PhUSE 2016

Paper PD09

On the Fast Lane to a Successful Submission – A Lead Programmer's Guide to an Efficient and High-Quality Submission Programming Deliverable

Michael Reich, PRA Health Sciences, Mannheim, Germany
Hannes Engberg Raeder, PRA Health Sciences, Mannheim, Germany
Jim Witt, PRA Health Sciences, Mannheim, Germany
Doris Kolb, PRA Health Sciences, Mannheim, Germany

ABSTRACT

Told from the point of view of the project's lead programmer, this paper illustrates certain key features of a submission project consisting of the integration and analysis of multiple studies, and involving a large programming team. We discuss technical aspects of the meta-documentation such as reviewer's guides and define.xml, the conversion of legacy data into SDTM, the integration of the individual study data into an integrated database, and the creation of ADaM data and Tables, Figures and Listing (TFLs) for the Integrated Summaries of Safety and Efficacy. In addition, we look at the interpersonal aspects and challenges of leading such a large team, at the challenges that the different experience levels of team members can present, and at communication issues, too. We go into each of these items in detail, discussing best practices and not so obvious pitfalls, and how the latter can be avoided.

INTRODUCTION

To a first-time lead programmer at the beginning of an integration and submission project, the sheer number of tasks and hours and the high stakes involved can be quite daunting. These projects will almost always entail the creation of integrated databases (IDBs) for the purpose of writing both integrated summaries of safety and efficacy (ISS and ISE). Where legacy databases in non-CDISC formats are involved, it can also require the preparation of data and metadata for multiple studies that feed into the IDB and/or the conversion of these studies' tabulation data to SDTM.

Integration of data into an SDTM IDB means striking a balance between harmonizing study data prepared in different formats and ensuring that crucial study-specific information is preserved along the way. The complexity of this task increases when the individual studies have been conducted by different vendors or sponsors and possibly many years apart. The harmonized SDTM IDB can then be used to program ADaM-formatted analysis datasets underlying the integrated analyses of safety and efficacy. Programming integrated ADaM datasets can be more complex than for individual studies and often requires special considerations and solutions to account for the diversity of the pooled studies.

Told from the point of view of the lead programmer, this paper illustrates the journey from start to end of such a project, which, depending on scope and timelines, can easily involve 20 programmers or more at a time. We focus primarily on the aspects of a submission to the FDA, but many aspects apply to submission work for other regulatory authorities. We discuss best practices and not so obvious dangers and pitfalls and also look at interpersonal aspects such as communication and the challenges of leading such large, sometimes heterogeneous teams. While this paper is written from a CRO perspective, much of this can be applied within a sponsor setting as well.

After describing on how the lead programmer should prepare for the task and verify the actual scope of the project, we start off where many projects end, namely with the meta-documentation (define.xml v2.0 and the reviewer's guides). We will show how some automation can make the majority of this work easier to manage and illustrate how the principle of "Metadata First" can be applied to all parts of such a project. We highlight the best practices for the SDTM IDB, and the ISS and ISE ADaM, respectively, and briefly cover less self-evident hazards that can be encountered.

Finally, standards and good processes cannot fully compensate for interpersonal obstacles. On the contrary: projects of this magnitude require strong leadership, especially as the teams grow in size and are made up of members with heterogeneous skill sets and experience. We highlight preferred communication approaches and how to ensure that the team pulls in the same direction, as well as how and when to delegate responsibilities to other team members.

PREPARATION IS KEY – KNOWING THE ACTUAL SCOPE

Instead of rushing to get the programming environment and tools set-up so that the programming team can start working on the project as soon as possible, it is necessary to first gain an understanding of its scope in detail, its timelines, and the assumptions underlying both. The scope at the outset may at first appear well-defined, but there are not so obvious details lurking about that can change the required effort drastically and will turn out to be nasty surprises if one is not well prepared.

Thus, to avoid any unwelcome surprises while the project is fully underway and to be able to build a solid foundation for an efficient approach, the lead programmer should first familiarize herself/himself with the underlying medication and how it is analyzed, as well as the available documentation and data of the included studies.

