PharmaSUG 2014 - Paper CC49

A Shout-out to Specification Review: Techniques for an efficient review of Programming Specifications

Sajeet Pavate, PPD, Wilmington, NC

Jhelum Naik, PPD, Wilmington, NC

ABSTRACT

One of the most common reasons for poor quality of programming output, higher costs due to large number of programming hours and missed timelines is poorly written programming specifications (specs).
A thorough independent review of the specifications by competent programmers and/or statisticians prior to programming activities is an important safety net to ensure the specs are in a suitable form for programming. This early intervention eliminates the risks to deliverable in terms of quality, costs and timelines.

This paper provides clear guidelines, tips and techniques to empower spec reviewers to perform an efficient spec review.

This paper provides techniques for spec review that can be implemented for different types of deliverables such as Transformed/Mapped Data, Analysis Database, Tables, Listings and Figures and for different types of organizations such as CROs and pharmaceutical companies.

The techniques discussed in this paper can also be adapted and used in organizations which have built-in standard specs – it empowers the foot soldiers on the project team to identify and raise concerns that may not be evident in standards generated by a working group or standards committee.

INTRODUCTION – IMPORTANCE OF SPECIFICATION REVIEW

Some organizations may forego the need for an independent spec review prior to programming in order to meet or reduce the timelines. This may produce a counter-effect wherein the quality of the product is not of high standard or may lead to frustration for the programmers working on poor or incorrect specs thus leading to overtime and risks to timelines.

Some organizations may include an independent spec review in their timelines. However, these organizations may not provide pertinent training or tools/techniques in doing a thorough spec review. They may also not emphasize the importance of spec review to their programmers. In such cases, the programmer may approach specification review as a step to be ticked off on a check-list rather than taking the opportunity to make sure the specs are in optimal shape for programming to begin. They may assume that all spec issues will be caught during programming. Unfortunately, if the specs are inaccurate, then this raises the risks during programming in some of the following ways:

- An inexperienced programmer may follow the specs on face value without proper analyzing the algorithms thus leading to a poor quality product.
- If specs are not clear, an experienced programmer may make assumptions in their programming that could be incorrect or the final code will not match the specs.
- It may lead to frustration as the programmer will have to spend more time communicating spec issues to the spec author and wait for specs to be updated before programming.

The issues mentioned above could possibly be caught during independent programming validation or during manual review of the output by Validator or a Statistical Reviewer. The point to emphasize here is that a programming spec review is an important first safety net that could catch a lot of these issues early on and reduce the time and costs involved in resolving them later.
An experienced reviewer should recognize this opportunity and perform a comprehensive spec review that will result in a collaborative and meaningful spec review meeting with the spec author and result in optimal specs.

An advantage of an efficient Spec review is that it can alert the team to the complexity of programming and thus the team can re-visit allocation and make appropriate changes to programming resources and/or timelines.

For CROs, spec review is especially helpful in providing stable versions of the specs to client for approval prior to programming.

TECHNIQUES – SPECIFICATION REVIEW

Now that we know why a detailed spec review is important, we can look at various techniques to achieve this result.

The techniques specified below give generic ideas that can be implemented for different types of deliverables such as Transformed/Mapped Data, Analysis Database, Tables, Listings and Figures. Some techniques can be applied to other deliverables such as Data Cleaning and Reconciliation Listings, custom updates to imports from external vendors or any other programming deliverables.

This paper takes a two-step approach towards spec review which includes (a) preparation to be done prior to spec review and (b) the actual review using programmatic methods and tips for manual review.

The first part is Preparation which should be planned as much in advance as possible. The second part is the detailed steps/techniques during the actual specification review.

