PharmaSUG 2012 – Paper IB03

Metadata: Some Fundamental Truths

Frank DiIorio, CodeCrafters, Inc., Philadelphia PA

ABSTRACT

An essential characteristic of a healthy workplace is its ability to increase and diversify its work load without compromising product quality. Workflow volume and quality can often be addressed by adding to existing staff and improving the computing environment. However, when the work *flow* becomes a work *deluge*, the need for a paradigm shift becomes apparent. Metadata – data describing corporate processes and data – is a potent tool for making this shift.

This paper discusses a number of conceptual issues related to the design, implementation, and growth of metadata-based systems. It identifies situations where metadata can improve processes and suggests how to evaluate both the benefits and costs of implementation. The treatment of the topic is necessarily abstract, with minimal product references and coding examples. The reader will gain an appreciation of the power of metadata-driven workflow, and will get an eyes-wide-open understanding of what resources need to be expended in order to achieve it. Although the scenarios used are from the pharmaceutical industry, the larger, take-away points are applicable to all sectors.

INTRODUCTION: GETTING TO TIME PRESENT

Perhaps the most useful way to understand the significance and benefits of metadata is to examine the change in the volume and complexity of workflow leading from data collection to an electronic submission ("eSub") to the FDA.

Time Past. Consider this admittedly generalized and simplified description of the eSub process in earlier, simpler times. Data was manually collected, transcribed from paper CRFs into databases. Programs that generated tables and figures either directly used these datasets or ran against datasets that were in a format more suitable for analysis than those created by the data capture system. Instructions for the data transformations and the variable attributes were usually stored in files in word-processing formats – electronic, but not easily readable by the programs. Programmers manually entered each variable's data type, length, label, and other attributes. Documentation of datasets delivered as part of the submission required yet another manual process. Finally, the case report form (CRF) had to be manually annotated with the dataset and variable names of each item collected on the form.

Also consider the FDA reviewer, who typically reviewed data and documentation from several companies. Dataset names and structures varied with each eSub, as did the quality and amount of documentation. The reviewer's context-switching ability, and thus the precious time required for evaluation, was severely taxed by different names for the same idea: BLBPS and BASESBP to represent baseline systolic blood pressure, for example.

The process described above is notable for several reasons. First, it required a significant amount of manual transcribing and copying of data attributes. This was a time-consuming and error-prone process, both in initial coding and for updates. A second, related feature of the process was the inability of dataset, table/figure, and other programs to programmatically access specification documents. These work flow characteristics led, in turn, to a third problem: slow and inexact reaction to the inevitable specification changes. Fore example, if a footnote used in 15 tables was changed, the change had to be communicated to the programmers and then made in the 15 tables. Features in the SAS® language (macro, SCL) lent themselves to streamlining this and other processes, but on balance, the work flow that moved a data point from a CRF to a reviewer's desk was labor-intensive, hyper-sensitive to change, and slow.

Time Present. Fast-forward to the present. Use of paper CRFs is on the decline, replaced by Electronic Data Capture (EDC) systems that improve data quality and decrease time to data lock. The naming and attributes of the collected items are often dictated by industry standards whose use is encouraged or mandated by the FDA. Notable among these standards are CDISC's SDTM and ADaM.

Adoption of these standards has significant implications for a project's critical path: ADaM datasets, for example, must be built using SDTM datasets (rather than EDC data) as input. This means that a new layer of data flow is

introduced, which requires increased coordination between data management and programming staff. The production of deliverables is also affected. Documentation is typically produced in the traditional PDF format and the new define.XML standard. Here again, work flow is affected. In this instance, however, not only is there more work (creating define.xml), but the new deliverable uses proven but possibly unfamiliar XML technologies (XSL, XPath, schemas, *et al.*). The organization producing the define file needs to acquire entry-level fluency in these technologies.

Sidebar: The CRO Factor. The organizational stress imposed by increased volume and complexity of work is even greater in the CRO environment. A pharma company can identify its Best Practices for submission-related datasets and documentation. These can cover everything from directory structures to macro naming conventions to methods for CRF annotation to the color of hyperlinks in define.XML. A CRO has a similar collection of Best Practices, but these have to accommodate the often wildly disparate needs of the clients. Being responsive to multiple clients' requirements makes the CRO's work flow much more complex. Adaptive systems at all phases of the submission process are essential for the viability and success of the CRO.