It is also important to establish a good working relationship and open channels of communication with counterparts in other functions, e.g. biostatistics, regulatory affairs, medical experts, and safety and risk management etc. These team members are a valuable source of support and also of information and insight that will ultimately determine which topics will likely be relevant for the integrated analyses and thus which studies and data will need to be included in the integration effort.

UNDERSTANDING THE STUDY DRUG AND ANALYSIS

Understanding the study drug, the underlying indication, and past study-level results greatly facilitate the identification of information that should be included in the integrated database and allow for reliable early predictions of what may need to be included in the analysis. Besides standard study and safety topics typically included in an integrated database (like demography, exposure, adverse events, or vital signs), understanding the specifics about the indication will give you an idea what parameters you can expect to be included in the efficacy analysis. For example, a study drug treating pain will require pain scores, while a medication used for sedation will require scoring methods indicating how well a subject is sedated.

In order to gain a deeper understanding of the study drug, and to better anticipate potential topics of interest (which will of course be ultimately defined in the statistical analysis plans for the integrated analyses), a review of both the Investigator's Brochure (IB) as well as of individual study results are beneficial. If for example in an individual study a safety concern was raised regarding a particular parameter, an integrated safety analysis of this parameter across the larger population available in the integrated database allows for further exploration of this concern, helping to understand whether it is a genuine issue or, for example, whether a particular group of subjects is at higher risk.

This not only helps the lead programmer and the team to decide what data needs to be included in the integrated database and what analyses are to be expected, but also ensures that the lead programmer understands which topics should receive extra attention and focus during lead reviews of produced outputs or assessments of incoming data from ongoing studies. As it cannot be expected that each individual programmer knows all these details while still maintaining a high efficiency, ensuring that the lead programmer is aware of these items will lead to a central reference point in the programming team that functions as the final quality gate.

ASSESSING AVAILABLE DOCUMENTATION

Even when legacy-formatted data is being included in a submission, the FDA requires that every database included in the submission is also accompanied by documentation that is adequate to help the reviewer understand the data submitted. Having access to this documentation also of course helps the lead and programming team to understand the study drug, data, and analyses. A useful starting point and source of information when beginning to assess both the available documentation as well as the data would be the Study Data Standardization Plan (SDSP) [1]. This is a relatively new document that the FDA expects to be included the submission process. The SDSP provides a great deal of information on study level data and a brief overview of the CDISC standards applied to those studies. If the client has not yet prepared such a plan, it is important to assess all available documentation and data to provide input to the sponsor who is ultimately responsible for creating this plan to be used in discussions with the FDA on the pending submission.

The following study-level documentation should be available to the programming team at a minimum for each individual study:

- Tables, Figures, and Listings (TFLs) for completed studies
- Clinical Study Report (CSR) for completed studies if available
- Protocol (including all amendments) for ongoing studies
- Define documentation of study-level data (CRF/SDTM and analysis/ADaM)

A good study-level define documentation should include a define.pdf (for legacy data) or define.xml (for CDISC data) as well as a CRF annotated to fit the available data (legacy vs. SDTM) and a reviewer's guide detailing specifics and data issues of the corresponding study.

Lack of adequate define documentation at the study-level is one of those items that can lead to unanticipated and drastic increases in scope. If the data to be submitted has little or no documentation available at the study level, there is a high likelihood this will need to be created to support the submission. Having a good define documentation available also greatly benefits submission programming activities. It allows programmers to more easily identify what study-level information is stored in which source datasets and format, and if it follows a specific code list. Should the scope thus increase to also involve the creation of study-level define documentation it is recommended to perform this task first before any integration programming activities on affected studies begin to ensure maximum efficiency.

Having the study-level TFLs available while programming activities start is advantageous, as they allow for quick insight into study-level analysis and data issue handling, facilitate understanding on requirements for the integrated analysis, and ultimately allow for cross-checks between the integrated and study-level analysis. These cross-checks will ensure higher quality as they enable the lead to verify that individual study populations in the integrated database match the reported study-level demography and adverse event data, providing that extra bit of security that no data was lost or had its meaning/interpretation changed in the process of creating the integrated database.