PREPARATION – BEFORE YOU BEGIN SPECIFICATION REVIEW

1. The most important item is to allocate enough time to do a thorough spec review. Include Spec review timelines in your overall timelines. For complex specifications such as efficacy specs or complex SDTM domains, try to build some extra time for initial review and time for another round of spec review, if it is needed.
2. Try to group domains so that similar domains are assigned to one set. For example, in an Analysis Database specs, the Safety domains can be assigned to one set while the Efficacy domains can be broken into one or more sets if there are similar algorithms. In a CDISC SDTM study, the general observation classes can be grouped as: Interventions, Events or Findings.
3. Ideally you should have two resources such as programmers or statisticians independently reviewing the specs. They can plan in advance and assign each other to investigate certain algorithms in more depth.
4. The programmers/statisticians performing the spec review should be familiar with the study protocol, input/source data, Statistical Analysis Plan text (SAP), specs from another sister or similar study and any other necessary documents. They should also be familiar with the organization’s tools, standards and processes and industry standards as applicable.
   Ideally, it would be good to assign the programmers as the spec reviewers for the domains that they will be programming. This is not mandatory, if the specs have been reviewed thoroughly by experienced reviewers.
5. Reserve the resources with the correct fit. Assign programmers based on their experience level to like domains as per the group assignments in the step above. The experienced strong programmers should be reviewing the most complex or highest priority datasets. For example, in an Analysis Database, they should review the master ADSL domain and Primary and Secondary Efficacy domain specs. In Efficacy related deliverables, it is best to have reviewers with therapeutic knowledge and experience.

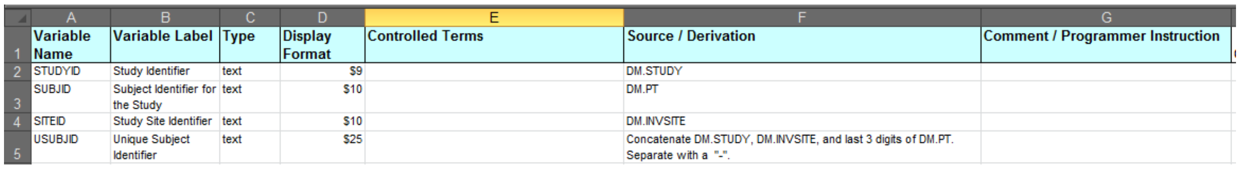
DETAILED STEPS AND EXAMPLES FOR EFFICIENT SPECIFICATION REVIEW

The following steps reference review of Analysis Database (ADB) as an example but it can be used for any programming deliverables.

1. If you have been assigned as a spec reviewer, make sure you allocate time only for spec review. This time should only be used for spec review and they should avoid multitasking during this time.
2. Make sure you have the latest copies of the required documents such as study protocol, SAP, specs etc. to assist in the spec review.
3. Review these relevant documents as well as structure of source data and/or raw data before you start the spec review.
4. Come up with the most efficient way to communicate spec review issues. It should be effortless for the spec author to go through the list of issues and not miss any comment inadvertently. It should be convenient for them to interpret to which variable the issue belongs to. For example, if the spec is in Microsoft Excel format, then entering the spec review comments in the adjoining column should make it easy for the spec author. They should also have the capability to respond to your comments. The comments should be searchable with the utility’s tools (if available).
5. Prepare macro programs/procedures that can research the source data to help out with spec review. Some examples are programs to show frequency counts of variables in the source data, simple merges of two

datasets, checks for duplicates in data which in turn can be used to determine the primary and secondary keys and any custom code that is study specific to test algorithms.

6. If the specs are in a format that can be read directly in to SAS, for example, Microsoft Excel, then it would be efficient to have a program to read in the spec to create a data set. If the spec is read in to a dataset, then several checks can be automated using macros.
   You can have simple checks to make sure that the length of the variable names and labels conform to either client standards or CDISC SDTM or ADaM standards. You can verify if any attribute is missing. For date, time and datetime variables, the variable names should be in a consistent manner. In such cases, you can verify that the attributes are consistent for such variables with the same length and format.
   You can also use this dataset to compare the attributes of the variables with the source dataset and report any discrepancies. It also becomes easier if you need to compare this spec to a spec from a similar study on the same compound or client standards. One example is given in Table 1 and Code Sample 1 below.



	A	B	C	D	E	F	G
1	Variable Name	Variable Label	Type	Display Format	Controlled Terms	Source / Derivation	Comment / Programmer Instruction
2	STUDYID	Study Identifier	text	$9		DM.STUDY	
3	SUBJID	Subject Identifier for the Study	text	$10		DM.PT	
4	SITEID	Study Site Identifier	text	$10		DM.INVSITE	
5	USUBJID	Unique Subject Identifier	text	$25		Concatenate DM.STUDY, DM.INVSITE, and last 3 digits of DM.PT. Separate with a "-".	