Coping Mechanisms. It's daunting: compliance with standards; new technologies; increased volume of studies and deliverables. How does an organization handle these and other changes? Traditional responses have included primarily:

- *More Technology*. The company can introduce technology that improves network connectivity, provide faster computers, enable batch submission of programs, and provide remote access for telecommuters.
- *More Staff.* The company can add employees or contractors. Training and other start-up costs need to be absorbed, and there must be at least the tacit belief that the larger staff results in more "signal" than "noise."

If Time Present was simply "Time Past, Only More So," then these responses might be adequate. However, the life cycle of a data point has changed so dramatically in the last decade that the traditional handling of new challenges is, at best, shortsighted. So how else could an organization respond to the demands of Time Present?

- *New Systems.* At some point, the SAS-centric approach fails to keep pace with the scope and volume of deliverables regardless of how much staff and computing power is added. Many non-SAS products now improve data capture and other parts of the deliverables' life cycle. Adoption of these products is not for the faint of heart or light of checkbook, but, if properly measured against the current and expected needs of the organization, can pay off handsomely. Their use leads us into the next point.
- *New Processes.* Work flow can be modified whether or not new systems are implemented. Personnel can be reorganized to better align with new deliverables. As creation of new deliverables becomes more routine and better understood, parts of the production can be identified as candidates for automation. Automated annotation of case report forms is one such candidate.

An organization can introduce a significant number of new systems and completely reengineer work flow. Or it can dip its corporate "toe" into the system and process the "waters" gingerly. However it reacts, there is likely to be a common thread running through whatever changes are made: **metadata**.

METADATA: THE FOUNDATION FOR QUICK AND COST-EFFECTIVE CHANGE

If *any* system or process is to have a positive impact on costs and work flow, it must be generalized. That is, it must be able to react to different types of proper usage without requiring program modification. Metadata is one of the key mechanisms that enables this generalization. It holds the description of the environment in which the system operates. *Anything* related to the management and components of workflow can be held in metadata and accessed by generalized tools. Let's consider two scenarios that help us understand what metadata is and how it can be used.

Scenario 1 – TFL Headers and Footers. This is something that any statistical programmer has experienced: a group of displays contains a footnote describing the derivation of a p value. The wording is determined by a statistician and communicated to the programmers tasked with table creation. The text is entered into the first program, then copied to the other programs. The process is tedious and error prone. Was the footnote inserted in all the appropriate programs? Was the new text entered correctly? Answering these and similar kinds of questions is not how programmers want to spend their time.

Then, inevitably during the course of the study, the statistician realizes that the footnote wording needs editing for clarity and/or accuracy. The text is updated, and the changes are, in turn, made to the affected programs.

A metadata-driven approach to this scenario would move the footnote text out of the individual programs and into a programmatically accessible location such as a SAS dataset or an Microsoft Access database. This method of storage is a step forward, but is effective only if the programs needing the footnote have a simple way to use it. Thus tools for moving metadata out of storage and into an application become vital. Rather than containing hard-coded text, programs use a tool to build the footnotes:

```
%hdrFtr(display=T13.1.1)
```

The macro opens results-level metadata for Table 13.1.1, examines the footnote list for the table, then opens the footnote table and stores the footnote text in a macro variable. The beauty of this solution is that no matter how many times the text is changed, the program creating the display remains the same. Manual cut-and-paste becomes a unmourned relic of the past.

Scenario 2 – Transport Datasets. The cornerstones of the FDA submission are the domain and analysis datasets. Their structure and contents are precise, and the origin of every variable needs to be identified. Which CRF page and EDC dataset does it come from? If not a CRF variable, then how is the variable derived? What are its data type and label? What is its position in the dataset? Similar questions arise at the dataset level. What is its label? How is it sorted? What variables uniquely identify an observation? The list of questions is not endless, but it *is* long and demands consistent attention to detail.

The metadata-free approach to this process in Time Past was classic brute-force programming. Dataset and variable attributes were entered manually, usually using specifications that came from a document in Microsoft Word or a similar, proprietary format. The process was labor intensive, tedious, and sensitive to changes of attributes and derivation.