For studies ongoing or planned at the time of project start the lead programmer should collect as much information as available so that these studies can be considered in the project planning. Usually the pivotal studies are ongoing or close to start at project start, but some Phase I studies could be still in an early planning phase (e.g. drug-drug interaction studies, etc.). It is a good idea to set-up a document to keep track of what study-level information is currently missing and when the team expect it to become available for all studies to be included, independent of the studies' completion status. Finally, to ensure that the outstanding data and documentation is received in the expected quality, it is critical that the lead programmer actively communicates with the teams responsible of the creation of said data or documentation.

ASSESSING AVAILABLE DATA

In parallel, the lead programmer should assess prior to kicking-off programming activities is the data actually available for the individual studies. The lead programmer needs to understand what can actually be expected from the individual studies to effectively guide the programming team in creating the integrated database best fitting the submission. A proven method in doing so is mapping out what data is available in which datasets and already defining the target integrated SDTM domains with some high-level instructions as applicable. If done right, this considerably speeds up the creation of the integrated database as programmers will no longer need to look for data applicable to the corresponding domain they are programming and can more focus on fitting the designated source data to the selected target domains.

For each study level submission package all data required to recreate study-level analyses need to be available and included, and also adhere to the FDA's Study Data Technical Conformance Guide (SDTCG). Submitted datasets, for example, must fulfill SAS® XPT v5 requirements [2]. Furthermore, while it is permissible to submit data that are not in English, translations of these data into English are required. Lack of SAS XPT v5 compliance (or loss of data due to misunderstanding of its format's limitations) as well as inclusion of non-English data are potential issues that can be encountered when assessing in particular legacy data for both submission- and integration-readiness. Both can result in updates to the submitted data itself, as well as to the study level documentation to ensure all metadata changes and the addition of variables holding translations are covered and well documented, ideally by including sufficient traceability to understand what changes were implemented to meet the submission requirements.

In case only legacy raw/analysis data is available for a specific study, another of those pesky non-obvious scope-increasing items rears its head. In general, the FDA will prefer to receive data in CDISC format. Past experience shows that it can be sufficient to provide SDTM alongside the original legacy raw/analysis data with corresponding documentation (for example, the define.pdf discussed in the previous section) showing full traceability between legacy data, SDTM, and study-level CSR TFLs; see the discussion on traceability issues with legacy data conversion in the SDTCG [3].

Should the requirement arise to create SDTM for legacy study data, the lead programmer should keep in mind that this can also be done while increasing efficiency down the road. By ensuring that the created SDTM for all affected legacy studies is as similar and harmonized as possible, the effort required to include these studies in the integrated database can be reduced drastically. When performing a legacy SDTM data conversion the main priority should be on ensuring that the created SDTM data can reproduce the study level CSR analyses first, then SDTM compliance, and lastly as much harmonization to fit the target integrated database structure as possible.

FORMULATING THE BATTLE PLAN

Following the items described in the previous sections the lead programmer should now have a good feel for the scope and requirements for this project. Using all the knowledge gained by reviewing the individual study documentation and data, the lead programmer should now lay out a plan detailing the order and timeframe in which individual sub-tasks are to be completed to allow both for being as efficient as possible while still keeping the intended submission timelines.

The most logical and efficient order of tackling all items contracted and newly identified during the data and documentation review would be the following, where each item uses the output of the previous steps:

1. Update source data to comply with FDA's SDTCG (translations and XPT v5 compliance)
2. Create define documentation for source data
3. Perform Legacy SDTM conversion
4. Create the integrated SDTM database from source SDTM (original or legacy converted)
5. Create the ISS and ISE ADaM databases and TFLs

Although ideal, the lead programmer might not be able to strictly follow this order step by step while still remaining able to guarantee the intended timelines. Careful consideration must be given here to decide which of these items can be done in parallel to retain as much potential efficiency as possible. For example, while Steps 1 through 2 are underway for legacy study data, Step 4 can already be performed by a separate team for studies that are readily available in an SDTM and ADaM format. As soon as the "legacy team" is ready to start Step 3, efficiencies can be gained by aligning the legacy SDTM conversion target structure to fit the integrated SDTM database structure now already created for non-legacy studies as close as possible. This will in turn also ensure that the legacy SDTM data can be included into the final integrated database more easily as well. To parallelize these steps even further, ISS/ISE ADaM creation could also already be started, basing it on the integrated database that includes non-legacy studies only.

