PhUSE US Connect 2019

Paper SD03

Making the Case for Metadata Development

Mike Molter, Wright Avenue, Cary, NC USA

ABSTRACT

Requirements imposed by clinical trial data standards have changed the way many in life sciences do their jobs. For example, no longer is it sufficient for SAS programmers to bury programming specifications and other metadata in SAS programs, or communicate them through informal emails, documents, or water cooler conversation. Being held accountable for documenting the design of our submission data sets, we owe it to ourselves as individuals as well as companies to prepare our metadata with as much structure and organization as we can. In this paper, we'll look at a web-based application that accomplishes this goal by doing away with traditional monotonous spreadsheet data entry in favor of conversational dialog between system and user. Also replacing the spreadsheet is a database to which access is controlled through the web interface. Finally, we'll see how such preparation facilitates metadata-driven downstream processes.

INTRODUCTION

While CDISC data standards such as SDTM and ADaM have been delivered to the industry by way of PDF files referred to as "implementation guidelines" (IGs), in reality, it would be difficult to describe such documents as *implementation standards.* At best, they can be seen as partial programming specifications combining hard requirements with flexibility for study variation. Because of this, the implementation and use of these standards, and ultimately a study's compliance, is in the hands of stat programmers. For the most part, for a number of reasons, not the least of which is the current workload and set of responsibilities of the stat programmer, that implementation has been based on what appears to get the job done fastest for the current study – transcribe what is documented in the IG, possibly into a programming spec in Excel, and then again into code in a SAS program. Efforts such as those by CDISC SHARE to make these standards available in Excel or XML made these tasks more manageable, but many issues, including those associated with the use of Excel, remained.

The more recent requirement of XML metadata submission shares many of the same challenges. Combining the fact that the define.xml requirement came later with the fact that it is a submission document has led to a common practice of "worrying about define.xml later." This "wait-til-later" approach has had a tendency to put define.xml into its own box, to be opened only when everything else is finished, closer to deadlines. At best, this approach has those responsible for its creation re-visiting decisions that were made at the time of data set design, hoping to find answers in programming specs or worse yet, SAS programs. At worst, particularly under tight timelines, the define box is a box to be checked, where the goal is to do the least amount of work within compliance guidelines. In this scenario, concerns are more around compliance than telling the story of the study.

In the hands of stat programmers, implementation solutions trend toward what is fastest, easiest, and most familiar. The Excel workbook is a three-dimensional file with rows and columns that function the same way they do in a database table, and tabs that together look like a collection of tables in a database. Everybody in the organization has access to it, and it's easy to use. But do these features make it the best tool to use? SAS programmers look at the IGs and see a program in their head. Some may even think SAS macros. But is programming difficulty really the implementation issue here? SAS macros are code generators. Is that what we need, or is there something deeper in the process we should be addressing?

At Wright Avenue we have been pondering the question, "is there a better way?". In this paper we invite you the reader to join us on this journey. I'll share with you how Wright Avenue is re-thinking the concept of study design in an effort to efficiently and accurately implement data standards, while at the same time, tell the story of your study. You'll see how these efforts are being manifested in new technology with an application we call Study Designer. You don't need to be an expert in new technologies to gain anything from the paper, but it does help to have an open mind that can see beyond the traditions of the past

METADATA DEVELOPMENT

The requirement of an XML document as part of your submission package to document information about your study and its data set the industry, and particularly, those in biostats departments, into a mild panic for several

reasons. For starters, just the addition of another new task – the documentation of study data – on top of an already heavy workload which has already increased due to SDTM and ADaM data standards, looked like more than these departments could handle. Worse yet were the three letters – X-M-L – spelling out a technology, or a computer language that was never part of these departments. That, along with requirements specific to ODM and define gave the appearance of a steep learning curve. To nobody's surprise and everyone's delight, a handful of macro developers and third-party vendors that were ahead in the learning curve came in to save the day, seemingly relieving the rest of the programmers from having to take it on.

