Developing Standards With and For the User

Robert E. Shanklin, Jr.
Computer Sciences Corporation
System Sciences Division
4600 Powder Mill Road
Beltsville, Maryland 20706

Abstract: The success of any set of standards depends on the degree of compliance at all technical and management levels. Several problems with standards produced by classical development efforts are discussed and are attributed to the failure to consider the standards user during the generation of standards. The people who use and need standards the most (e.g., programmers using coding standards and designers using PDL standards) should be involved in the generation of those standards. Users want standards to be clear, short, relevant, and easy to reference. An alternative approach to standards development is presented, and that approach is shown to solve some problems with standards compliance.

I. Introduction

There is some recent good news on the general subject of standards. The current emphasis on process [e.g., Software Engineering Institute (SEI) studies [Reference 1]] and on quality [e.g., Total Quality Management (TQM) initiatives] provides an environment that encourages the development of standards and the development of the training, measurement programs, and methodologies that support (and are supported by) those standards. Several good development methodologies [e.g., the Computer Sciences Corporation (CSC) Digital System Development Methodology (DSDM) [Reference 2]] have been developed that serve as a sound basis for good standards. Large numbers of authoritative standards, procedures, and data item descriptions (DIDs) have been generated over the last 2 decades.

There is, however, some bad news. Many programmers (particularly those with fewer years of experience) don’t like standards because:

• Standards are imposed from "on high" without involving those who "do the work."

• Many standards are too long for routine reference use.

• The content and language of many standards are not addressed to the user of those standards.

• Many standards are written to cover the universe (e.g., cover both hardware and software) and, hence, are not tailored to the user's work.

• Little, if any, consideration has been given to making their standards handbook an easy-to-use, quick reference.

There is a single underlying cause for all of these complaints—the user, that is to say the standards customer, has been largely ignored. The people who use and need standards the most would like to be involved in their production and would like those standards to be clear, short, relevant, and easy to reference. Note that all of these requests are reasonable and productivity oriented. A clearly defined process increases productivity by freeing producers to focus on technical issues. [Reference 3]

At CSC, a complete and general methodology (DSDM) and a supporting set of standards and procedures have been in place for several years and are continually upgraded. In 1989, on CSC's billion-dollar Systems, Engineering, and Analysis Support (SEAS) contract with NASA's Goddard Space Flight Center (GSFC), a contract-specific methodology (SEAS System Development Methodology (SSDM)) that is consistent with DSDM was released. For the past year and a half, a set of contract-specific standards and procedures has been under development. The principal early focus of this development has been on software standards.

The process used on CSC's SEAS standards development effort is upside-down relative to the usual process: the users (the people who do the work) write the standards; managers and senior staff (e.g., product assurance officers) are responsible only for facilitation and reviews. The rules guiding this new approach can be summarized as follows:

• Use workers (not managers and senior staff) as authors.

• Publicly recognize the authors' contributions.

• Use first- or second-level line managers (managers in direct contact with workers) as coordinators and early reviewers.

• Use senior staff and senior managers only for final review and approval.

• Ensure that faith is kept with the authors, i.e., that no additional requirements are added during the review cycle.
The objective of these rules is to produce five-page standards that the person doing the work will find truly helpful in requests from technical staff for under-the-table advance results to date are encouraging. There have been repeated producing quality software. copies of standards. There have been numerous overheard-in-the-hall discussions (Note that with this standards development approach, the users and authors are colleagues with immediate access to the-hall discussions This effort has produced to date more than 50 standards by self-enforcement) by the technical staff. procedures, and DIDs that were collected from software developers. This paper first presents a list of criticisms of standards, classically produced standards. Next, a new approach to the generation of standards is presented that is designed to be responsive to the development that addresses most of the complaints; Section IV discusses some of the problems with and tradeoffs required by the presented approach. Note that only the production methods and, to some extent, the format and style are new; boundary restraints typical of a normal standards effort are applicable. That is, the new standards are compatible with all applicable NASA, GSFC, and CSC standards and methodologies and are generated with the prime goal of producing uniform, quality software.