ON A FAST PATH TO A HIGH-QUALITY DELIVERABLE

One of the biggest challenges of integration deliverables is the sheer volume of outputs created, from SDTM to ADaM to TFLs, where even the smallest change can require a lot of time to implement and it is very easy to mistakenly exclude a particular change in a subset of the affected outputs. On the other hand, this volume of outputs allows for a great amount of standardization that would not be worth the investment of development time in a single-study setting.

Thus, the main goal for the lead programmer should be to ensure that the environment and tools supplied to the programming team automate as much as possible to enable the team to make minor updates in a fast and consistent manner. One way this can be achieved is by setting everything up to be metadata driven and by being aware of various items that can be automated or simplified for each of the individual upcoming tasks by using a robust package of SAS macros.

METADATA AND AUTOMATION

A metadata-driven approach to creating outputs is essential for the efficient and consistent creation of an integrated database and analysis. A significant portion of the work required to produce the various outputs can be automated and largely controlled by the metadata, which needs to be created and maintained in order to generate the documentation required for the submission anyway, or other central supporting documents.

The aim of this approach is to reduce the amount of items and variables the team needs to cover in their programming while at the same time ensuring that any updates to these items can be implemented in one central location automatically perpetuating to all affected outputs across all studies. Not correctly identifying items or variables that can be covered by this will have tremendous impacts when updates are required, especially if these are updates that need to be implemented to SDTM. There are only few things more frustrating than needing to update and re-verify the outputs of one program per domain and study because something trivial but still universal needs to be changed like the sort key of a dataset.

This automation starts out with simple items, e.g. ensuring that the integrated SDTM and ISS/ISE ADaM datasets match the variable structure given in the corresponding define.xml files by automating the application of labels, formats and ordering of variables; but must also be extended to cover more complex items for which we will provide some examples in the following sections. To facilitate the implementation it is advised to develop standard domain/output templates using all the standardized tools and macros created beforehand. These will ensure that the amount of time the programming team needs to spend on producing and maintaining outputs is further reduced, and they are a good entry point for programmers unfamiliar with the assigned work.

Further, building all automation on the foundation of 'define-ready' specifications ensures that the presented data and the documentation are in alignment and drastically reduces the amount of time needed to create the define.xml file and accompanying documentation. Experience has shown that doing define.xml and reviewer's guides after the database is final constitutes a significant risk, since often it's only during their creation, when the programmer adopts the more macro-view of a potential reviewer, that issues in the data itself become apparent.. Doing metadata first thus adds another layer of quality control, catching and eliminating another source of potential issues early on in the process.

EFFICIENTLY CREATE THE INTEGRATED SDTM DATABASE

Of all the programming tasks, the integrated SDTM database may offer the most potential for automation. Besides the two items mentioned in the previous section (variable labels and ordering, and –SEQ creation), another straightforward item that can be automated is obtaining decodes for selected variables based on define.xml code lists, under the assumption one takes the 'metadata-first' approach described above. Although SDTM by design does not contain many variables with a code-decode relationship, variables where this can be applied would primarily be –TEST, VISIT, and –TPT. Creating these decode-variables automatically ensures that the programming itself would only need to be adapted if the corresponding counterparts (--TESTCD, VISITNUM, or --TPTNUM) are to change, while any change in the decode can be controlled and automatically implemented from the metadata level.

Other SDTM variables that generally can be automatically populated are –BLFL, EPOCH, general visit mapping, --DTC and –DY variables. Their automation relies less on "metadata"-based tools than on standard tools and macros. Besides these there are other, more domain specific items that can be automated, such as LB mapping (source to SDTM including unit conversion) or the creation of the complete SV domain.

Assuming the above has been implemented, the part actually requiring the most time will be the writing of the specifications/metadata themselves. To support this step and ensure consistency, it is advised to design the specification documents to allow for easier cross-referencing between studies. Assuming a workbook-based specifications document, this can already be facilitated by simple things, such as ensuring that there is one tab per target domain containing variable-level specifications for all studies, either side-by-side or top-to-bottom. Setting up the specification either way simplifies the writing and programming implementation of specifications that are harmonized across studies, as they allow the specifier and programmer to more easily look-up how specific variables were created for the same domain in other studies.