Lack of XML knowledge or the specific requirements of define are among the more obvious, and to some degree valid, reasons for panic, but what may be the most valid, and maybe most overlooked challenge is the development of the metadata itself. After all, XML is a text-based language with only a handful of rules that most can learn quickly. Because it's only text, the SAS data step can generate it with basic statements such as FILE and PUT. Yes, there is a brief learning curve to know how the elements and attributes go together, and what metadata goes where. The tools and macros that are available handle all of this for you, but most, if not all, leave it up to you to provide the content. That, in my opinion, is the hard part.

The development of metadata may be one of the most daunting tasks we face, but not only for technical reasons. After all, we know how to work with spreadsheets, and if you're creating your own SAS-based tool, you probably know how to import from spreadsheets without too much trouble. The real trouble starts when we begin to think of volume. Estimating conservatively, if a data set has 20 variables in it, and we need roughly 10 pieces of information about each variable, and we have, say 20 SDTM (or ADaM) data sets, then that amounts to at least 4000 pieces of information we are responsible for providing in our spreadsheets. And that's only variable-level metadata, and so doesn't include data set-level metadata, value-level metadata, controlled terminology, and others. In the end, a define.xml will most likely contain tens of thousands of pieces of information. Industry spreadsheets such as those provided by CDISC SHARE and NCI help to pre-populate some of these spreadsheets, but having to review what is pre-populated, determine what needs to be changed for the study, copying from these files to your spreadsheets, all amounts to management and movement of a significant amount of metadata with a tool like Excel that doesn't know anything about what you're doing. As we can see, the challenge of developing metadata is both a technical *as well as a process* challenge.

Everything we've said up to this point in this section about metadata development has been stated inside of a context in which metadata is simply a submission requirement. For many, this justifies a process that budgets time and resources for a define.xml process to begin after database lock and creation of all SDTM/ADaM data sets. However, as we'll see, by thinking of metadata only in these terms, we're missing the bigger metadata picture. In the remainder of this paper, we'll see how an expanded vision of what metadata is can lead to important uses of metadata, thereby making an efficient and intelligent process of metadata development, accompanied by smart technology, that much more important.

THE WHAT, WHERE, AND WHEN OF CLINICAL TRIAL METADATA

Before we go any further, let's make sure we are on the same page with regards to what clinical trial metadata is. In the spirit of keeping it simple, let's start by saying that metadata is *information about your study and its data*. With this definition, even if you're accustomed to thinking of metadata only as a submission artifact, you can see that metadata is all around a clinical trial, and it enters into existence at several different time points in a trial. Let's look at a few examples.

- One place we find metadata is in the study protocol. The study protocol is a text document in paragraph form, much like this paper, normally stored in a PDF or word processing format, that simply informs the reader about the study (history, conduct, schedules, etc). Put another way, it is the documentation of decisions made by those designing the study at the time the study is designed.

- The statistical plan (SAP) also contains metadata in a PDF or word processing format in paragraph form. Its content reflects how data is to be analyzed. Like the protocol, we can also think of it as a documentation of study decisions when analyses are designed.

- TLF shells, found in the SAP or a separate document, can be thought of as information about how analyses will be presented in a clinical study report.

- The IGs can be said to contain metadata according to our definition above. As we know, this information applies to all studies using the standard, but anything the IG notes as required can also be thought of as study metadata.

- We can even think of SAS programs as containing metadata.

While each of these formats contains information about the study and its data, the only practical way of consuming such information is to simply read it with our own eyes. As mentioned earlier, to put it to use means transferring what has already been transferred from the source to your brain through the act of reading it, into another format through the act of transcription. While this process can be successful for the short term, in the long term it can cost more time and lead to errors and inaccuracies. For that reason, it is usually more beneficial when the information in our definition is collected and organized into a data format that software programs can read. This quality is referred to as being *machine-readable*. Because of this, metadata is sometimes defined more precisely as ***data*** *about your study and its data*. Below are some examples.

