Drug Information Journal, Vol. 32, pp. 897–904, 1998
Printed in the USA. All rights reserved.
0092-8615/98
Copyright © 1998 Drug Information Association Inc.

USING PROGRAMMING AND DATABASE STANDARDS TO PROMOTE EFFICIENCY*

J. MEIMEI MA, PHD
Associate Director, Clinical Data Support and Analysis, Quintiles, Inc., Research Triangle Park, North Carolina
SANDRA SCHLOTZHAUER, MS
Principal, Schlotzhauer Consulting, Chapel Hill, North Carolina

The pharmaceutical industry cannot function efficiently without standards. What is meant, however, by standards for clinical research programming and analysis? How can standards help with the activities of data management, data cleaning, statistical analysis, study tabulation, and statistical writing? Where can descriptions of relevant standards be found?

Programming standards are discussed in the areas of programming style, variable names, documentation, formatting, and use of software options. Database standards are discussed in terms of consistency across studies and the effect on programmer efficiency. Software standards are also briefly discussed. Because of differences in terminology, first the terms as used in the paper are defined.

Disadvantages of standards include the need for a strong corporate culture, requirements for updating standards, and increased overhead for tools and management. More importantly, advantages of standards include increased communication and better teamwork, more effective use of programmer time, and increased ability to perform computer systems validation efficiently.

Key Words: Data management; Programming standards; Database standards; Validation; Electronic submissions

INTRODUCTION

CLINICAL RESEARCH PROGRAMMING and analysis includes the activities of data management, data cleaning, statistical analysis, study tabulation, and statistical report writing. In this context, programming involves a team of database administrators, data managers, statistical programmers, and systems programmers. From an intellectual viewpoint, this area of the pharmaceutical industry cannot function efficiently without standards. The term “standard,” however, is one with multiple definitions. In addition, the word “efficiency” includes multiple meanings. Focusing on how standards are used provides a foundation for justifying the implementation of standards. A broad range of standards, from programming style and database structure to software selection, are discussed. The examples focus on analysis programming, but the principles also apply to data management.

Reprint address: J. Meimei Ma, Quintiles, Inc., P.O. Box 13979, Research Triangle Park, NC 27709–3979.

*TRADEMARKS—All brand and product names are registered trademarks or trademarks of their respective companies.

TERMINOLOGY

Defining Standard

A dictionary definition for *standard* is: "a model to which others must conform" or: "something accepted as a basis for comparison." This paper distinguishes three different types of standards: de facto, external, and internal.

In a computerized environment, de facto standards evolve to the point that following them avoids time-consuming incompatibilities. For instance, Microsoft Word® is commonly used for word processing on microcomputer platforms. Most de facto standards are external to any single company.

External standards may be de facto standards or those defined by a group recognized by essentially all companies in the same industry. For instance, International Conference on Harmonization standards or HTML standards are external.

Internal standards are those established and enforced within a company. Usually internal standards are described as *guidelines* or *conventions* or are embedded in standard operating procedures or training manuals. Note that internal standards often follow de facto or external standards. This paper concentrates on standards in three interrelated areas of clinical research: programming, databases, and software.

Defining Efficiency

Like the idea of standards, defining efficiency must take into account a variety of elements. Although ultimately cost may be the most important measure, the top priority may initially be time to completion. Think of time in the sense of a delivery date for a final study report. To maximize efficiency, the goal should be to balance the elements of time and cost, which implies setting priorities differently for each situation.

The primary efficiency elements for research programming separate into two main categories: *machine* and *human*. Machine elements include: computer processing time for calculations or data manipulations, and processing time for reading or writing computerized information, called input/output (I/O). Elements related to human efficiency include: time, experience level, and the priority attached to producing results quickly rather than inexpensively. Examples of activities that require programmer time include:

- Designing databases,
- Planning or writing new programs,
- Revising or running existing programs,
- Maintaining documentation, and
- Learning new techniques.

When working with programmers, another efficiency factor is whether work is to be done by a group or an individual. Setting standards is important in either case. The methods of enforcement, however, are quite different.

PROGRAMMING STANDARDS

Standards for programs are critical for any project that lasts longer than a week, regardless of what computer language is used. This includes almost every aspect of clinical programming, with the possible exception of one-time ad-hoc analyses. Since one-time analyses are often repeated for a later protocol, however, use of standards is recommended even in this situation. A minimum set of internal standards should describe a consistent programming style, naming conventions for variables and data sets, and program documentation. For some software packages, additional programming standards can include using titles and footnotes, using standard format libraries, and documenting program options. Programming standards always promote human efficiency and can be designed to help machine efficiency too.