Table 1. Sample Analysis Database (ADB) specification spreadsheet

```
/*** Import the Analysis Database (ADB) specification spreadsheet using
     proc  import ***/
filename impxl "ADB_Spec.xls";
proc import datafile=impxl out=adslSpec dbms=excel REPLACE;
  getnames=yes;
  sheet="ADSL";
  mixed=yes;
  scantext=yes;
run;

/*** Check if variable names are greater than 8 chars in length or
     variable labels greater than 40 characters in length in the
     dataset created after importing the ADB spec ***/
data _null_;
  set adslSpec;

  Variable_Name=strip(Variable_Name);
  Variable_Label=strip(Variable_Label);

  if length(Variable_Name) > 8 then
    put 'WARNING: Invalid Length of variable name ' variable_name 'in
         ADSL spec';

  if length(Variable_Label) > 40 then
    put 'WARNING: Variable label is greater than 40 characters in
         ADSL Spec '  variable_name= variable_label=;
run;
```

Code Sample 1. Read in sample ADB Spec and check length of variable name and label

7. Look through the entire spec and verify to see that the variables are arranged in some kind of order: either variables that are read in from the same source dataset are kept together; or variables having similar algorithms are kept together; or variables that are dependent on other variables follow these variables. If these are not in order, include this in your spec review comments and order the spec in order to make your review efficient. If needed and available add a sorting key, so that you can order the spec back to the original order.

8. Check the variable name and label. This should give you a good picture of the expectation of the variable. Now check the derivation/calculation for this variable. It should be consistent with the expectation of this variable. Compare against the source data, study protocol, SAP as needed to verify that the calculation is accurate as per the expectations. Is the derivation written in a manner conducive to programming? If other

variables are dependent on this variable, then check those variables as well to make sure that their calculations are accurate.

9. If the spec is not clear at all or contains too much SAS code with detailed steps, then make a note asking the spec author to update the code to make it more consistent. If possible, provide suggestions.
10. The spec should be clear and specify which variables to use and not just add a reference to other documents such as study protocol or SAP. Otherwise, the spec will be open to interpretation. The spec reviewer may read the spec one way and the programmer another way during programming leading to issues in the final output. Make a comment if you see any such spec issues.
11. Check the variable derivation to see if any assumptions on the data are being made. For any assumptions, make a suggestion to add QC checks to ensure that there are no datapoints that fall out of these assumptions. These QC checks will then need to be programmed by the programmer or validator as needed.
12. If the derivation of a variable is complicated that cannot be verified in an easy manner, then make a note of it in the spec and pass this on to the programmer and/or validator that this variable was not reviewed so that they pay special attention to this variable while programming.
13. For Tables, Listings and Graphs (TLG) specifications check to see if the correct variable from the ADB is referenced. Check if the correct subset flags are referenced. Check that the values of the variables correctly correspond to the rows or columns on a TL. Check to see if any complex algorithm in the TLG can be efficiently done in the Analysis Database instead. Review the Title and Footnotes and makes sure that it is accurately represented in the TLG. Also, verify if any assumption or derivation needs to be in a footnote.
14. If you end up writing code during the SAS review, then make sure you pass the code to either the programmer or validator for them to use while programming.
15. Suggest Spec Review meetings with the spec author and other core team members. Provide your detailed comments prior to the meeting and come prepared for the meetings to discuss your comments.

CONCLUSION

This paper emphasizes the need for specification reviews to build a robust foundation of specs to ensure a quality deliverable with reduced costs and timelines. This paper attempts to provide tried and tested methods with some novel ideas that programmers/statisticians can utilize to perform a thorough review of the specs. We hope to inspire spec reviewers to perform to their best potential by applying these techniques with the knowledge that rewards will be reaped down the line by making a huge impact in downstream activities.

Happy Spec Review!!!

REFERENCES

CDISC Study Data Tabulation Model (SDTM) v1.3 and Study Data Tabulation Model Implementation Guide (SDTMIG) v3.1.3. Available at
http://www.cdisc.org/sdtm

ACKNOWLEDGMENTS

The authors are grateful to PPD colleagues for their support and guidance.

DISCLAIMER

CONTACT INFORMATION

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

Name: Sajeet Pavate
Enterprise: PPD
Address: 929 North Front Street
Wilmington, NC 28401
Work Phone: +1 910 558 8622
E-mail: sajeet.pavate@ppdi.com

Name: Jhelum Naik
Enterprise: PPD
Address: 929 North Front Street
Wilmington, NC 28401
Work Phone: +1 910 202 4761
E-mail: jhelum.naik@ppdi.com