Paper RS09

Managing The Change And Growth Of A Metadata-Based System

Jeff Abolafia, Rho, Inc., Durham NC
Frank DiIorio, CodeCrafters, Inc., Chapel Hill NC

Abstract

Regulations governing the pharmaceutical industry require extensive metadata to support datasets and displays when submitting a New Drug Application (NDA) to the US Food and Drug Administration (FDA). The complexity of the submissions process increased markedly with the relatively recent introduction of standards from CDISC and other non-governmental organizations. In two previous papers we outlined the metadata architecture needed to support a large and complex project as well as how the metadata is used throughout the life cycle of a project. We also demonstrated how well-designed and easily-accessible metadata improves work flow, efficiency, and project management throughout the life cycle of a project.

This paper builds on the previous two papers and discusses lessons we have learned as the scope, user base, and complexity of our metadata systems have increased. Topics include: metadata design; user interface; developing tools to access the metadata; repurposing metadata throughout the life cycle of a project; and software development and maintenance issues. These topics should be of interest to those who are just starting to develop metadata-based systems as well as those who have implemented such systems and are being challenged and perhaps a bit beleaguered by their maintenance.

Organization

A simple approach to preaching the gospel of metadata would be to present a list of do's and don'ts based on our experience. This is easy to write, readily rendered in PowerPoint, and reasonably valuable to programmers and analysts interested in metadata-driven applications. This method of presentation is inadequate, in our view, because it doesn't adequately describe *how* we arrived at the lists. The story of why our current, effective toolset looks the way it does is replete with success stories and a "not small" number of miscues. A discussion of the rationale for our decisions during the system's evolution is far more valuable than a simple set of guidelines.

Thus we structure the discussion as follows. We divide the evolution of the system into five stages. For each stage, we describe the internal and external challenges that needed to be addressed. We also identify the metadata that was developed or enhanced, tools that we developed to access the metadata, solutions that worked, solutions that failed, and other aspects of development that had both expected and unanticipated repercussions (we never would have imagined in 2001, for example, that XSL and VBA would be cornerstone technologies for some of the subsystems).

We place emphasis on key lessons learned at each stage, and deliberately avoid detailed discussions of how the solutions were coded. Even though the programming was challenging and our solutions often had a significant "gee whiz" factor, we want to focus here on how issues emerged and were resolved during each phase, and what we did to address them. This, ultimately, is more valuable at an organizational level than discussing the in's and out's of, say, creating XSL and Javascript to sort a table in an HTML file, cool though that may be.

Background

In its early years, Rho was a contract research organization (CRO) working on relatively small pharmaceutical data management and statistics projects. A project usually consisted of: data management activities; creating an analysis plan; creating analysis datasets; and producing dataset listings, and statistical tables and figures. We usually had no more than 25 simultaneous projects, none of them large submissions.

During this time, each project was configured on an individual basis. The set up was influenced by the preferences of the project's statisticians and programming team. Specifications for the creation of analysis datasets (source or derivation of variables, variable attributes, etc.) were stored in Microsoft Word™ documents. This meant that the "data about the data," the project metadata, was not stored in a programmatically accessible format. *That*, in turn, meant that it could not be re-purposed (used by different programs for different purposes throughout the project life cycle). Descriptions of the content and layout of graphic and text-based displays were also entered in Word. Here again, the Word documents were comfortable and familiar tools for the biostatisticians writing the specifications, but the document format proved to be a limiting environment, for reasons we will describe later.

The Word-based work flow meant that variable names, labels, and other attributes were either manually copied from the specification document into a program or entered manually. Changes made to the specifications had to be identified and made in all of the associated programs. If, for example, a footnote change affected ten programs, the change had to be manually made to all ten programs. Obviously this was not a very efficient process and not what we wanted programmers and statisticians spending time on (and it was *certainly* not what programmers and statisticians enjoyed spending time on, either). While this was not the most time or cost-effective process, it worked for managing a relatively small number of single study projects.

CRO versus Pharma. It is worth noting here the distinction between Rho's status as a CRO versus a pharmaceutical company. A pharmaceutical company is relatively self-contained and can implement fairly small range of methods for handling work flow. The CRO, by contrast, provides services to many clients. This means that it must be prepared to enter data in-house, accept data from the pharma company or vendors, sometimes in multiple formats. Throughout the project life cycle, the CRO must be responsive to different pharma companies' ways of handling data validation, dataset and display specifications, and even what is sent as part of the submission package to the FDA. The greater the number of clients, the greater the range of dealing with even the most basic tasks. Even though our goal is always the same – timely delivery of a quality product – there are often pronounced differences in how we arrive there. Metadata-based systems are a critical technology that allows Rho to have the requisite flexibility to compete in the CRO marketplace.