Programming Style

Programming style can be divided into two areas: *form* and *content*. *Form standards* cover visual cues (such as indentation) and certain aspects of naming conventions. *Content standards* cover logic guidelines, naming conventions, and testing strategies. External standards for programming style can typically be found in software-specific references; for a general set of programming principles, see Davis (1). If internal standards follow de facto standards for form, such as indentation and blank lines, then training new programmers will be simpler.

Naming Variables and Data Sets

Consider how variable naming conventions make programs easier to understand. Suppose a written standard is used to train all new programmers. Imagine the impact of the following guidelines:

- Variable and data set names should be descriptive,
- Names must be eight characters or less, and
- Use consistent names across programs.

Even with these simple guidelines, programmers would not be wasting time because of:

- Deciphering mystery variables such as X1, X2, Y, Z; or mystery data sets such as DS001,
- Resolving mismatches between analysis files and the CRF database, SAS® and ORACLE®, for example, and
- Finding sex and gender in different programs.

Clearly, more detailed naming conventions lead to more time savings and better teamwork. Maintenance of a standard that listed every possible variable, however, would be impractical. A comprehensive but obsolete standard will cause people to lose time worrying about—or waiting for—an updated version of the standard.

Program Documentation

The importance of documentation cannot be overstated. In a regulated environment, the only way to pass a quality assurance audit is to have programs with documentation that is consistent and complete. For instance, if every program begins with a header (a set of comment lines containing the same information in the same format) then anyone can quickly understand the documentation for the program. Standards for program documentation should also include guidelines about when to invest time creating and updating program comments.

A typical program header could include:

- Program name and location,
- Programmer name,
- Purpose of the program,
- Other programs to be run prior to this one,
- Data issues (is a specific data set structure or data set name needed?),
- Input and output files,
- Macro parameters,
- Program history (dates modified, by whom, summary of modifications).

With this type of header information, a programmer revising a program will spend less time deciphering code and more time on actual programming. Note that some of the items would not naturally exist in the code. The actual format of the header is not critical. The important point is to have a well-defined standard header style that all programmers understand and follow.

Documentation Using Titles and Footnotes

Documenting program output using standardized titles and footnotes should be easy to implement, but is often neglected. It is much easier to understand and replicate output with titles and footnotes that identify the data sets, program name, and program version. Having output with dates is fundamental to trace changes during development and testing.

Standard Format Libraries

Many software packages provide some form of code/decode tools. Using SAS software as an example, assume a standard database coding schemes exists. When using the SAS system to generate study tables, establishing SAS format libraries helps enforce the coding standard. In other words, before a standard format is worth storing in the library, a consistent definition such as 1 = YES and 0 = NO for all Yes/No variables in a protocol should be established. Using a single format library helps standardize formats across studies for a drug project by ensuring that any variables that come from multiple sources are assigned to the same format. This improves programming efficiency for safety analyses that combine data from many protocols.

Documenting Options

Many software packages provide options that affect the appearance of the output. Not only should standards exist for these options, they should be documented explicitly in all programs. To understand why setting a standard for the documentation of options is important, consider the situation where no standard exists and a new programmer needs to reproduce a statistical table. The original program is run and the rerun table contains the same numbers, but slight differences exist in the output layout (font, margins, etc.). New programmers are forced to be less efficient because they must run the program multiple times, testing different combinations of options, before successfully reproducing the exact same table.

DATABASE STANDARDS

The initial objective of database standards must be to create a usable database for a specific purpose. Once this objective is met, standards can address consistency across studies and issues that affect programmer efficiency. In clinical trial research, implementing database standards must start with the design of the case report forms (CRFs). For example, if the CRF includes weight but does not include height then creating a body mass index (BMI) will be impossible later. Database standards are an important aspect of quality data management, as noted by Calvert and Ma (2).

Consistency Across Studies

Much of the data collected for different drug projects falls into consistent content areas. For instance, almost all studies include medical history, adverse events, concomitant medications, vital signs, and demographics. Developing standards for the content of these data sets increases efficiency in developing programs that use the data sets. Balancing strict database structure requirements against the complexity of implementation for diverse Phase I protocols, however, may result in different decisions than for Phase II/III protocols. For critical safety information, enforcing standards across all drug products will generally pass a cost/benefit analysis. Enforcing consistency is more practical for critical safety-related items because the standards apply to a relatively small set of variables.

Database Standards Affect Programmer Efficiency

As an example of how database standards affect programmer efficiency, consider a table that summarizes adverse events by various demographic categories such as sex, age category, and treatment. If the programmer knows that the adverse event data set contains all relevant variables, the program can be written fairly quickly. If she/he knows that the adverse event data set must be merged with the demographic data set, and that the two data sets follow a standard structure, then the program can still be written fairly quickly. If, however, she/he does not know which data sets are needed, or cannot rely on a standard structure for the data sets, the program development and testing will require more time and effort.

