PhUSE 2012

Paper PD05

How specifications get lost in translation

Ralf Minkenberg, Ingelheim, Germany

ABSTRACT

The first ideas for ad hoc analyses are born very often without any Programmers or Statisticians involved. Some examples are shown how specifications first are generated and then are developed with different functions involved. It is discussed if the final specifications the programmer get (generally from a Statistician) are still the same as wanted from the author, or are changed on their way to be programming-ready like Chinese whispers. A solution could be to involve the programmer in the specification process as early as possible. But is there a common understanding between specification requestors (key opinion leaders, drug safety, marketing, health economics, ...) and programmers at all? Example processes are described to show different ways specifications can be specified and clarified, and how programmers can handle them. Interaction between Programmers, Statisticians and Medicals are the key to get final analyses which fit the needs of all interested and involved functions.

INTRODUCTION

During your life as a Programmer in the pharmaceutical industry you will face very different tasks. But one common feature of all those tasks should be proper definitions and specifications, what should be done, how data should be prepared, how analyses should be run, how results should be presented, and so on. But another common feature, highly related to the previous one, is that definitions and specifications will be misunderstood, data would be transformed incorrectly, wrong analyses would be performed, and results would be shown biased. Although these issues appear from time to time in each task Programmers want to solve, the problem increases for analyses based on an ad hoc request and/or required by functions not involved in the daily business of data management, programming, or statistics. If in addition the specifications will be reviewed, commented, corrected, expanded by people from many different functions (with different views on the current problem) the likelihood of improper specifications (or at least gaps in definitions) increases significantly. Very often you (as a Programmer) feel as you are at the end of a Chinese whisper chain, getting only fractions of the originally request with many modifications and misinterpretations along the way. What really is wanted seems to be hidden somewhere in this chain – but how should you find the real needs of the authors in the final specifications?

PROBLEM EXAMPLES

Before trying to give some advice how to answer this question, let's look into some examples for requests which are (obviously?) changed during the specification process, but the Programmer is not (or cannot) be aware of some important discussions to finalize the request specifications.

Example 1. A paper just released in a scientific journal stated that a special adverse event, not considered to cause any harm in the past, seems to appear significantly more often in your company's treatment than in any other comparator. To reply to this unfavourable publication, the Drug Safety Department wants to analyse the mentioned adverse event through all available trials with patients under the suspicious treatment. This has to be done as soon as possible. Therefore, the responsible Physician writes down his request in an email to the responsible Statistician. He refers to the publication and points out the importance of getting reliable data as soon as possible. The Statistician now writes down more detailed specifications for the Programmer, in which the trials to be analysed are defined, which standard adverse event analysis tables should be created, which treatments should be displayed, … Finally the Programmer starts to write programs to satisfy the specifications, but faces many not considered problems. For example, the trials were reported in different MEDDRA versions, the defined search category for adverse events seems to be ambiguous, data of some trials to be included in the analysis are not accessible, … Now the Programmer starts to ask the Statistician, who then has to ask the original requestor if anything is unclear. This procedure continues, time is wasted, and the requested tables cannot be finalized in time.

Example 2. The results of an important trial are discussed on management level and additional post hoc analyses seem to be necessary for better understanding of some results. Ideas for such analyses are suggested from different people – the therapeutic area Physician, the Medical Affairs Department, the involved Statisticians, Upper Management, Drug Safety, … A detailed definition what should finally be analysed will be agreed in the minutes of the meeting. After review and conciliation the final minutes with the proposed analyses are forwarded to the responsible Statistician, who adds some details to the document before forwarding it to Programming. For many requests the Statistician has to ask what really should be produced because important details are missing in the minutes. He gets different answers, depending on who was asked, and tries to summarize all suggestions best as

possible. The Programmer, obviously, raises additional questions to get an idea what is recommended here. After some time with endless discussions with the Statistician he delivers a first version of results, hoping that they match the original needs as far as possible. After sending out these results with additional comments from the Statistician it is no surprise that he gets back at least as many questions as tables produced. Again he is not able to understand some of the comments, therefore, asks back to the Physicians etc. Again the Programmer gets new requirements to change already created tables and to add new tables. This process continues several times, most of the time wasted by missing mutual understanding between different functions and by transcription errors when statements are forwarded from one to another.

PROBLEM SOLUTIONS

There could be many more examples showing all one serious problem. The origin of analyses are very likely found by persons not familiar with the problems we encounter during accessing the needed data, due to not unique definitions what should be done, and during programming tables which show the requested answers. Therefore, when Programming gets specifications many people have been involved before, many ideas have been born (and also died), many discussions have taken place. Sometimes it looks like a Chinese whisper with the Programmer at the very end – what really is wanted (and what is the rationale behind the requests) is completely lost. Lost as the Programmer who tries to create high-quality output in shortest time with specifications giving sometimes only hints and no clear definitions.

If a request is to be fully understood, it is not sufficient to get instructions which data should be accessed, how they should be transformed and analysed, and how they should be presented. To be more than a "servant" of a Statistician (or any other function), knowledge about the background of a request is essential. With a reasonable background Programmers can judge (or at least help to do so) if specifications really match the original request, if the analyses will be done in a proper way, and if the presentation of the results will be helpful for the request authors.

To fill this lack of information seems to be easy. The Programmer should be involved in the discussions and meetings which will comprehensively define analysis requests to be finally produced by the Programmer. But it is not as easy as it sounds. Firstly Programmers have to be accepted as equal partners in meetings with Statisticians, Data Managers, and Medical people. This would be the only possibility that they can give valuable input on discussions and, therefore, accelerate request specifications (and output, too). Programmers can give these input – they are very likely the ones who know best the data problems during analyses, missing definitions in requests, ambiguities in formulas, etc. If other functions dealing with the data on which requests are based, especially Data Management and Statistics, are also included in the process of discussion, design, and specification there will be a great chance that from the beginning until the end clear specifications are done, what is wanted, and how it is wanted. The Medical people will better understand which problem we face with the data. On the other hand we will better understand how Medics think, how they analyze our outputs, how they want to present results.

If this is what Programmers want to do, they will have to expand their knowledge in many areas. Good programming skills are not the only prerequisite. Good knowledge of (at least) basic statistical methods (e.g., statistical testing, analysis of variance, regression analysis, …) would be crucial. You should also be well informed about the basic medical concepts in the therapeutic area you are currently working. Besides this (more general) knowledge in medicine, some ideas how your current drug is acting, which side effects are common, etc. are very helpful. You should have multidisciplinary background, be able to explain your work (and esp. your problems) people outside your subject area (e.g., Medics), and also be able to understand questions coming from those people. Programmers should have life-time training not only in SAS (or other software products), but also in Statistics and Medicine.

As still the main contact for Medics should be the Statistician, in all regular and ad hoc discussions, on which additional or new analyses could be defined, Programmers (and Data Managers) should be a mandatory part. The requests can directly be formulated for the one who will have to create the request outputs, and all issues not obviously seen by Medics (or other involved functions) can be discussed immediately.

CONCLUSION

Interaction between Programmers, Statisticians and Medics are the key to get final analyses which fit the needs of all interested and involved functions. Only if all functions which are able to give input to any request are involved as early as possible, can you be sure that everyone at least has the opportunity to understand what is needed, why it is needed, and how it should be presented.

CONTACT INFORMATION

(In case a reader wants to get in touch with you, please put your contact information at the end of the paper.)

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

Ralf Minkenberg
Wackernheimer Str. 7
55218 Ingelheim
Email: r.minki@web.de