Moving Out of "Stage 0". About six years ago, we began an NDA submission project involving almost thirty studies. For each study, we were required to produce a database consisting of the raw (collected) data, a database containing derived or analysis datasets based on the raw data, documentation ("define" files in FDA parlance) describing the distinct databases, data displays and associated documentation, and a clinical study report. We also had to produce an integrated database of data from all thirty studies and data displays to support the ISS and ISE. In total, this project required us to produce about thirty study-level databases, an integrated database, over one thousand displays, and documentation in the FDA-prescribed format.

Given the complexity and sheer volume of this project, we realized that our current processes and tools were likely not going to work. We needed standardization across projects, the ability to handle constantly changing specifications, the ability to easily modify variable attributes to conform to FDA regulations, a mechanism to easily drop or reformat variables, and the ability to repurpose metadata throughout the life cycle of the project. In other words, we sensed the need for technology that would allow us to handle the demands of a high volume of new and differently-formatted deliverables.

Stage 1

Our initial metadata system was born out of the need described above. Since this was a tool to be used primarily by SAS™ programmers and statisticians and to be utilized by SAS programs to create SAS datasets and data displays, our initial metadata was stored as a collection of SAS datasets. Metadata was entered and viewed using the native SAS Viewtable window. The SAS/Share server allowed multiple users to edit simultaneously. The metadata consisted of four datasets: STUDIES, DATASETS, VARIABLES, and TABLES.

The STUDIES dataset had a structure of one record per study and contained data about each study that:
- described the design of each study
- described the phase of each study
- identified the network location of each study
- provided the location of the components of each study (i.e. analysis datasets)

DATASETS held dataset-level metadata, one record per dataset in the study. It contained fields that:
- describe the contents and structure of the dataset
- discuss how the dataset was created (essentially background material useful for programmers)
- list variables that uniquely identify an observation in the dataset
- filter a dataset, identifying whether it should be included in particular types of output; a dataset may be part of an Integrated Safety Summary (ISS) database but not an Integrated Safety and Efficacy (ISE) database, for example
- text fields to enter comments about the dataset and any known quirks in the input dataset

The VARIABLES metadata is the companion to DATASETS. For every dataset in the DATASETS metadata, VARIABLES contained:
- descriptors such as type, length, format, and label
- if “raw” data, the source page number in the Case Report Form (the principal, paper-based data collection instrument)
- if derived data, a narrative of how the variable is created
- controlled terms or codes
- the desired variable order when writing datasets for FDA submission
- filter variables that facilitate selection of variables for different types of output

The TABLES metadata group describes key features of each display. The TABLES dataset contained:
- display type (Table, Figure, Listing)
- display number
- titles
- datasets used by the table
- location of the display and display program

These four metadata datasets were a good start to creating a metadata system and gave us some much needed new capabilities. At the study level we were able to programmatically:

- allocate dataset libraries
- find specific study directories
- select the studies needed for a particular analysis (i.e. all Phase I studies)
- determine what stage a study was at (i.e. ongoing or final data received)

At the dataset set level we gained the capability to:

- Provide information to both programmers and the FDA from a single source (such as a description of the dataset, the structure of a dataset, and the key fields for a dataset)
- Programmatically select the datasets required for a specific database

The VARIABLES metadata had the potential to provide the most functionality:

- Provide information about each variable to both programmers and the FDA from a single source. This dataset was used to provide specifications to SAS programmers and to generate the all-important "define file" required by the FDA.
- Programmatically select the variables required for a specific dataset
- Programmatically derive variable attributes from the dataset specifications. This not only saved a tremendous amount of valuable manual labor, but also allowed us to easily deal with constantly changing specification
- Programmatically allow us to rename, re-label, recode, and remove user-defined formats from variables to meet NDA requirements

At this stage, the metadata system was primarily used for datasets and variables and was only used for the single NDA submission project. Program-driven use of the metadata was limited: we generated code fragments to rename, relabel, and reformat variables to meet regulatory standards. Once datasets were finalized, the metadata was used to generate the transport datasets and "define files" required for the submission. Specifications for creating analysis and raw datasets in other, single-study projects were still being manually copied from Word to SAS datasets.

Stage 1 – "Trial (and Error) by Fire" – Summary

User Base	*Skill Set*	*Interface*	*Key Applications and Tools*	*Metadata*
✓ Statisticians ✓ Programmers	✓ SAS	✓ SAS viewer	Macros to: ✓ rename, relabel, reformat variables ✓ filter datasets, variables ✓ create transport datasets and documentation	✓ studies ✓ datasets ✓ variables ✓ displays

Lessons

✓ Moving specifications from Word to machine-readable format improves work flow even with a limited number of metadata access tools

✓ The tools that were developed, even with their study-specific coding, were effective because they were able to exploit the similarity across the various studies of dataset and variable names as well as directory structures.