As another example, the identifier used to match records across files must be described explicitly and used consistently to avoid extra effort. For instance, imagine the situation where a patient is identified with PATID = 21 in an initial protocol and PATID = 10021 in the extension study because the information for investigator was attached. Matching is still possible but requires more programming and quality control.

Using character variables may enhance both programmer and machine efficiency. Defining a standard for a "value set" across protocols is important. For instance, less effort is needed if gender is always coded as 1/2 for male/female instead of being mixed with 0/1 for a few protocols. There is a need to balance these efficiencies with the need to minimize requirements for end-users to remember decodings. One approach that provides a value set is to use a standard formatting approach, such as SAS format libraries, to handle decodings.

Database standards also impact electronic submissions to the Food and Drug Administration (FDA) and other regulatory agencies. If the electronic submission is simply a collection of database files (such as SAS transport files) and analysis programs, using standard data file names for different protocols and organizing the data files in directories for each protocol can be an efficiency tool. If the electronic submission provides more context, such as PDF files that identify data set names, variable names, variable formats, and the like, using standards can be an efficiency tool in preparing these files. For a nonprogrammer reviewing the data at a regulatory agency, the ability to see the same variable names and labels across studies can be an efficiency tool. Reviewers become more comfortable with the database when fewer exceptions or inconsistencies appear during the learning process.

SOFTWARE STANDARDS

Unlike programming and database standards, differing corporate priorities and computer environments may lead to different, but equally valid, choices for software standards. Relevant issues to consider include:

- Support availability,
- Importance of consistency,
- Flexibility,
- Compatibility, and
- Longevity.

If internal standards are established around a single software package or language, then support and training will be less costly. Following a de facto standard may reduce such costs even further. Groups with special analysis requirements, however, may want the flexibility that would be possible if a greater variety of software tools were allowed.

External standards should influence the selection process. For instance, the growing use of web browsers and HTML suggests that selecting software that does not easily create HTML output is a weaker choice. In this industry, transferring files that are SAS data sets or ORACLE tables has become far more common than dBase® or DB2® because of changing external standards. No choice should be made without considering whether or not the software is likely to be around in five or 10 years.

DISADVANTAGES OF ENFORCED STANDARDS

Although the authors strongly support using and enforcing standards, disadvantages do exist. Managers at all levels need to be aware of potentially difficult issues related to establishing and enforcing standards. Three disadvantages most relevant to programming activities are:

- The corporate culture must encourage the use of standards,
- Standards must change over time, which requires ongoing effort, and
- Effective standards require dedicated staff time for using related tools.

Corporate Culture

One potential disadvantage is the effort required to develop a corporate culture that encourages the use of standards. People at all levels must be involved. In many programming cultures, the model is to rely on individual programmer heroics. In these situations, the culture emphasizes getting a project done without considering maintenance and support costs for inconsistent programs and databases. As described in Humphrey (3), the Capability Maturity Model includes techniques for building corporate cultures that encourage standards.

For current programming staff in particular, both programmers and managers should acknowledge the increase in time and effort required to comply with standards or to adjust individual programming style to corporate standards. This extra time and effort is offset, however, by the increase in efficiency of the group as a whole. Sharing of programs becomes easier, as does modifying existing programs to meet new needs.

With support from the corporate culture, people required to follow standards will recognize that enforcing standards is not a personal affront but is required just to accomplish the goal in an efficient manner. For new programmers, communication about the importance of adhering to standards must begin during the hiring process by assessing whether the candidate is willing to work within such a culture. Hiring and training someone who wants to fight corporate standards is expensive and counterproductive.

The importance of a corporate culture that supports standards is difficult to overstate. While a given group of programmers and a given manager may recognize the advantages, these people need to know that their efforts are recognized and valued. Otherwise, attempts to use or enforce standards will occur in an environment that ultimately cripples the effort.

Standards Change

The reality is that standards change over time. Maintenance of standards is an ongoing challenge for both large and small organizations. Often external changes such as software updates or new de facto standards will cause internal standards to become obsolete.

Increased Overhead

Another potential disadvantage of standards is the need for appropriate tools and management of the tools. While the need to hire an additional programmer is often recognized, the need to hire staff to provide and maintain standardization tools is rarely appreciated. If the company requires standard documentation of programs, compliance will be much higher with a tool that helps the programmers create such documentation. This tool could be as simple as a Word template stored in a central location. All programmers would access Word, open the template, provide the information, and store the documentation in a central location. Without such a tool, program documentation is less likely to adhere to the internal standard. Remember that the Word template will eventually need revision. The template could even need to be replaced if computing environment changes make Word less accessible to programmers, for example, changing to UNIX.

ADVANTAGES OF ENFORCED STANDARDS

On balance, the authors believe that the advantages of standards clearly outweigh the disadvantages. Key advantages of programming, database, and software standards include:

- Increased communication and teamwork,
- More effective use of programmer time, and
- Increased ability to perform computer systems validation.