In addition to the tools and methods already discussed above, giving some thought to how granular the individual programs should be for integration programming can also result in increased quality and efficiency. For example, if a larger part of the incoming study data are very similar (structure- or even content-wise), it should be considered to integrate each individual domain for all of these studies in one program per domain, instead of each study and domain combination individually. If on the other hand the incoming data is too diverse, quality and efficiency can be maintained more easily if each study and domain combination is encapsulated in individual programs.

DEVELOPING THE ISS AND ISE ADAM DATABASES

Because ADaM is more flexible than SDTM, it's more difficult to automate variable creation in the former. Still, there are a number of opportunities for automation. The code-decode automation mentioned in the previous section can be utilized a lot more in ADaM, and there are some possibilities to automate the creation of variables, such as --DT, --TM, and --DTM, as well as variables reflecting periods, phases, and planned/actual treatments. Finally, automating the process of identifying and merging on ADSL "header" variables to non-ADSL datasets by using the information stored in the metadata will also be a huge time-saver. This becomes especially clear when the ISS/ISE analysis matures and more subject-level variables are needed to allow for new subgroup analysis requirements identified after the first integrated analysis results were created.

When developing the ADaM databases programmers should naturally always keep the target TFLs in mind in order to leverage uniformly across all analysis datasets the automation opportunities that ADaM offers. For example, by having numeric variables that represent categorical results (like age, race, etc.) in the order they are supposed to be presented in the tables. Combining this with the code-decode automation mentioned above, any changes in the display value can be implemented centrally in one place while it would automatically be applied to all affected outputs.

Similar to a single-study ADaM database, an integrated ISS/ISE ADaM database usually uses a single data source in form of the integrated SDTM database. This also means that any other good practices to improve efficiency and quality already established for single-study ADaM programming can mostly be utilized here as well. Nevertheless, there are still some key differences and challenges that arise from combining multiple studies into a single ADaM database.

For one, before any ADaM programming starts, it should be carefully defined how exactly the individual ISS/ISE tables will select the population for the individual treatment columns, as this can widely differ when considering the design of the included studies. Tables based on Phase 2 or 3 studies alone might work just as well when using the standard planned and actual treatment variables TRTP and TRTA, but as soon as Phase 1 cross-overs or core and extension studies are involved this might no longer work reliably, especially if only events under any dose of the main study drug are to be analyzed for these studies.

Lastly, depending on the focus and complexity of the efficacy analysis and the design and data availability of the pivotal studies, it might be worth considering creating the ISE ADaM database primarily out of the individual pivotal study-level ADaM databases and only utilize the integrated SDTM database for high-level information. This approach would especially make sense if the ISE and pivotal studies are in the same hand, as it especially depends on how standardized the studies to be included are. More often than not it is worth evaluating if this is a viable approach, as it can save quite some time if it can be done.

MASS-PRODUCING HIGH QUALITY ISS AND ISE TFLS

The last programming task in an ISS and ISE analysis are the TFLs, of which in total easily thousands can be required. There are some ways to automate and support multiple smaller items, for example smaller macros for the creation of summary statistics of given parameters in a standardized format or to automatically obtain population Ns, along with standard templates facilitating the creation of the various types of TFLs that can be encountered (i.e. event incidences, shifts, subgroups, etc.). Additionally, the number of repeat tables in submission projects are usually staggeringly high, but compared to a single study the displayed treatment columns could differ greatly between those repeats. Careful consideration can turn this problem into an efficiency, if macros are set-up to ensure the treatment columns belonging to the current analysis group are displayed correctly for each individual repeat table.

Otherwise, there are a lot of little things that can incur significant maintenance hours if not handled correctly. For example, a generally good practice to aid review is to include the table numbering in individual output file names. Unfortunately, this at the same time introduces the problem that any renumbering of the outputs would also require updates to the TFL programs to ensure that outputs are still named correctly. This can be alleviated by defining unique names for each output and using a central document to link those unique names to the final output names (and even to the to be displayed titles and footnotes). With the right macros, these final output names and titles and footnotes can then be dynamically applied during runtime of the TFL programs, while they can also be easily maintained centrally.