Stage 2

After the completion of the NDA project, the metadata system lay dormant for about a year and was not used for other, smaller projects. When we were awarded the contract for a second NDA project we decided to evaluate the metadata, tools, and work processes used during the our first project. In addition to revising NDA-specific and generalized tools in our library, we conducted focus groups with programmers and statisticians. These meetings helped us determine how we could improve the system and understand why it was not being considered for use in the more typical single-study projects. This led to the improved Stage 2 system. We addressed problems with the existing system, specifically: the metadata entry user interface; the lack of metadata tools; and improving the content of some of the metadata datasets.

The Importance of the Interface. The most significant lesson from the focus groups was to pay close attention to the metadata entry interface. Statisticians, who were used to entering specifications in Word, disliked the SAS interface. It was clunky and had none of the editing capabilities of Word. Our first Stage 2 task was to experiment with alternative interfaces. Microsoft Excel™ seemed quite popular among other companies in the pharmaceutical industry, and so warranted investigation.

Excel provided users with many of the editing tools that they had become accustomed to using Word. Users preferred the Excel interface to SAS, since data entry was faster and easier and the Microsoft Office paradigm was familiar. Additionally, SAS provided tools for easily reading Excel tables, thus keeping the specification document in a programmatically accessible format. While there were several significant advantages of using Excel, there were also some noteworthy shortcomings, most notably an inability to allow multiple users to edit a spreadsheet. We also found that in some instances, SAS produced errors while reading Excel tables. Finally, Excel did not contain any of the security features of a true database.

Next we decided to test Microsoft Access™. It had many of the features that we liked about Excel plus many additional capabilities. Like Excel, Access provided a user-friendly interface, many of the familiar editing tools of Word and Excel, and transparent access of tables and queries by SAS. Access also offered:

- a multi-user interface
- a true relational database
- customized forms for data entry
- the ability to create a Web front end interface
- remote access capability
- the ability to use Visual Basic for Applications (VBA) to perform rudimentary error checking
- the ability to auto-populate fields
- improved data integrity
- an audit trail which could log changes to the metadata

At this stage we only utilized the basic features of Access. We created a simple form for each metadata table with a handful of fields containing drop down boxes and checkboxes.

The Need for Tools. The second lesson we learned from our focus group sessions was the importance of metadata tools. Our users pointed out that for small and medium sized projects you would not gain efficiency – indeed, you would likely become *less* productive – if there were no applications to make access to the metadata more or less transparent.

To underscore the need for access tools, consider the programming requirements needed for accessing metadata that describes a display such as a statistical table. The table number must be located in the DISPLAY metadata. We must also gather and correctly sequence the footnotes from the FOOTNOTES table. Finally, we retrieve standard headers and footers from the GLOBAL table. Once all the pieces are identified, they need to be presented to the table-writing program in an agreed-upon format (macro variables, datasets, etc.). To be thorough, we should add checks to ensure data quality – are all the footnotes in the DISPLAY table actually present in the FOOTNOTES table? Do we have complete title text? And so on.

We could, of course, write code in each table program to perform these actions. More likely, we would want a tool that would do the work for us, reading the required tables, and creating a set of macro variables that would make the metadata readily accessible. The process to create macro variable for Table 10.1 should be as simple as:

```
  %getSpecs(type=table, id=10.1)
```

The macro would perform all the activities described above, and would produce diagnostics that would quickly give the table programmer an indication of success or failure. The SAS Log would contain messages showing what, exactly, was successfully retrieved by `%getSpecs` and what was cautionary or problematic.

Clearly, a library of tools for display generation, LIBNAME assignments, option setting, and the like were needed. So, in this stage we began to develop general purpose utilities to interact with the metadata. This included tools to:

- obtain a list of all datasets for a project
- obtain a list of variables to include in a given database
- rename and re-label variables based on the metadata
- drop variables to be excluded from a given database
- produce dataset specifications in a user-friendly format
- check the metadata and compare the metadata to the data
- access the metadata needed for a given display

Richer Metadata. The third lesson related to the metadata tables. Users asked for additional fields and the refinement of existing fields. So, new fields were added to each of our metadata sets and the interface was improved for several existing fields. During this stage, the TABLES metadata dataset underwent the largest overhaul. Fields were added to this dataset to automate the programming of data displays. These new fields combined with newly developed metadata tools allowed programmers to easily use titles, footnotes, population variables, subsetting statements, and stratification variables from the metadata. This meant that changes to display specifications could be made by research assistants (or non-programmers) in a single location, instead of programmers having to identify changes in the specifications, find all programs that were affected, and make modifications to each of these programs.

The changes we made during Stage 2 led to wider usage of the metadata system. While use of our metadata system was not required, over time more small to medium-sized projects began to use metadata and realize the efficiencies gained by its use.