Increased Communication

Most drug development projects today involve multidisciplinary teams, including more than one programmer. Standards enable programmers to communicate more easily among themselves and with people from other disciplines. One reason is that standards provide a common vocabulary that everyone understands. Thus, team members may focus on the problem rather than being distracted by the need to decipher what another person is trying to say. This increase in human efficiency can occur when discussing a specific program, reviewing or revising another programmer's work, or training new staff. Better communication among programmers also affects manager efficiency in communicating effectively with programmers. See Humphrey (3) for detailed discussions of managing technical staff.

An important example of how standards can improve communication is the reduced effort required to train new programmers. The existence of standards usually implies that resources such as a central library of programs or written guidelines about naming conventions already exist. Thus, the training process is likely to be simpler and shorter—and less expensive.

The existence of standards increases the likelihood that people using diverse computer applications will be able to easily plan to share data or program code. Common examples of different types of applications that require standards for effective data sharing or systems integration include data management systems for CRF data based on ORACLE, safety surveillance applications for regulatory reporting of serious adverse events, and analysis tools based on the SAS system. Typically, the people using diverse systems are in different functional groups. Consider the advantages if everyone already understands a standard set of patient identification variables when discussing a process that requires data from all three applications.

Effective Use of Programmer Time

Standards help reduce confusion when programmers work on previously created programs, whether their own or someone else's. Tools such as a program header containing version number, appropriate titles and footnotes, and standard names for data files and variables all reduce the programmer's efforts to make sure she/he is looking at the latest and most correct version of the program.

Standards also help reduce duplication of effort. Suppose a programmer must start by deciphering another programmer's code. Without programming standards, the second programmer might simply decide to start over because understanding the program would take more effort than writing and testing a new program. In a scenario with standards, suppose a programmer is asked to create a new table similar to an existing one. She/he can quickly find the existing program by checking the program documentation. Less time will be required to test a modified program compared to developing a completely new one because the program documentation follows a standard.

Maintaining programs in the pharmaceutical industry has its own set of issues, involving potential future audits by regulatory agencies. The legacy version of the program must be kept in case a rerun is required for a previously submitted marketing application. The company must work, however, to stay current with software changes, database changes, or operating system changes. Obviously, software standards increase the likelihood that old programs can still be run, while assuring that new capabilities are incorporated in an organized fashion.

Natural Computer System Validation (CSV)

Adhering to principles of CSV is especially important in this industry, with the ever-present possibility of a regulatory audit. To check that CSV was applied appropriately, review by quality assurance (QA) of data management and analysis programs must occur. QA audits are more efficient when standards exist because internal auditors can spend less time achieving a "comfort level" with the procedures used. External auditors will also take less time. More importantly, they will find it easier to conclude that work was done in a high quality environment. Elements reviewed by many auditors are discussed by Chapman (4).

Imagine two contrasting situations: one with and one without standards. In the first, the auditor finds no enforced standards, only haphazard locations for programs, inconsistent naming conventions, a variety of documentation approaches (including no program header!), multiple versions of programs with no apparent connection, and inadequate communication between programmers. In the second, the auditor finds a centralized storage location for programs or macros, consistent naming conventions, consistent documentation approaches, version control of programs, and good communication between programmers and other team members. Needless to say, the audit of the situation with standards will take less time to complete.

Standards promote the creation of documentation, which is the heart of CSV. To begin with, the descriptions of the standards themselves are natural documentation. Programming style and naming conventions lead to self-documenting data files, programs, and output with little extra effort. Although this type of documentation is not the only one required for CSV, it certainly helps. Remember, ideally, CSV efforts should be the responsibility of team members from all disciplines, not just programmers.

STANDARDS PROMOTE EFFICIENCY

A broad range of ideas related to standards have been covered, but the underlying theme remains that the existence of programming, database, and software standards will result in greater efficiency. Greater human efficiency is expected from better communication and teamwork. Greater computer efficiency is expected from centralized storage, reuse of optimized algorithms, reuse of well-designed templates for database structures, and more.

Clinical research requires a team effort to successfully complete any drug development program. Therefore, the investment in creating and enforcing standards always pays dividends in the long run.

REFERENCES

1. Davis AM. *201 Principles of Software Development.* New York: McGraw-Hill Inc.; 1995.
2. Calvert W, Ma JM. *Concepts and Case Studies in Data Management.* Cary, NC: SAS Institute Inc.; 1996.
3. Humphrey W. *Managing Technical People: Innovation, Teamwork, and the Software Process*. Reading, MA: Addison-Wesley; 1997.
4. Chapman KG. *Computer Systems Validation for the Pharmaceutical and Medical Devices Industries*. Second Edition. Libertyville, IL: ALAREN Press; 1994.