Similarly, it can be quite frustrating if the order of columns or just the column headings needs to change and improper preparation would require the team to go through each individual program to implement this. Again, with the right set of centrally maintained macros and documents these kinds of minor but time-intensive updates can be simplified considerably.

Extra consideration should also be given to weighting how many outputs should be created by a single program. At first it seems to be efficient to create those 200+ AE sub-group tables with the same program, while in reality a program with these many outputs is not only harder to maintain but every little change – even if intended to only affect a fraction of the outputs – will require a re-verification of all outputs created by the program, just to be sure that everything is still created correctly. Consider instead breaking these up into smaller chunks, for example by having dedicated programs for TEAEs and serious TEAEs, or separating demography-based subgroups from subgroups based on concomitant diseases or medications. The loss in efficiency of maintaining multiple programs is already less than the extra-work required to re-verify all outputs every single time, but can be further minimized by first verifying that the TEAE-specific program works perfectly and then subsequently recycling its code for the serious TEAE-specific program.

HERDING CATS - BUILDING A SUCCESSFUL AND EFFICIENT PROGRAMMING TEAM

Building and managing programming teams for these kinds of projects is a challenge in itself. While having good preparation, standards, processes, and tools greatly facilitate and support the efficient creation of high quality outputs, all stands and falls with how well the programming team performs.

A good programming team should be balanced and consistent throughout the submission project. The lead programmer must know how to communicate with the team and what tasks should be delegated to ensure that the lead herself/himself does not become the bottleneck of the project. A good indicator of a well–managed team would be the lead being able to just be out and enjoy his vacation, without the need to worry about the project health or working overtime the weeks before and after the vacation to compensate for the time out of office.

BALANCED AND CONSISTENT TEAMS

In a perfect world the programming team members working on the submission project are experts in assessing data, understanding protocols and other study documentation, building SDTM domains that exactly reflect the collected data and the intentions of the protocol authors and the CRF designers, always seeing the simple but not so obvious solution while specifying and programming the ADaM datasets. An additional luxury would be to have the same programmers create the ADaM datasets who also mapped and harmonized the diversely formatted legacy databases, as they are intimately familiar with these studies. This will rarely or ever be the case. Instead, the lead programmer will need to integrate programmers of different skill levels, some of whom may even only be available for a limited amount of time before being needed on competing projects.

Losing experienced team members to other project work or for other reasons is a risk, and steps should be taken to minimize both the risk and the impact it can have. While planning resources globally to fit all concurrent project needs seems to offer some safety and consistency, the situation can change quickly and drastically. This especially applies in a CRO setting, where an unexpected scope change in one project can require reassignments across the whole project landscape – for better or worse. It can help to reduce the uncertainties these situations entail by defining early on in the project a set of key programmers as core team that can oversee certain tasks, which are then planned to stay on the project from start to finish, and obtaining buy-in from programming and project management on that core team.. Nevertheless, the lead should prepare for situations were even key personnel might become unavailable and have some succession planning in place, while at the same time ensuring that resourcing concerns are communicated early and clearly to the lead's line management and project management.

Of course every lead wants to have the most experienced and savvy programmers in the organization working on his or her submission project, but this is usually unrealistic. Instead, there will be a mix of junior and more senior team members available, or those who have integration experience and those who do not. Important is to strike a healthy balance so that the more experienced programmers or the lead do not become bottlenecks. It will severely impact the performance of the team when the proportion of the inexperienced programmers is too large, as the few remaining senior programmers will then need to spend most of their time supporting the inexperienced ones. The experienced programmers will then primarily be caught up with verifying and correcting the work and will subsequently also be less focused on driving the project forward on the more complicated topics.

Considering the complexity of a submission project, it is advised to have two experienced programmers for each inexperienced one. It is important to keep in mind though, that even an inexperienced programmer should at least already have worked with CDISC SDTM and/or ADaM standards in the past to ensure that the actual amount of support required by the inexperienced programmers can be reduced to advanced CDISC or integration and submission topics.