Fortunately, that process is also one of the easiest to reengineer using metadata. As in the previous scenario, the metadata design objective is to move data out of the program and into a programmatically accessible location. Features such as dataset and variable labels, variable type, label and position in the output dataset are obvious candidates for migration to metadata tables. Less obvious – because they aren't part of the program being abstracted – are indicators for the variable's inclusion in the output dataset and text describing how the variable is derived.

The bare-bones metadata described above is the basis for tools that can used throughout the eSub work flow. A macro can create an ATTRIBUTE statement using variable-level metadata. The same tool could also create a KEEP statement, thus ensuring that only the expected variables are written to the output dataset. The program using the macro can remain largely unchanged, since changes to items such as label, type, and position are communicated from the metadata to the program via the macro. Parts of the program simply become an exercise in knowing how to use a metadata-aware macro (in this example, it creates macro variables DSLABEL, ATTRIBUTES, and KEEPLIST):

```
%getAttribs(data=ae)
data sdtm.ae(label=("&dsLabel.");
     other statements
     attrib &attributes.;
     keep &keepList.;
run;
proc sort data=sdtm.ae;
     by &sortVars.;
run;
```

The metadata tables can be used throughout the project life cycle. Once the tables are in an easily read data store, they can also be used to create a dataset specification document. It is a relatively easy task to read the dataset and variable-level metadata and neatly display dataset sort order and variable-level attributes and derivations. Parameters to the spec-generating macro can give the user control over variable display order (alphabetical or by position), level of detail, and so on.

Defining Metadata

Many other scenarios could demonstrate the power of systems built upon the metadata foundation. Let's assume that the two described above are sufficiently enticing. Before discussing some of the underlying Truths about metadata, let's first define the term even though its meaning may already seem clear.

The classic definition of metadata is simply "data about data": attributes of operational or analytical data that are relevant to one or more programs that read, write, or update the data. The literal interpretation of this definition is certainly useful, but consider Scenario 1, where we had "data about statistical displays." A broader and more robust definition for our purposes is "data that describes inputs and outputs used throughout the life cycle of a project." This pushes the metadata "door" wide open to include data attributes, characteristics of deliverable items, and the system environment they inhabit. Metadata can include display fragments (as shown in Scenario 1), file locations, switches to control inclusion of variables and datasets in deliverables, and directives for producing program fragments. When one looks beyond the canonical "data about data" definition, it becomes obvious that virtually all features of a system can be abstracted into metadata.

Let's now turn our attention to some of the realities of metadata design, implementation, and cost/benefit metrics.

TRUTH 1: METADATA REDUCES COSTS AND IMPROVES QUALITY

Metadata-based systems produce higher quality deliverables at a lower cost. By moving data and other descriptors from hard-coded entries in programs to programmatically accessible tables in a secure, controlled database environment, tedious and error-prone manual processes are eliminated. Once the tools that access the metadata are validated and programmers are trained in their use, the size of production programs is reduced. Effectively designed programs are sensitive to changes in the metadata they access. Recall Scenario 1's example: when footnote metadata is updated, all that has to be done is rerun the programs using the metadata. The updated footnote text is distributed to all affected programs. Costs are reduced because no time is required for reprogramming. Quality is improved because the possibility of misspelled or misplaced text is eliminated.

It is interesting and instructive to note a side effect of the metadata's evolution. Not even in the happiest of worlds does metadata "just happen." The content and format of metadata tables requires thoughtful examination of an organization's workflow. What hard-coded items can be moved into metadata? Can entire programs be abstracted into one or more metadata tables? How will the metadata be made available to programs? How will the needs of the accessing programs vary? These and many other questions are, of course, essential to successful implementation. Even if analysis of workflow had the unlikely conclusion that no process can benefit from using metadata, *the very act of reviewing workflow is a useful organizational exercise.*

TRUTH 2: METADATA CAN BE USED THROUGHOUT THE LIFE CYCLE OF A PROJECT

One of the strengths of a metadata-based approach to program and system design is the involvement of the metadata throughout the life cycle of the project (a term now being popularized as "end to end" usage). While this may seem to be an academic distinction (single-phase versus broad spectrum of use), it has other, meaningful implications for design. These are perhaps best illustrated with a metadata table describing the variables in a set of deliverables.

Let's consider a highly simplified description of SDTM deliverables:

1. Data Capture. Raw data is created by an EDC system and/or provided by a client.
2. Data in Step 1 is copied or transformed to create SDTM Domain datasets.
3. Datasets in Step 2 are validated to assure conformance to both the programming specification and the SDTM standard.
4. Deliverables (SAS transport datasets and define.xml/pdf) are created.