Stage 2 – "Greater Usage, Improved Functionality" – Summary of Changes and Enhancements

User Base	***Skill Set***	***Interface***	***Key Applications and Tools***	***Metadata***
✓ Statisticians ✓ Programmers ✓ Research assistants	✓ SAS ✓ Excel ✓ Access	✓ End Stage 2 using Access	Macros to: ✓ retrieve dataset and variable attributes ✓ perform metadata quality control ✓ automate creation of data displays ✓ create specification documents	✓ studies ✓ datasets ✓ variables ✓ tables ✓ footnotes

Lessons

- ✓ User feedback can improve quality of the metadata entry interface, the contents of the metadata, and the tools used to access the metadata
- ✓ Develop tools to make the metadata easily accessible to statisticians and programmers
- ✓ Specifications for data displays can be effectively produced with metadata and metadata applications
- ✓ Datasets and displays can be created by research assistants rather than programmers if adequate documentation and toolsets are in place.

Stage 3

The efficiencies gained by the earlier metadata systems led to usage on a company-wide level several years ago. Use of the metadata entry system become mandatory. Also, study setup routines were made more generalized (for single as well as multiple-study projects) and the directory structure that they created conformed to our newly-adopted corporate standards. Thus Stage 3 is characterized by standardization.

Implications of Standardization. Standardization begets uniformity, making it easier for programmers and statisticians to move from project to project, a not-trivial consideration in the "fluid" CRO environment. LIBNAMEs have identical meanings, directory structures are similar, and the familiar metadata user interface all send a message that while the *project* may be new to the statistician or programmer, the *look and feel* is not.

Standardization also meant a larger user constituency and a wider range of user requests that had to be satisfied. One of the key features that had to be implemented was the creation of user-friendly Access forms to make metadata entry easier. Based on statistician feedback, we improved the user interface's screen navigation, added more sophisticated error trapping, and provided fields useful on both project-specific and corporate levels.

As we learned in Stage 2, metadata without transparent end user access is at least one step backwards in productivity. As the metadata contents stabilized, we continued developing easily-used tools to provide access to statisticians and programmers.

Tool Building. An attendant part of the spec writing process was creating a spec document for the programmers. A macro, `%printSpecs` – was developed to create a PDF or RTF that clearly displayed the variables that needed to be created for a dataset. Color-coding and other text rendering techniques were used to highlight important items. Recalling our experience from earlier projects (most notably, "live by the metadata, die by the metadata"), we developed a macro to examine the metadata for consistency. Among other tests, it examines the metadata for:

- like-named variables with different attributes (e.g., variable `SUBJID` has length 12 in dataset AE and length 15 in dataset `CM`)
- variables with attributes that are not suitable for FDA submissions (i.e., those that are not acceptable in SAS transport files)
- other conditions which could not be readily programmed in the Access user interface

We also wrote several programmer-oriented macros, among them `%ATTRIB`. It reads metadata for a dataset and creates fragments of ATTRIB, KEEP, and LENGTH statements. This relieves the programmer from having to manually create labels, assign formats, and other variable-attribute related tasks. This is especially helpful when attributes, or even the inclusion of the variable in the dataset, can change frequently. With the metadata-driven approach, all that has to change is the metadata and the `%ATTRIB` macro calls remain the same. That is, changes are made once, "upstream" in the work flow, rather than manually and possibly many times further "downstream" in the process.

Generalized Tools. Several other factors influenced tool development during this time. Unlike the earlier, deadline-driven NDA projects, we now had the luxury of at least a little bit of time. We removed project-specific coding from the macros and screens, and ended up with a suite of generalized tools, written without any study-specific references. The transition from coding for a specific study to all possible studies presented a number of programming and design challenges. The benefit of generalization is that the macros, over time, became written in a consistent and bullet-proofed manner, following a set of internally-developed Best Practices.

An outgrowth of the generalized metadata tools was the parallel growth of a library of low-level macros appropriate for *both* metadata and other applications. Some of the tasks addressed by these macros were:

- count the number of observations in a SAS dataset or Access table

- create macro variables containing the count and distinct values of a variable in a dataset
- enclose individual tokens in a macro variable in quotes, optionally converting them to upper case
- build datasets containing variable and dataset attributes in a library
- given two macro variables containing blank-delimited lists, create counts and macro variables for tokens appearing in one or both lists

The macros performed extensive error-trapping, so if there are problems at the start of or during execution, they and, in turn, their calling program, could fail gracefully and clearly inform the user of the reasons for failure. Aside from their impact on metadata tool development – more robust applications developed faster with less coding – it is interesting to note something that we did not anticipate. This period saw the development and refinement of a great number of generalized, low-level tools, while other stages did not. That is, the focus of our development efforts changed over time, driven by internal and external (client, FDA) needs.

Training. As the number of tools grew, so did the need for tool documentation and training. We developed presentations and documentation that described metadata entry and use of the `%attrib` and other macros. There was also a growing need for internal documentation that described how the different subsystems (project initialization, program setup, and others) were related. This is an ongoing process whose final form is still undecided.