The alternative development approach to standards in use on CSC's SEAS contract can be described as bottom-up. The alternative development approach to standards in use on CSC's SEAS contract can be described as bottom-up. The

- Ensure that the review process, in attempting to "improve the purity" of the standards, does not remove examples, references, and explanations that the users want and need.
- Ensure that the review and editing process keeps the language simple and direct, i.e., does not introduce formal, ponderous, or legalistic phrasing.
- Develop training courses centered on specific standards.
- Solicit and use criticism and suggestions for updates.

The objective of these rules is to produce five-page standards that the person doing the work will find truly helpful in producing quality software.

Results to date are encouraging. There have been repeated requests from technical staff for under-the-table advance copies of standards. There have been numerous overheard-in-the-hall discussions as to software standards alternatives. (Note that with this standards development approach, the users and authors are colleagues with immediate access to each other.) These requests and discussions bode well for "ownership" by the technical staff and for acceptance (and self-enforcement) by the technical staff.

This effort has produced to date more than 50 standards by more than 40 authors.

This paper first presents a list of criticisms of standards, procedures, and DIDs that were collected from software developers. These criticisms support the hypothesis that classically produced standards are not oriented towards the developers who are the users (i.e., the customer) of the standards. Next, a new approach to the generation of standards is presented that is designed to be responsive to the developers' criticisms of historically typical standards efforts. Then, a list of difficulties with the recommended new approach is discussed. Finally, the status and preliminary results of the effort are summarized.

II. User Criticisms of Typical Standards

This section conveys several criticisms and complaints collected during informal discussions with software developers, the majority of whom were degreed scientific programmers of varied years of experience. (The more strongly worded criticisms were typically from the less experienced members of that group.)

Several developers commented that standards were developed and imposed from "on high" with no consultation with those who "did the work" (applied the standard). One interviewee commented that this is contradictory to TQM and modern quality approaches; if quality must indeed start at the bottom, then standards should be based on bottom-up consultations. Another developer commented that the standards author (a very senior staff member) was located in corporate headquarters and could not really know the local work environment.

There were numerous complaints about the length of comprehensive standards—"When I'm in a hurry to find out the answer to a simple coding question, I still have to read 11 pages of a 47-page standard."

Many programmers said that the linguistic style and content were legalistic. The ideal standard should say, "Do this, do this, don't do this, and watch out for this," in simple, direct words. [Reference 4] Also, if the real objective of standards is to help people implement quality software, a few words of justification or an example should occasionally be allowed in standards if such exceptions could clarify the whole thrust of the standard. Why don't DIDs have a cover page giving purpose, applicability, and even a reference or two so that a person writing a document for the first time has an easy head start? [Reference 5]

Several comments were received that focused on the global, "cover-the-world" content of some standards. Options that were not allowed on the contract or that were not relevant to the contract should not waste space and the reader's time. [Reference 6]

Several reviewers noted that, if standards are to be used as a routine reference, standards handbooks should have the amenities of a good reference, e.g., a overview, table of contents, a logical order (rather than chronological order of issue) with subsections (see example in Figure 1), tabs, glossary, cross-references, and perhaps even an index.

Note that all of these criticisms are reasonable from the user's viewpoint and that most are truly oriented toward improved productivity and quality.

III. Developing Standards With and For the User

Two important patterns to the complaints in the previous section are clear. First, users of standards (e.g., the programmer for a coding standard) want to be involved in the process that generates the standards, and they want the standards to be clear, short, and easy to reference. Simply, the standards customer wants to be treated as such. Second, many of the criticisms suggest approaches that are contrary to the carefully thought out classical approach and suggest goals that will require tradeoff compromises with the classical approach.

This section presents an alternative approach to standards development that addresses most of the complaints; Section IV discusses some of the problems with and tradeoffs required by the presented approach. Note that only the production methods and, to some extent, the format and style are new; boundary restraints typical of a normal standards effort are applicable. That is, the new standards are compatible with all applicable NASA, GSFC, and CSC standards and methodologies and are generated with the prime goal of producing uniform, quality software.