A bare-bones, variable-level metadata table could contain the following fields:

- Dataset name
- Variable name
- Type, length, label, format

- Source (CRF, derived, other)
- Derivation (programming-level description of derivation)
- Derivation (descriptive)
- List of value mappings and enumerated values
- If a direct copy of a source dataset variable, its Case Report Form (eCRF) page name and number
- Flag for inclusion in dataset

Now let's see how the table can be used throughout the life cycle of the deliverables:

Step 1: Data Capture. EDC vendor metadata can be used to populate eCRF page name and number.

Steps 2 and 3: Dataset programming/validation. Variable attributes can be read by a macro that generates ATTRIB, KEEP statements. A different macro can read the metadata and format it into a PDF that can be used as a programming specification for the dataset.

Step 4: Deliverables. Dataset and variable-level metadata comprise the *de facto* list of datasets and variables that are to be part of the eSub. As such, they can be used to control the transformation of native SAS datasets into the open source SAS Transport File format required for submissions to the FDA. Since the metadata also contains both programming and descriptive definitions, it can be the basis for the datasets' documentation, the all-important "define file" containing dataset and variable-level attributes and definitions.

Notice that we don't have all fields in play at every phase of the process. The descriptive variable definition and CRF locations are used only at the end of the process, during the creation of the eSub documentation. Variable attributes, on the other hand, are used continuously – during dataset creation, while creating the transport files, and as part of the define file. Isolating metadata into tables or datasets by workflow phase (creation, validation, deliverables) is counterproductive. Keeping all information about a variable or any other entity in the same metadata table is the most effective approach.

TRUTH 3: METADATA IS USEFUL ONLY WHEN IT IS ACCOMPANIED BY TOOLS

The workflow improvements described above cannot be attributed solely to metadata. Metadata is not useful *by itself*. Simply having a collection of well-designed, information-rich tables that describe processes, system architecture and the like is not the reason for using metadata. Rather, the objective is to have both the metadata and tools to make access to it nearly transparent. We have already alluded to such tools, among them the ATTRIBUTE statement generator and the programming spec writer.

Consider a program that creates an SDTM domain. One of the steps in the process is identifying a list of variables in the domain. In a metadata environment without tools, the programmer could write a few unchallenging lines of SQL code:

```
proc sql;
     select var into :varList separated by ' '
     from meta.variables
     where domain = 'AE' & select = -1;
quit;
```

This coding is only a little tedious: the programmer must remember metadata variable names, the selection criteria, and the two-level name of the metadata source. It is important to note that this is code that is designed to be stress-free: there is no consideration of error-producing conditions, such as metadata source not allocated; or the WHERE clause returning no observations because the domain value was not found; or there are no variables selected for the domain.

More desirable is a simple tool that creates the list and checks for the unexpected conditions noted above. A call to such a generalized tool might look like:

```
%varList(domain=ae, listVar=aeList)
```

Macro varList creates a list - macro variable aeList – of metadata variables that were selected for domain AE. The macro reduces code volume. If we assume that it follows good programming practices, the macro provides the added benefits of performing error-trapping and messaging that the in-line SQL code neglected. The case for

generalized metadata tools becomes even more compelling as the scope of the task and the complexity of the required tables increases.

TRUTH 4: METADATA IS BEST IMPLEMENTED GRADUALLY BY A DIVERSE TEAM

Two keys to implementation of successful metadata-driven systems are time and teamwork. Implementation should be gradual, especially if metadata usage is foreign to the organization. A small, relevant, well-focused pilot project should address a current workflow issue. A post-mortem of the pilot will reveal pros and cons of the approaches taken, and can serve as a guide for subsequent projects.

The metadata implementation effort should be driven, particularly at its inception, by a team dedicated to the effort. Since a full-blown, mature system reaches "end to end" from data entry to FDA submission, the team should draw from a variety of constituencies. Data management, statisticians, programmers, and regulatory departments are stakeholders in the production of eSub deliverables, and so should be represented when scoping and prioritizing the metadata and tool development effort. Issues of metadata storage and access are relevant throughout development, so IT and data management should be part of the team or at least be available for consultation.