Variety of Tools. The scope of activity described above hints at one of the other key lessons we learned during this exciting and challenging time: "it's just not SAS any more." In order to provide the range of output that both internal and external clients required, we had to become newly or more deeply conversant with new tools and techniques. Among these were: a better knowledge of Access forms and database design, VBA, and some familiarity with PDF and RTF internals (the latter because some clients wanted FDA deliverables in this format). A skill that is less tangible but is essential to success is a good aesthetic sense. If the metadata entry screens were badly designed and cryptically worded, their acceptance would have been slower.

Stage 3 – "Standardization" – Summary of Changes and Enhancements

User Base	***Skill Set***	***Interface***	***Key Applications and Tools***	***Metadata***
✓ Statisticians ✓ Programmers	✓ SAS ✓ Access ✓ VBA ✓ RTF, PDF	✓ Improved Access forms ✓ Online help	Macros to: ✓ validate metadata ✓ create specification document ✓ create SOP-compliant directory structure ✓ create label, other statements directly from metadata	✓ datasets ✓ variables ✓ project configuration

Lessons

- ✓ Standardized directory structure, naming conventions improves statistician, programmer productivity
- ✓ Continually enhance tool set, applications based on user feedback and R&D team members' usage of tools when they were needed for project work
- ✓ Application functionality and effectiveness is greatly enhanced when other languages/technologies are used. Even staying within SAS "box," still need to have some knowledge of other file formats' structure.

Stage 4

We felt that the generalized, study-independent code written during Stage 3 benefited a large portion of the internal user community: project initialization became routine, and was usually delegated to administrative aides; metadata entry and display was performed using stable tools; and a variety of ad hoc tools was developed to meet some of the more unique and unanticipated needs of both internal and external clients.

SDTM: New External Standards. This standardization and responsiveness was tested when we began dealing with a new set of requirements, this time coming from an external source. During 2006 we began receiving requests from clients to produce data and documentation compliant with the Clinical Data Standards Interchange Consortium (CDISC) Study Dataset Tabulation Model (SDTM) specification. Prior to this, our main compliance focus was on FDA submissions, which emphasized formatting and documentation of datasets and left dataset and variable naming and content to the discretion of the applicant.

The SDTM introduced a new layer of complexity, namely the control of dataset structure and variable content. Prior to the introduction of these standards, a variable representing a subject's date of birth might be called `BDAY,` stored as a SAS date variable with a label of "Date of birth (YY/MM/DD)." By contrast, the SDTM model *required* variable `BRTHDTC,` a label of "Date/time of birth of the subject" and a value stored in a format compliant with the ISO 8601 standard. The preferred format for documentation was XML formatted according to a variety of schemas and transformed by XSL. These and a host of other requirements had an impact that was felt from data capture throughout the system to final deliverables.

Fortunately, these requirements were introduced into our work place at a time when it had become metadata-centric. Also, the SDTM specifications were partially implemented in an electronic format that was easily converted to our metadata structure. To meet the demands of the SDTM standard, we revamped some applications and developed entirely new ones as needed. The metadata and its Access-based interface were largely rewritten for several reasons:

- SDTM required more tables than our earlier models
- SDTM had metadata fields that could not be changed for some datasets
- Some variables were required by the standard, and so could not have the "Include this variable" box unchecked
- The extra metadata and the need to join tables in the interface introduced the need for new VBA code and Windows API calls.

Given our experience with the evolution of the non-SDTM interface, and with knowledge of the SDTM implementation guidelines in hand, we were able to write the first version of the SDTM interface with a considerable amount of error-trapping.

Other familiar tools were rewritten and modified, using the older tools as a starting point. The spec-printing macro was modified for the new fields and tables. We also developed metadata QC tools and modified the `%ATTRIB` macro to process the SDTM metadata.

New Client Deliverables. One of the unanticipated side effects of the new standard was that clients unfamiliar with the CDISC requirements required more documentation of our processes. In order to keep the client informed about the various mappings, transpositions and other transformations required by the SDTM model, we began delivering `%printSpec` output. In other words, what began as an internal tool for the biostatisticians and programmers now became a deliverable to the client. A variety of clients leads, in turn, to a variety of ways the clients wanted to see the metadata. Thus the spec-printing program had options added on that controlled the file format (PDF or RTF), color-coding, and other features. Everything, it seemed, was in play and in flux: the format of the data, the way in which we documented it for the FDA, and even the internal (now also external) presentation of the metadata.

Changes to Metadata Content. During this time, the analysis metadata tools continued to be enhanced. These are, after all, the heart of the FDA submission package, the basis for the statistical tables and figures from which safety and efficacy will be determined. Based on feedback from biostatisticians and programmers, we continued to enhance the metadata and the user interface to the metadata. One of the key improvements was a variable definition repository. This let the user say, in effect, "I'm defining the Adverse Event table for study 104 in this project. It looks a lot like study 102. I can use the repository to copy the metadata from 102 into 104, then tweak the definitions for the particular needs of the study."