- Today's electronic data capture systems (EDCs) collect, organize, and store information that was collected at study sites and other study vendors. In addition to this, metadata is stored in a highly structured XML format called ODM (Operational Data Model). This is not the submission metadata, but the two do overlap.
- In the same way that the IGs contain metadata defined as information, industry spreadsheets such as those provided by CDISC SHARE and NCI contain machine-readable metadata.
- Of course define.xml also fits into this category. Define is stored in a structured XML format that is based on the ODM format. While tools such as XSL stylesheets are available to make consumption by reading/viewing a browser are available, the structured XML makes it possible to create software that consumes it and allows anyone to use the content in other ways.

All of the above bullet points represent different sources of metadata throughout the life cycle of a clinical trial, but of course it's this final bullet that is in the hands of the biostats department and necessary for submission. Most of the other bullets represent sources from which we can pull to assemble our define.xml. There is one more source of metadata that belongs in the second list that deserves special attention - the programming specification.

Too often we don't think of programming specs as metadata, and to do so leaves us missing out on some opportunities. At a minimum, a programming spec is everything a SAS programmer needs to know to build a data set. This includes a handful of data set and variable attributes, as well as programming instructions detailing what kind of values belong in the variable. The obvious usefulness of such metadata is that it serves as a structured method of communication from data set designer to data set programmer.

If we take this minimalist approach to programming specs where the data set programmer is the only consumer, then we miss out on the opportunity to document other information for other consumers (i.e. regulatory agencies) in real time when decisions are being made. Like define.xml, the programming spec is the responsibility of the stat programmer. Short-term solutions to standards requirements without the benefit of time and resources to consider more robust long-term solution, has been to develop minimal content for each of these documents at the time they're needed. So while both documents reflect, at least in part, decisions made about data set design, development of them tends to be separated by significant time, since define is only needed at submission, long after data sets were initially designed. The following are just some potential consequences.

- An ADaM designer creates a flag variable populated with 0s and 1s. The designer includes specific programming instructions in the spec for the programmer, but does not include a verbal explanation of what the 0s and 1s mean. While this may seem reasonable for programming purposes, this information would be useful to an FDA reviewer. Later, when developing define, we have to go back in time to recall this information and record it in our define metadata, long after the design decision.
- An SDTM designer may include a description of a variable's derivation, but without explicitly noting that the variable's values are derived, as opposed to copied or assigned, we leave it to the define developer later on to go back and extract this information by reading the spec.
- A data set designer may enter a free text programming instruction of the form "if x, then 'a', else 'b'", which is useful to a programmer, but a define developer has to read into this instruction to see that a define codelist (list of allowable values required for applicable variables in a define) is present, and then manually document this codelist in their own metadata.

With these examples and plenty more, the disadvantages of an approach like this become obvious. Too much time is being spent re-inventing the wheel, re-visiting decisions that were made long ago because we didn't take the opportunity at design time to document everything about our decision. In addition, we're working with technology that requires significant amounts of manual work (data entry, copying/pasting, etc.), and that isn't able to keep us in compliance with the standards, thereby inevitably leading even the best and brightest standards experts to inadvertent errors. All of this costs time, which ultimately costs our clients money, and sometimes costs us our reputations. Delays and missed deadlines due to unforeseen circumstances will never go away, but in this century, deliverables should never be late due to outdated processes and technology.

At Wright Avenue, we're asking questions about better approaches. For starters, what if we start collecting *all* metadata at one time, as decisions about data are being made? Clinical trial metadata will always contain tens of thousands of data points, but how can we develop it more efficiently than entering data into spreadsheets? What is the most optimal way to leverage the standards, and how can we best combine standard requirements with study flexibility while staying compliant? In the next section I'll share with you tales of a journey toward answering these questions that we're still on.

