELEMENT in the trial design domains
- Consistency in variables across domains (EPOCH, VISIT, ELEMENT, etc.)
- Some domains were harder to understand than others – DS, LB, SE, RELREC
- Controlled terminology – when it must be used, when it should be used, and when lists could be extended
- Avoiding truncation (especially in supplemental data sets)
- How to address issues found by the OpenCDISC validation program
- Differentiating between the various reference variables in DM (especially reference end dates)

Initially, we addressed these issues as they came up, but once we noticed we were seeing these things often with new programmers, we updated our training materials to mention common techniques to handle them. In some cases, such as handling the output from the OpenCDISC validation program, we added additional training. We also pointed programmers to further resources on the web – these problems are not unique, and several very good papers have been presented on how to handle them.

We also found that in many cases, our tools and processes needed to be improved. Sometimes it was a matter of updating documentation – when there was just a small group working on submissions, everyone knew how things worked. When more people were brought in, we found that the documents we were pointing them to were in some cases obsolete, and had not been updated when the tools were updated.

In other cases, having new people come into the process opened our eyes to some efficiencies – we had been done things one way long enough that it had become tradition. Having new programmers question the some of the repetitive nature of some tasks lead to streamlining how CRF locations were entered, for example, and guided improvements in the tools we use to manage metadata.

In some cases, having more people work on submissions provided the impetus to finally go ahead and resolve some issues in our metadata tools – it was worth fixing things rather than consistently work around them. For example, in cases where two studies were very similar, it was easier to copy the metadata from one study rather than starting with the gold standard SDTM implantation guide. Originally, that process required human intervention – it was automated so the person handling the study's metadata could do it on their own.

NEXT STEPS

We approached expanding the number of people working on submissions activities by beginning with programmers – after all, they're the ones who will be creating the submissions data sets. Also, many of our programmers also had ADaM experience, which helped provide context. However, many of the submissions tasks do not require a SAS® programmer to complete, and a key element of successfully tabulating the data for a study is understanding how the data was collected. Therefore, the next step will be to bring some study builders into the process. Their insight, based on their knowledge the structure of the incoming data and the CDASH model, should prove very helpful.

CONCLUSION

Overall, we found that the best way for programmers new to the SDTM model to learn about constructing a submissions package was through one-on-one interactions with a more experienced team member, and small group discussions so that programmers can learn from each other. This is resource intensive, but effective. Providing checklists keep the new programmers on task, and updating our tool and processes made their jobs easier. In the future, we will include study builders – people who construct the clinical databases for studies – in the SDTM submissions process as well.

CONTACT INFORMATION

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

Richard Addy
Rho, Inc.
6330 Quadrangle Drive
Chapel Hill, NC 27517
(w) 919-595-6579
(f) 919-408-0999
richard_addy@rhoworld.com
http://www.rhoworld.com/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.