This has the potential to save enormous amounts of biostatistician data entry time. Other metadata enhancements also meant that we began to be bogged down with legacy applications: the common spec printing and validation tools had to work with the older systems as well as the newer, feature-rich ones. We realized only after the fact that we had become software developers without a true appreciation of the need for versioning and change control.

Part of this was solved by upgrading older applications. Another approach was to break up some of the larger applications, inserting a process that determined which version of the metadata was being used and converting the metadata temporarily and on the fly into a standard format for use by the error-checking, transport file creating, or documentation tool.

Expanding Skill Set. We also found that SDTM implementation brought the benefit (to the FDA) of standardization, but among the many costs was a dramatic broadening of the skill set required to support it. As in the previous stage, not only was it not SAS any more, some parts were like nothing we had ever seen before. Creating the documentation file – define.XML – so that it worked reliably and predictably with FireFox and Internet Explorer required a knowledge not only of XML but also of XML Schema and, most bizarre of all, XSL, the tool by which the raw XML is transformed into the HTML-like screen displaying the metadata. Making the viewing experience effective and user-friendly also required a working knowledge of HTML and JavaScript. A calm demeanor and a dark sense of humor were assets during the early stages of development, when many of disparate technologies did not "play nice" together.

The SDTM usage of ISO 8601 dates, times, date-times, and durations required development of a variety of macros. We had to be able to convert SAS date, time, and date-time variables into the ISO 8601 extended date-time format. Likewise, event durations had to be stored according to the standard. While these values are compliant with the needs of the SDTM specification, they also render previously straightforward values such as

```
17501
```

into readable, but hard to manipulate text such as

```
2008-05-12
```

A duration that could be measured as the number of days between two events becomes:

```
P11M12D
```

Fortunately, the SDTM specification allows the creation of supplemental data sets that can contain variables with more computer and algorithm-friendly values.

The road to SDTM compliance became a bit bumpier last year, with the release of a new version of the standard. The majority of changes were easily handled by adding parameters to macros, but identifying what needed to change was tedious. Even something as prosaic as our ISO conversion routine had to be changed to accommodate dates with "reduced accuracy." In earlier versions, a known year, unknown month, and known day had to follow the principle of "appropriate right truncation." For example, using the initial SDTM standard if only values for year and day were known, a value would look like:

```
2007
```

Once a date component was missing, other components to its right could not be displayed. In the new SDTM version, however, the value could be represented as:

```
2007---14
```

Accordingly, a parameter had to be added to the ISO conversion macro, and the macro had to be revalidated.

The movement to the CDISC SDTM standard has gained so much momentum that its use may soon become mandatory for FDA submissions. This has a number of implications for a CRO, which by its very nature requires that it serve multiple "masters." First, we must develop the necessary tools for SDTM compliance and we must train a variety of constituencies: biostatisticians, programmers, sales and marketing representatives, and clients. Second, we must continue to support projects using older systems and provide the means to make the transition to SDTM should the client desire to do so. The variety of SDTM standards, use of eCTD or more traditional directory structures, and other choices requires flexibility throughout the project life cycle. The process has to be configurable throughout the cycle, and much of this flexibility is provided by both enhanced metadata content and new and improved tools.

Stage 4 – "SDTM: Complying with New Standards" – Summary of Changes and Enhancements

User Base	*Skill Set*	*Interface*	*Key Applications and Tools*	*Metadata*
✓ Statisticians ✓ Programmers ✓ Clients (for "informal" deliverables)	✓ SAS ✓ Access ✓ Windows API ✓ XML, XSL ✓ Standards (SDTM, ISO)	✓ Repository ✓ Audit trail ✓ Improved navigation ✓ Upgrade legacy MDBs	Macros to: ✓ create SDTM spec document ✓ validate SDTM domains ✓ manipulate ISO8601 values ✓ document macro library	Audit trail for: ✓ datasets and variables ✓ SDTM (domains, variables, value-level) ✓ upgrade legacy metadata

Lessons

- ✓ Compliance with new standards is made easier with pre-existing metadata experience, database design, and toolset
- ✓ New standards creates opportunities for new client deliverables
- ✓ Anticipate versioning of metadata and standards when developing applications and tools

Stage 5

In seven years the use of metadata at Rho has grown from a good idea implemented for a single multiple-study project to a corporate standard implemented in over 150 projects with a user population of over 100 biostatisticians and programmers. One of the key challenges facing us now is keeping the user base happy and growing, while at the same time trying to keep things manageable with respect to version control, new tool rollout, and the like.

Metadata Storage. Currently, each project has its own Access database for analysis and SDTM metadata. Thus there are hundreds of MDB files scattered throughout the network. Since they are files in the Windows XP file system, they are visible to any one with sufficient security permissions and thus can be deleted, accidentally moved, or renamed. While these events are rare and recoverable, they do lead us with a bit of urgency to the next logical step – storing all projects in a single, industrial-strength Oracle™ database.