What does all this add up to? Pilot projects. Implementation team from multiple departments. Planning meetings, progress review meetings, and post-mortems. All of these take time and money.

The organizational commitment to allocating these resources requires the most important member, *de facto* or otherwise, of the implementation effort: the advocate. A small, successful proof-of-concept pilot done in someone's spare time may prove compelling enough to warrant formal adoption. But for larger-scale systems whose use will be sanctioned or mandated at a corporate level, it is best to have the blessing (and checkbook) of senior management.

TRUTH 5: METADATA-BASED SYSTEMS REQUIRE AN ONGOING INVESTMENT OF TIME AND RESOURCES

The benefits of metadata-driven processes are significant but not without costs. This section identifies some of the investments that need to be made to support successful implementation. The intent is to simply identify the cost and why it needs to be incurred, *not* to provide hard and fast metrics for cost/benefit analysis.

Design. Sometimes "metadata happens," at least in a raw form not specific to specific organizational needs. Any SAS session or program has access to the dictionary tables and views, which describe datasets, variables, indexes, options, and a host of other features. CDISC provides members with machine-readable versions of standards (ADaM, SDTM, Controlled Terminology, among others). Vendors – SAS/CDI, Medidata/RAVE, Pinnacle 21/OpenCDISC, *et al.* – rely on metadata in a variety of common file formats such as CSV, XLS, and XML.

These sources are often just the starting point for applications that effectively use metadata. They can be used to seed or transfer values to other, custom-designed, home-grown systems. They can also be used in parallel with them (notably the SAS Dictionary Tables' rich dataset and variable-level content).

Time needs to be spent both understanding the metadata that comes out of the box with SAS and with other vendors' products. Likely even more time is required to identify the processes that benefit from abstraction into metadata. Once identified, data design issues. Content and granularity of tables, method of storage, and others – have to be addressed. These are items of concern not only for those directly involved with the subject matter but also database administrators, IT staff, and others with a stake in the creation of quality deliverables.

Research and Development (R & D). Any use of metadata is a journey into new terrain for many organizations. Treating pilot and subsequent projects as an R & D exercise underscores the new, experimental nature of the task and helps stakeholders at all levels – management, statisticians, programmers – to regard the development effort as lessons learned rather than *mistakes*.

Interface. If design is the framing of the metadata "box," then the metadata interface is the tool for filling the box. Just as the design of the metadata tables themselves required time and expense, so does the design of the metadata entry interface.

It is possible, of course, to use the raw, native rows and columns presentation of Excel or similar products. But a host

of security, sharing, and data management issues come with this approach. Problems with data entry also occur, e.g. detecting incorrect values; difficulty in conveying the meaning of some columns' content; and accessing archival and reference data.

The viability of the default interface is also diminished by the hierarchical nature of the metadata. Consider the SDTM VS domain variable VSTESTCD. The variable itself requires attributes and definitions for define files. It also requires additional metadata describing individual values assigned to VSTESTCD (the sensibly-named "value-level" metadata). Rather than requiring the user to remember to manually switch to a different spreadsheet, a well-designed interface will automatically direct the user to the appropriate table or, at minimum, highlight this variable's need for additional metadata.

A good interface requires a skill set that encompasses both technical expertise and a sense of aesthetics. When both skills are employed, the result is a polished, professional tool that encourages usage and preempts errors. Metadata quality increases, and the time and expense of entry drops.

Tool Building. Earlier sections touched on the need for tools to make different forms of access to the metadata as simple as possible. Their output can range from a simple variable list to an XML file transformed by customized XSL. Regardless of the scope or complexity, the tools should be designed and maintained by a dedicated team of developers who are familiar with the metadata architecture. Again, *metadata without tools is nearly useless*. An organization that embraces the idea of using metadata must widen the embrace to include tool building. The cost of that activity cannot be billed to a client and must be absorbed as overhead.

Skill Sets. At the risk of oversimplifying, the Time Past, pre-metadata organization's skill set could be described as "a place to store data, SAS, and a word processor." This would be enough to carry a data point from data entry to CRT/domain datasets into analysis datasets and various tables, graphs, and listings. Documentation (the "define file") was a manual process, as was annotation of case report forms.