A BETTER WAY – RETHINKING DESIGN AND TECHNOLOGY

When a team of homebuilders builds a house, they do so under the guidance of a blueprint – a plan that illustrates how all of the components fit together. What they don't do is build the garage, then put all of their tools away, get rid of any unused supplies, throw away their garage plans, and then start anew on the first floor bathroom. When you write a term paper, you don't do so one paragraph at a time. Rather, you develop a theme for the paper and a plan – a design – for how to support that theme through the flow of sections and paragraphs. When you're in charge of cooking for a dinner party, you don't shop only for the ingredients for the main entrée, go home and make it, then start making plans for the sides, then the appetizers, then the dessert. Instead, all of the individual components are treated as part of a bigger whole. In all of these cases and countless others, when we plan for a project in this way, the result is something greater than the sum of its parts.

At Wright Avenue, we're exploring what happens when we take the same approach with study design within the biostats realm of a clinical trial by *re-defining design*. As noted earlier, biostats is responsible for metadata that surfaces at two distinct points in time in the clinical trial data process. Historically, the quickest and easiest way to implement standards for the current study has been to open a blank Excel workbook and manually enter programming instructions based on what's in the Implementation Guidelines – and worry about the define.xml later. Unfortunately, this short-term solution has lasted for more than ten years. What we're discussing in this paper is not the design of specs or the design of define metadata, but rather, the design of the study as a whole, manifested in metadata. And just as each individual dish of a dinner party is produced from an overarching plan, just as each room of a house comes from the blueprint of the house as a whole, the programming spec and the define.xml are products, not only of a grander database of metadata, but also of a much richer concept of design and metadata. This is being accomplished through the ongoing development of a web-based application called Study Designer.

STUDY DESIGNER

Study Designer is defined by two primary features, the need for which can be seen clearly upon opening a new Excel workbook. These features are user experience and storage.

Thinking of a "user" (i.e. designer) as the one in biostats who designs the studies (metadata), user experience is what it sounds like – the experience of developing tens of thousands of metadata data points, some of which are pre-determined by clinical data standards. The user experience with Excel is one of unsolicited, unguided data entry, hoping everything is entered completely, accurately, and in compliance. The requirement for so much metadata doesn't change across all of the available tools for metadata collection, but through secure storage of the standards in combination with a user interface that displays required standard metadata and *solicits* study-specific metadata, Study Designer turns the user experience into one of a storytelling nature, and a dialog with the software.

Metadata is the story of our study. When we share a story with someone through dialog, particularly one with complexity, it's oftentimes most effective to start with high-level information, and then work our way down into the details. Study Designer solicits a story by first asking high-level questions. What is the name of the study, the name of the protocol? How would you describe this study? What version of Meddra is being used? What study documents are of interest (e.g. reviewer's guides, annotated CRFs, etc.)?

As the user moves through the dialog, the questions gradually move from a high level like those above to more and more detailed levels. On the SDTM side, next up are trial design-level questions about trial arms, inclusion/exclusion criteria, trial phase, trial objectives, age range of subjects, etc. Questions about the visit schedule allow the designer to lay the groundwork for VISIT variables in SDTM domains. Answers to all of these questions go a long way toward populating SDTM trial design data sets. They also lead a designer down a design path that starts with getting to know the study. Given enough up-front time, it's not unusual for designers to browse through the protocol. Study Designer gives them a chance to document what they learn, storing it for access later.

Eventually the user is ready to design study data sets. On the ADaM side, this starts with the creation of ADSL. As the designer gets more and more into the technical structure of data sets and variables, Study Designer attempts to ask questions in a manner that is consistent with how we think about these structures. At the data set level, not only are we questioned about data set-level properties such as data set name and label, but also about structure. Questions about what makes a row unique (i.e. one record per subject) are required in our define.xml. Also, what are the record source data sets – data sets from which records are derived? From a SAS perspective, these are

the data sets listed on your SET statement. Answers to these questions help build a programming spec from which programmers can easily begin to build code.

When a data set's properties and high-level structure have been defined, they can then move on to the data set's variables. CDISC defines variables in groups (e.g. subject demographics, treatment variables, analysis parameters, etc.). Upon initial design, some groups of variables are presented to the designer one at a time. For example, since it's common to first define identifier variables (e.g. STUDYID, USUBJID) – the low-hanging fruit – this group of variables might be presented first. When finished with these, the designer then moves on to the next group.