The advantages of this move are compelling:

- the metadata would be secure, not subject to the vulnerabilities of traditional file-based data storage
- a properly-designed Oracle database's performance can be significantly better than Access, especially when processing large amounts of archival, repository data
- the tables can be viewed and updated with Access forms, which means the current interface would not have to change
- Since it puts us in a true database environment, we would not be able to respond quickly to a user request for new fields or tables. While something of a negative, what this effectively forces us to do is bundle changes, make a set of database modifications, and deploy a new version of the system. This would bring a welcome end to the incremental and hard-to-track changes that characterize the current system.

The move to Oracle promises to be a large, complex undertaking. Nevertheless, we still need to be aware of potential re-use and reinterpretation of our existing metadata. Two ongoing activities illustrate this.

Legacy Metadata. The first of these is conversion of legacy metadata. New projects and studies are being deployed with the new analysis and SDTM user interfaces. These are, in effect, bright, shiny new toys, and users of the older systems, naturally, want to upgrade. We have developed conversion tools that make older versions of the metadata tables compatible with the newer interfaces. Since there are hundreds of MDBs of varying ages/versions, this only fuels our desire for a single database – the Oracle solution described above.

SDTM Work Flow. The second area of attention to existing metadata is based on the SDTM-related work flow initiated during Stage 4. Once the metadata stabilized and we completed several projects, we explored the possibility of using the metadata to create program statements that would create variables. Since many of the SDTM variables were copies of raw data or simple transformations, it seemed that the definition field of the metadata could hold specially-coded instructions. The following list shows some examples of the syntax:

- Assign a character constant: `constant | 'text'`
- Assign a numeric constant: `constant | number`
- Copy a variable from a dataset: `copy | libref.dataset.variable`
- Convert a numeric variable: `recode | libref.dataset.variable | format.`
- Convert a character variable: `recode | libref.dataset.variable | $format.`
- Convert a date variable to ISO 8601 format: `date | lib.dataset.variable [ | format. ]`
- Convert a date-time variable to ISO 8601 format: `dtm | lib.dataset.variable`

As always, the metadata is complemented by a tool that makes it easily accessible to the end-users, in this case, the statistical programmers. The `%seedDomain` macro passes through the metadata for a user-specified domain, examining definitions for keywords. It generates a plain text file containing all the necessary LIBNAME, DATA, SET and assignment statements to create the variables. The programmer can modify the output file to create the variables not automatically generated. This saves a good deal of programmer time, and allows the programmer to focus on coding for the more difficult, non-automated variables. It is also interesting to note that this tool required no *structural* changes to the metadata. All that changed was the *content* of existing metadata fields.

Integration. Our department at Rho focuses on supporting FDA submissions. Other groups have developed systems to support other aspects of the company. Thanks to focus groups, department "appreciation" presentations, and word of mouth we are becoming aware of the potential for drawing on other systems' data (we have also become aware of the need to better document our metadata for potential users outside our usual clientele). One of these systems, used by project managers to track the progress of dataset and display creation, supplements one of the more complex and heavily-documented submissions services offered by Rho. A view to the Oracle tables gave us access to the tracking system's contents and saved us a significant amount of interface and tool design effort. One of our queries to the system is reading information about displays. The display numbers, program names, titles and footnotes and other information, for example, are used as part of our automated documentation tools.

Our use of the project tracker also demonstrates the phasing out of one of our systems. One of our earlier projects required rapid turnaround of hundreds of displays, many of which varied only by populations (gender, age group, treatment, etc.). We developed metadata and tools that exploited the similarities of the displays. Once the larger-scoped, Oracle-based system was developed, it became apparent that the display-related information in our old system was not needed, and so the TABLES metadata was retired. You can't be afraid to throw things out.

Training. Throughout the life of this and other stages' projects, we've seen new interfaces, new tools, new services we can provide to both internal and external clients, and *many* new biostatisticians and programmers. Training has become one of the most important functions of our department; if a helpful interface feature or time-saving macro is not made known to our various constituencies it might as well not exist. Given the rapidly changing and ever growing demands placed on the department, it's natural for us to adopt a work flow that boils down to "got the spec, wrote the code, tested the code, what's next on the list?"

What needs to be added to the quote is: "let me email the users and let them know what I've done." If the work was incremental – a new screen for the metadata entry interface or an additional option in an existing macro – an email may be sufficient. If, however, the work is significantly different or requires a work flow change, more formal training may be in order. *How* to inform and train the users cab be a matter of personal preference, but *whether* to inform and train is not negotiable.

We should also note that periodic refresher training is also worthwhile, even if a tool has not undergone any significant changes. By way of example, think about your use of a PROC or DATA step feature that you use on a daily basis. It is likely that you learned a large portion of its functionality years ago and, having grown comfortable with its use, you have not returned to the manuals or help screens. It is also likely that the tool in question has been enhanced since your first exposure to it and that you'll be surprised when you read about the "x" option (helpful, and present years ago) or the "y" option (*really* helpful, and added since then). A refresher training session is always helpful, if only to re-present the Big Picture of how the metadata work flow is organized and to remind you what tools are available.