DEVELOPING PROGRAMMER'S SKILLS WHILE MAINTAINING EFFICIENCY

Few projects offer as many interesting tasks and opportunities for professional development for the individual programmer as a submission project. In the CRO setting, which typically has a larger proportion of less experienced programmers, this offers favorable circumstances to develop programmers. Legacy study conversion and the programming of the integrated database present an excellent opportunity to grow SDTM skills and understanding of CRF annotation. ISS and ISE programming offers up interesting ADaM challenges, with BDS and OCCDS datasets consisting of a combination of cross-overs, phase I to III studies and other peculiarities seldom seen in single-study databases. Few if any courses or tutorials can offer the hands-on experience a submission project can.

Assigning team members who may not have that much lead experience to manage discrete components of integration projects can help these programmers develop skills and gather experience as a lead without all the responsibility entailed with it, as long as an "overall" lead continues to manage all the programming pieces.

Still, integration and submission projects should not be treated as if novices can learn here everything they need to know. Programmers new to CDISC or even to the industry will be of virtually no value for the project as whatever they produce will have taken too much time from the new programmer as well as the experienced staff. It is thus recommended that programmers assigned to these kinds of projects for learning opportunities at least bring some CDISC knowledge with them, and they should at least have supported one or two single study projects to ensure that they are familiar with clinical trials in general.

Keeping all of the above in mind, the department should thus select a set of programmers who are deemed ready to take on this challenge and who can profit from the experience, while still maintaining a good balance of overall experience in the pool of assigned programmers. Some care should be taken to ensure that programmers are assigned to tasks that fit their skill level and experience. This will help both to maintain overall project efficiency while at the same time enabling programmers to hone their skills.

COMMUNICATING WITH THE TEAM

The more the individual team members understand about the indication, the product, the scope and the timelines, the more committed they will be to the success of the tasks ahead. Indeed, when team members are not given access to crucial project information and are informed only on a “need-to-know” basis or not at all, it is likely that it will have a negative impact on efficiency and quality.

To this end, the lead programmer should set up a compact project information document that summarizes and reflects his or her own familiarization with the project scope, indication, and timelines, as well as including links to the protocols, any single-study statistical analysis plans that are available,, annotated CRFs, specifications of SDTM and ADaM domains, and any other vital information related to the project. The project information document should also contain all the project milestones, the project scope and the contact information for any team members functioning as sub-leads for certain parts of the project, but also for other functional team leads, such as biostatistician or medical experts. This document should serve as an introduction and reference point throughout the life of the project life for both new and existing team members and should be updated as needed during the course of the project.

A project of this magnitude will consist of several milestones and sub-milestones; the intensity of the work will vary dependent on the phase of the project. The lead programmer, the lead programmer’s line manager, the biostatistician and the project manager will be aware of changes and sub-goals, but the team needs a reliable channel of information and to be kept up to date on changes and upcoming timelines. The lead programmer should set up regular team meetings to brief the team, highlight past achievements and introduce upcoming tasks. The frequency of these meetings will depend on the ongoing activities.

Finally, as is the case on any collaborative effort, as a lead it is important to establish an open communication culture so that programmers are not afraid to ask questions, and are encouraged to bring up their ideas and to participate in discussions during and outside of the regular team meetings. Programmers who are made to feel they are part of the team will be more committed to the team’s goals, which in turn is invaluable for ensuring high quality in the vast amount of outputs the team will generate during the course of the project.

DELEGATE, DELEGATE, DELEGATE

Even for relatively manageable single studies with a much smaller team it is advisable to delegate tasks to less experienced team members as practicable; for a submission project it is absolutely necessary. The main reason to delegate tasks it to free up the lead programmer and ensure that she/he is able to support the team to the fullest. It is not seldom that the lead programmer first must learn to start trusting others, and rather take on the role of a reviewer than a producer of outputs.

There are a lot of items that can be delegated, and it is up to the lead programmer to gauge who of the assigned staff is best equipped to take on tasks the lead would have done otherwise. There is, for example, no need for the lead programmer to also be the one developing a larger portion of SAS macros used to automate items; nor must the lead programmer necessarily implement the source-to-target data mapping programs needed for the integration or legacy conversion. Of course, the scope of the project, the timelines, and the team size will also determine to what extent the lead is able to delegate some of these tasks. As suggested above, on particularly large projects “sub-leads” can be assigned to component pieces. Candidates for such components are legacy study conversion, the SDTM integrated database programming, and the ISS as well as the ISE analysis, not to mention legacy statistical documentation, if needed.