One of the first pieces of information that a designer thinks about and communicates to a programmer when defining a variable is the source of that variable – how does that variable come into existence. Are its values copied from values of another variable? Are they derived from one or more other variables? If so, where are these source variables? If derived, what does this derivation look like? Alternatively, do a variable's values spring into existence out of the designer's head (e.g. –TESTCD)? Once again, not only are answers to these questions documented in define.xml (Origin, Methods), but the process of asking and answering them, along with the sequence in which they are asked, is intended to guide a designer down a natural path of study and data development, while guaranteeing that biostats has everything it needs for programming and submission.

Study Designer attempts to identify and address concepts, without using technical jargon, that are necessary for documenting at submission, and that are sometimes casually buried in free text programming instructions in a specification. One example of this is controlled terminology. While Study Designer has a designer thinking about how a variable is built, it invites you to think about what the values should look like. The standards tell us that certain variables are each associated with a set of allowable values (i.e. a codelist). Study Designer knows which variables (and values) these are and brings it to the designer's attention. Additionally, even for variables without this association, it invites you to custom-define a codelist. When a programming spec for a variable defined by a finite, discrete set of values is entered into a spreadsheet without any guidance, we often miss the opportunity to extract these codelists for later purposes. Instructions of the form *if x, then a; else b* might be easier to see the codelist in an Excel spec than other forms, but such a form is made an option by Study Designer, so that the codelist is made available for later purposes.

In an effort to make the tool as intuitive and natural with respect to the design process as possible, Study Designer tries to stay away from technical jargon that sometimes causes confusion. Controlled Terminology is one such term. The concept is simple – identifying allowable values for a variable – and should always be considered in variable spec development. Study Designer provides a user experience where the tool explicitly asks you the designer not only to consider this, but also to document it, in real time. Another such concept is value-level metadata.

Value-level metadata is one of those concepts that designers would prefer to avoid. Some may even think of it as something they only need to do because it's required in define.xml. The fact is though that that couldn't be further from the truth. The CDISC requirement that measurement results, recorded in SDTM Findings data sets or ADaM Basic Data Structure (BDS) data sets, be "stacked" into one results variable, sometimes from multiple raw source variables, inevitably leads a programmer to the necessity for define programming instructions for a variable like –ORRES in subsets. Communicating to the FDA information about a variable like AVAL can be done effectively only by specifying a parameter context. When Study Designer knows that a designer is defining a Findings or BDS data set, knowing that such a data set is centered around parameters or tests, then part of defining the overall data set structure will involve defining parameters (or tests). Which parameters do you want? Do you want to define parameter categories? Numeric parameter variables? Do you want to create DTYPE records (records in an ADaM BDS data set that summarize a group of records using a DTYPE variable)? When these questions are answered, variable-level metadata for the desired parameter variables is automatically created. When defining variables, it will ask if the definition of a chosen variable should be done so in subsets based on the parameters already defined. Often overlooked are instructions for variables for derived records. Study Designer knows which DTYPE records are being defined in a BDS data set and explicitly asks you how any variable is to be populated for these records.

FOR THE TECHIES

The second main feature of Study Designer is storage. The tool knows standards such as variable names and labels, which of them are required, and which are associated with codelists because this information is kept behind the scenes in a secure database. Study metadata is also stored in a database. As the designer answers questions while progressing through the interface, the information entered is then stored. It is only through answering these questions that write-access to this database is granted to any user. This is in contrast to traditional spreadsheets. Earlier we pointed out that Excel looks like a database, and that everyone has it and it's easy to use, and we asked if these features made it the best tool of choice. In other circumstances this may be the case, but in a highly regulated industry in which we are required to adhere to metadata standards, we owe it to ourselves to protect our

metadata the best we can. Keeping it in easy-to-use writeable files that can be attached to emails and stored on network drives accessible to everyone in the department falls short of this obligation.