Stage 5 – "System Maturity" – Summary of Changes and Enhancements

User Base

- ✓ Statisticians
- ✓ Programmers

Skill Set

- ✓ SAS
- ✓ Access
- ✓ Oracle
- ✓ version control software

Interface

- ✓ Interface to SDTM
- ✓ Forms point to Oracle back end

Key Applications and Tools

Macros to:
- ✓ Display archive/audit-trail metadata
- ✓ Automate creation of some SDTM variables
- ✓ Interface with other systems

Metadata

- ✓ SDTM archive
- ✓ redesign for Oracle
- ✓ display values taken from other systems
- ✓ TABLES retired

Lessons

- ✓ Consolidation into a single database is more secure, easier to maintain, at the possible cost of responsiveness to unique needs required by a project
- ✓ User input is especially important for tools that affect work flow (e.g., automating SDTM variable creation)
- ✓ Knowledge of other data sources, and the skill set to extract data from them, can save needless, redundant development time.
- ✓ Training/refresher courses are especially important when staff growth is rapid.

Conclusion

The Rho metadata system has evolved from a collection of metadata tables with a minimal interface to a relational database with a customized user-friendly full featured front end and a set of complementary tools. In 2001, at the onset of our first NDA project, we did not have the necessary metadata or related tools to complete such a large and complex project. The following example demonstrates how far our capabilities have come.

We recently completed an NDA submission project containing over twenty studies databases, an integrated database, and over one thousand displays. The client provided analysis datasets for some of the studies. We wrote custom programs to seed our analysis metadata tables, using macros from our general-purpose macro library to gather information from the SAS dictionary tables. As is typical in such projects, during the last four days before delivery to the client, specifications for six of the study databases and the integrated database were still being updated. Because we had well-trained users, a solid interface, and easily-programmed, time-tested tools for creating output we were able to continually update the submission databases as the metadata changed. This required a continual revalidation of the submission databases, creating and validating the transport datasets, and producing the documentation for each database. This process was repeated seamlessly as errors were found during the QC process and corrections were made to the metadata. Such rapid (about an hour to reproduce) turnaround for so many projects simply would not have been possible without the metadata, tools, and our body of experience. Likewise, the initial population of metadata for the client-supplied datasets was greatly simplified by the use of macros that were developed over the years.

As our metadata system evolved we learned numerous valuable lessons:

- Metadata driven applications have tremendous potential to improve efficiency and to automate work flow.
- A set of complementary metadata tools is essential to effectively leverage use of the metadata.
- The user interface for metadata entry is critical. If the users like it, they will use it. Otherwise, it might as well not exist.
- Continuously hold focus group sessions with end-users. End users are the best source of data for what's working and what's not working.
- As the number of users and projects increase, consider and plan for upgrades to the backend database.
- We're not longer just application programmers. We're now in the software development business. If you build it, you own it. That means dealing with maintenance, versioning, archival, and software life cycle issues in general.
- Documentation and ongoing training are vital.
- Build a development team that has people who are comfortable with strategic, high-level perspectives as well as the day-to-day, low-level hunt for missing semicolons.
- Systems have a much better chance of success and longevity if they are developed in an environment where it is all right if "Plan A" does not always work. Research and development teams learn as much from failure as success.

Contact

Your comments and questions are valued and welcome. Address correspondence to:

Jeff Abolafia: Jeff_Abolafia@RhoWorld.com
Frank DiIorio: frank@CodeCraftersInc.com

References

Previous papers by the authors, focusing on metadata and tool development:

- Abolafia, Jeff, “What Would I Do Without PROC SQL And The Macro Language,” 2005. Proceedings of the Thirtieth Annual SAS® Users Group International Conference.
- DiIorio, Frank, “Controlling Macro Output or, ‘What Happens in the Macro, Stays in the Macro’,” 2006. Proceedings of the Fourteenth Annual SouthEast SAS® Users Group Conference.
- DiIorio, Frank, “Rules for Tools: The SAS Utility Primer,” 2005. Proceedings of the Thirtieth Annual SAS® Users Group International Conference.
- DiIorio, Frank and Jeff Abolafia, 'From CRF Data to Define.XML: Going “End to End” with Metadata', 2007. Proceedings of the Pharmaceutical SAS Users Group Conference.
- DiIorio, Frank and Jeff Abolafia, “The Design and Use of Metadata: Part Fine Art, Part Black Art,” 2006. Proceedings of the Thirty-first Annual SAS® Users Group International Conference.
- DiIorio, Frank and Jeff Abolafia, “Dictionary Tables and Views: Essential Tools for Serious Applications,” 2004. Proceedings of the Twenty-ninth Annual SAS® Users Group International Conference.
- Also see http://www.CodeCraftersInc.com for other papers and resources not directly related to the current paper.

Fine Print