Time Present is far more complex. Data is stored in a variety of formats and can be validated using a variety of open-source and proprietary tools. The XML documentation file is itself data, and must be constructed according to a strict set of rules and transformed via XSL to be made readable. Some of the technologies in play here are XML, XSL, JavaScript, database forms (controlled by Visual Basic), HTML, and CSS. A knowledge of PDF, PostScript, and RTF internals is also helpful. Clearly, in order to move in the current, metadata-based world, an organization has to be multilingual. This fluency requires time and resources to acquire the skills and keep them current.

Training. The demands of the expanded skill set described in the previous section also require a commitment to allocating time and other resources to training tool developers. Resources also need to be allocated for training the user community: those who use the metadata interface need to know how to most effectively use it; programmers must be taught how to take advantage of time-saving and quality-improving metadata access tools. The training should be required for new hires and should be offered periodically so that all users can be aware of upgrades and new tools.

Redesign. Recall the inclusion of "ongoing" in this section's title. Once the design – of the metadata, interface, and tools – is stable and implemented, change is both *inevitable* and *desirable*: users suggest changes to interface functionality; programmers request enhancements to existing tools or have ideas for entirely new ones; conforming to new standards or client requests requires changes that range from localized in a single program to system-wide. Even in a static, unchanging environment, the simple growth in the size of the metadata tables may warrant redesign. In the context of the current section, expenses attributed to metadata-based systems should be viewed not as front-loaded and one-time, but rather as ongoing and worthwhile, since this means that the metadata systems are being used and their scope is expanding.

TRUTH 6: METADATA REQUIRES ADAPTABLE ARCHITECTURE

It is tempting to limit design of both metadata and tools to address the immediate requirements of a project. However, as we noted earlier, one of the hallmarks of a successful system is its ability to gracefully handle increased scope. This usually requires modifications to existing tables or creation of entirely new ones, along with parallel changes to existing and new tools.

What are some of the sources of change? It may be prompted by internal, in-house requests which require new tables and/or modifications to tools. Other sources can be external. An new EDC system may store its metadata in ODM-compliant XML. A client may want to transmit data and descriptive metadata (recall "data about data"), thus requiring bridging technology from their systems. Finally, an external standard may be adapted or modified. The adoption by the FDA in the mid 2000's of SDTM IG 3.1.1 and later upgrading to IG 3.1.2 demonstrates the need to be responsive to external requirements.

Just *how* metadata and tools can be designed for this essential flexibility is beyond the scope of this paper. But some general considerations and types of tools are:

- Tools (e.g., SAS macros) should be highly parameterized. Interface selection lists should be populated by tables rather than hard-coded as a form property.
- Build migration and bridging tools that facilitate conversion of data formats and connect the metadata "silos" in the corporate data "farm" (an example of the latter is using Medidata/RAVE ODM-compliant XML to seed CRF values in SDTM variable-level metadata).
- Move hard-coded data sources into metadata. Rather than code a data source in a program, move it to a metadata table. Remember, metadata doesn't simply describe data attributes; it can also describe data locations
- Interface forms and data-handling code should be separated.

Sidebar: What *Not* To Do. These ideas should be credible on their own merit. But to reinforce their worth, consider this cautionary tale. An early (c. 2003) version of our analysis metadata was stored in Microsoft Access databases, one per study per project. This was an adequate solution, since the number of studies using the system was small. However, once usage of the metadata was mandated for *all* projects, not just eSubs, the number of MDBs increased dramatically, as did maintenance headaches when adding fields or improving interface functionality. As a result, in 2008, we migrated all the data stored in MDBs to Oracle. We continue to use Access forms for the interface. Now, when the user enters the interface, the forms code checks for and automatically deploys updates. Front-end maintenance went from being a time-consuming chore to being transparent.

TRUTH 7: METADATA SHOULD BE TREATED AS A VALUED CORPORATE RESOURCE

Let's exit the metadata world for a moment and consider some systems common to all organizations. Without reliable storage and access to financial data, an organization doesn't have the ability to pay for the present and plan for the future. Without reliable human resource systems, the organization cannot track its most valuable assets and can run afoul of regulatory agencies. Even if tracking finances and people is not the organization's mission, it is potentially ruinous if attention to these systems lapses.

Now reenter the metadata environment we have created in this paper. We have described the investment of resources needed to for metadata-driven projects.. Given this expense, and recognizing metadata's importance in driving of the submission process, it follows that metadata should be treated as a mission-critical resource. This has implications for both metadata and the tools that provide access to it.