Rather than using a traditional relational SQL database, Study Designer makes use of what is called a *graph database*. Ironically, some might say that a graph database is more relational than a traditional relational database. Traditional relational databases are made of tables with data, and that's it. Relationships between tables are abstractions designed by the database designer through special fields (i.e. variables) called keys that must be exploited by programmers that query the database. In contrast, relationships are concrete mechanisms of a graph database. Rather than tables, graph databases contain nodes, each of which may or may not contain any number of properties. We can think of a single node as a record in a table. We assign labels to these nodes, and so we can think of the collection of all nodes with the same label as a table. Relationships can be pictured visually as edges that connect nodes. Like nodes, relationships can also contain properties and labels.

The name of the database we use is Neo4j, and the proprietary query language that accompanies Neo4j is called Cypher. In some ways, Cypher contains a few superficial similarities to SQL, but for the most part is altogether different. With an ASCII-based syntax that is meant to reflect the relationships queried, the Cypher query begins by identifying the graph pattern to be found in the database. Once found, Cypher can then return property values or create new patterns. While small databases may show no performance difference from an SQL database, databases with significant amounts of relationships will run faster in graph databases. This is because while SQL joins (or SAS merges) have to process at runtime, the nature of a relationship as a concrete object means that this processing doesn't have to take place in a graph database.

Behind the scenes, Study Designer is powered on the front end by traditional tools such as HTML5, JavaScript, and CSS. On the server side are Python scripts, and pulling it all together is a web framework called Django. On the front end, the user sees a combination of a handful of HTML files combined with several modals where user input is solicited. JavaScript is used to collect the input and store it in hidden variables in the web page until a form is submitted. At this time, through the web framework, the values of these hidden variables are then passed to server side Python scripts. Most often the main job of the Python script then is to use this information to generate Cypher code, make a connection with the Neo4j database, and then either make modifications in the database or query the database. After this is complete, the script then renders a new HTML page.

At the end of the day, when all is said and done, the study team has a comprehensive pool of metadata stored in a secure database. The house blueprint is now made, the ingredients for a five-course meal have been purchased and the recipe cards are laid out in front of you, and an outline for the term paper illustrating a flow of sections and paragraphs that support a common theme is complete. The programming spec and the define.xml can be seen as two standard query outputs stemming not from isolated efforts that live within their own boxes, but rather as two products, two related offspring of a more grand, robust metadata concept.

While the biostats personnel who designs data sets and/or submission data is not typically one who actually designs the study from the very beginning, the name of this tool is meant to convey the simulation of such a role. At the time of this writing, Study Designer production has been focused primarily inside of biostats on the development of ADaM metadata, with an eye on SDTM in the near future. Long-term goals are centered around widening this scope to include more of the clinical trial data cycle. This will include the development of TLF/analysis metadata further downstream. We also hope to expand it far enough upstream to put it in the hands of those that are making study design decisions at the time they are making them.

CONCLUSION

It has been my intention throughout this paper to convey the notion that metadata is more than just a submission requirement, and that define.xml is more than just the document that contains that requirement. Metadata is the language in which our study's story is told. When someone wants to know about our study, the metadata is the way we talk about it. Define.xml is simply how we formally organize that story, so that not only people, but also software can consume it. When that someone is a regulatory agency like the FDA, someone who has influence over whether our drug is approved or not, then we want to be sure to tell that story as clearly as we can, with as much detail as we can. This focus is easy to lose when you're trying to get through tens of thousands of spreadsheet cells. We like to think though that when a database has the standards, and the interface draws you into a story about the study, then that story can also be made clearer for those that need to hear it.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my boss, Bill Donovan, founder and CEO of Wright Avenue, for supporting my efforts to design, build, and bring to market the innovations discussed in this paper. I would also like to thank the members of our team ranging from marketing to technical experts to investors, all of whom have played a significant role.

CONTACT INFORMATION

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

Name: Mike Molter
Company: Wright Avenue
Location: Cary, NC
Email: mike.molter@wrightave.com
Web: www.wrightave.com