The primary role of the lead programmer in this case is to support sub-leads, to help coordinate the sub-teams, to manage cooperation with other functional team leads, and ultimately to take on the overall responsibility for the health of the project and the quality of its products.

SMALL THINGS THAT MAKE THE LEAD’S LIFE EASIER

The following list is by no means complete, but presents additional items that can help simplify things or support the lead programmer a bit more.

From a team management perspective, especially with multiple and sometimes overlapping sub-teams, having a central vacation tracker will help in planning resource availability. If the tracker also indicates key milestones or crunch times, this can help both the lead and the team members better manage and plan their time off and to avoid conflicts with milestones or other key team members’ vacations.

Internal team – and in the CRO-setting, client team -- issue trackers have also proven to be invaluable. The internal issue tracker is used by team members to enter questions or issues they have identified, while the lead and corresponding sub-leads will either provide responses or take other measures as needed. This tool can also be used to communicate and document questions and the corresponding feedback to the lead statistician or to other functional

areas. Issues that result in e.g. additions to the project documentation such as the define.xml file or reviewer's guides can also be flagged in these trackers. The external issue log is then used to track all questions that require escalation to the client, as well as their ultimate resolution by the client. Because it is cumulative, it is an instrumental project management tool to document the resolution of all issues raised during the project, but also making sure these get addressed during the project and do not fall through the cracks.

CONCLUSION

Large-scale submission projects can be daunting and are always challenging, but they are indeed "manageable" if one is well-prepared. This means, among other things, ensuring that the proper tools are in place and that requisite information is available before the broader team is involved, and also understanding how to best utilize and leverage the team's abilities. Naturally, one submission will never be the same as the next, which is why one always needs to be vigilant and flexible. Still, many tools and processes can be standardized for such projects to make life easier for both the lead and the support team.

Despite all the hard work required, the uncertainties and dangers that lurk beneath the surface, and the sometimes mind-bogglingly large scope where tiny mistakes can have tremendous impacts, submission deliverables are one of the most exciting and rewarding projects a SAS programmer can lead. They require all the knowledge a programmer gained during his tenure and will still present a challenge. Because of the feeling of accomplishment that comes from having worked on a successful submission, it is not unusual for a SAS programmer who has successfully led such a project to never want to go back to single-study projects. We hope that this paper will help programmers overcome that overwhelming feeling they might have at the beginning of a submission project and encourage them to take on the responsibility when given the chance.

REFERENCES

.

[1] FDA Study Data Technical Conformance Guide, July 2016, Section 2.1, Study Data Standardization Plan
[2] FDA Study Data Technical Conformance Guide, July 2016, Section 3.3, SAS Transport Format
[3] FDA Study Data Technical Conformance Guide, July 2016, Section 8.3.2.1, Traceability Issues with Legacy Data Conversion

RECOMMENDED READING

1. CDISC Guidance on SDTM: http://www.cdisc.org/standards/foundational/sdtm
2. CDISC Guidance on ADaM: https://www.cdisc.org/standards/foundational/adam
3. FDA Guidance on Data Standards: http://www.fda.gov/forindustry/datastandards/studydatastandards/default.htm
    a. Study Data Technical Conformance Guide
    b. Data Standards Catalog
4. FDA ISE Guidance: http://www.fda.gov/downloads/drugs/guidancecomplianceregulatoryinformation/guidances/ucm079803.pdf
5. PMDA Guidance on Data Standards: http://www.pmda.go.jp/english/review-services/reviews/advanced-efforts/0002.html
    a. Technical Conformance Guide on Electronic Study Data Submissions
    b. Data Standards Catalog

CONTACT INFORMATION

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

Michael Reich
PRA Health Sciences
Gottlieb-Daimler-Strasse 10
68165 Mannheim, Germany
Email: reichmichael@prahs.com
Web: prahs.com