Metadata. The data-handling ideas here are no different than those for any other data resource. Access to the metadata needs to be controlled by directory permissions, via the user interface, or by the database itself. An audit trail needs to be maintained, providing a reliable mechanism for identifying when changes were made to a data value. Backups need to be taken on a regularly scheduled basis.

Some of the items described above are built into systems such as Medidata/Rave and SAS/CDI. Users of home-grown systems, particularly those storing metadata in spreadsheets, are hard-pressed to duplicate this functionality. Particularly problematic are enforcement of access restrictions, maintaining audit trails, and building repositories. The metadata resource is best housed in a database environment.

Tools. The tools that provide users with transparent access to the metadata are no less important than the metadata itself. Two practices – one familiar to clinical programmers, the other possibly not – come into play here.

Users of any program providing metadata access can regard the code (SAS macro, Oracle or Access form, etc.) as a "black box" that expects certain input (parameter value for a macro, metadata value entered in the user interface) and creates output (ATTRIBUTE statement, metadata value) using documented rules. The user does not have to be concerned with the operations performed by the tool because the tool has been validated. Validation is an essential

process, giving internal users, clients, and external auditors the knowledge that a tool performs to spec, and does so reliably.

Once you start building general-purpose metadata tools, you are no longer a programmer with some good ideas. You are someone who is writing code that is used not just by you but possibly the entire organization. This increased exposure and clientele necessitates use of some tools familiar to software engineers. Version and source control become important. Some tools may require formal requirements specification, testing and other tasks that are part of the traditional software development life cycle.

TRUTH 8: METADATA WILL BECOME ESSENTIAL, NOT JUST "NICE TO HAVE" OVER TIME

If you aren't convinced by now that metadata-based systems offer the prospect of improved quality at what are ultimately reduced costs, consider this: even if *your* organization elects not to "think metadata," it is almost inevitable that you will do business with an organization that does. The external push for metadata comes from several sources:

- **Standards.** Industry standards such as CDISC's ADaM, SDTM, and Controlled Terminology are commonly stored as Excel or CSV files. These files provide both a description of the standards and the opportunity to seed in-house systems with a "Gold Standard" copy of dataset and variable descriptors.
- **EDC.** Metadata in the form of ODM-compliant XML is also becoming the *lingua franca* of EDC systems. Confining use of the XML to an upstream process such as data entry squanders an opportunity to realize the benefits of this programmatically accessible resource throughout the study's life cycle.
- **Clients.** Data transfers between the CRO and its clients are more frequently accompanied by program-accessible metadata. Even if the CRO is not required to *deliver* metadata in SAS, XML, or Excel files, it must be prepared to *receive* this information from a client and, ideally, have tools in place to load the files into in-house or other, proprietary systems.

Finally, even if an organization existed in a standard-less vacuum with adequate in-house systems, metadata will ultimately be utilized to handle an increasing volume of work. As noted above, there comes a point where upgrading computers and adding staff has marginally positive or even negative benefits. When management scrutinizes workflow and processes, it is likely that metadata will be at least part of the organizational correction. Better to initiate a pilot project now, when the waters are rippling, than later, when a quality-compromising deluge is upon you.

CLOSING COMMENTS

An organization that hopes to be competitive in today's clinical trials environment must pursue, rather than shirk from, cultural and paradigm shifts. We have seen in this paper that implementation of metadata-based systems is one such paradigm shift. Processes can be automated and re-tooled, and high-quality deliverables can be produced more quickly with well-designed, adaptable systems. It is essential to realize that the benefits that accrue to metadata usage are the result of the continual investment of time and other resources. Without a clear understanding of the need for solid metadata design, for access tools, for skill set expansion and the other features discussed in this paper, the implementation effort will be ineffective at best. Clear-eyed recognition of and commitment to the complexity of the implementation effort is the first step down the path to long-term success.

CONTACT INFORMATION

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

Frank@CodeCraftersInc.com

ACKNOWLEDGEMENTS

As always, thanks to **April Sansom**, copy editor extraordinaire. This paper was the most “challenging” (her word) in recent memory. I appreciate the “tenacity” (my word) she displayed while trying to convert my sometimes vaguely coherent ideas into readable prose. Thanks as well to **Russ Lavery** for comments that brought both sharpened focus and better grammar to this paper.