The alternative development approach to standards in use on CSC's SEAS contract can be described as bottom-up. The
SEAS approach is bottom-up in that the standards are written by technical employees who are directly responsible for doing the work with which the standard is concerned, e.g., the software unit design standard was written by an employee who designs units. Senior staff (e.g., product assurance officers) and senior managers were involved only in the final review stages. The nine basic groundrules of the approach are as follows:

1. Use expert authors who actively work in the area covered by the standard. For example, use a programmer to write a coding standard, and use a project-level product assurance officer to write a DID for a project-level product assurance plan.

2. Thank all authors by public recognition and by acknowledgment in the standards handbook. This recognition clearly assigns the source, ownership, and responsibility for enforcement, i.e., clearly supports the tenant that quality is everyone's responsibility.

3. Use workers and first- and second-line managers (those with direct contact with the work) as coordinators and early reviewers.

4. Limit the involvement of senior management and senior staff to sponsorship, review, and final approval.

5. Ensure that faith is kept with the authors and early reviewers by not permitting the addition of requirements during the review process unless the author representative concurs with the additions. This rule is needed to show that the ownership and authorship truly rest with those who do the work.

6. Ensure that the review process does not remove occasional explanations, justifications, examples, or references that the authors and early reviewers felt necessary for clarity to the user. This rule can be difficult to uphold—the desire to improve the purity (as defined by the classic philosophy of standards) and the admirable desire to ensure absolute unambiguity will tend to metamorphose the standards to the classic traditional style and content during the review process.

7. Ensure that the review and editing process does not introduce stiff, complex, ponderous, or legalistic language.

8. Develop an appropriate training program that emphasizes the application of standards to specific phases of the software development life cycle.

9. Solicit and use criticisms and suggestions. Publicize the fact that quality requires change and improvement—standards must be continually updated. The ability of all to request updates also reinforces the joint ownership of the process and of the standards.

These rules can be summarized in a single goal: produces short, clear standards that the person doing the work will find most helpful in producing quality software. Conversely, do not permit the goal of producing pure, classically worded standards to override the goal of usefulness to the worker.

IV. Problems With the Suggested Approach to Standards

Several pitfalls and weaknesses of the suggested approach to standards development require discussion.

First, the use of many authors tends to produce material that is inconsistent in style and level of detail. Extensive editing is required, and that editing must be sensitive to the goals of preserving the language level and style desired by the user.

Second, the standards will exhibit a somewhat motley content, e.g., occasional procedural steps within standards, tutorial and justificatory asides, and DIDs with cover pages. This lack of purity can be viewed as a defect. However, as one first-line manager stated, "The average programmer with 2 years' experience doesn't know the difference between a standard
and a procedure and couldn't care less; he/she wants the five pages of coding information that will produce the highest quality code; period." Compromise and common sense are the only answer to this difficulty. An occasional parenthetical remark or an example that clarifies the intent must be kept. The notion that standards, like law books and insurance policies, require specialized interpreters and require purity at the expense of usefulness to the true customer must be opposed. Quality, clarity, and consistency (not purity) are the major objectives.

Finally, although user ownership and enforcement is encouraged, management must not delegate the responsibility for the standards. Quality starts at the bottom but is fostered from the top. Management must be on guard against unenforceable or inefficient rules.

V. Maintenance, Enhancement, and Enforcement

As with any set of standards, an efficient and responsive program for maintenance and enhancement is essential to ensure quality and continual improvement. A specific standard that includes a form for suggested changes is essential. The standards development approach recommended in this paper strongly encourages participation of all levels of employees because authors and users are colleagues.

Special-case exceptions to compliance with standards are handled by a formal waiver board reporting to senior management.

VI. Status and Initial Observations

The approach to standards development described here has, to date, produced 54 standards and procedures covering most of the software life cycle. Forty-six authors contributed to those standards. Sixty-eight more standards are in various stages of development.

Sufficient data are not yet available as to the effect of these standards on software quality, and that data will clearly be difficult to analyze. Moreover, distinguishing the quality produced by standards from the quality produced by training or methodology is, at best, extremely difficult. The demonstration that these standards are better than standards generated by classical methods will be harder still.

Nevertheless, several qualitative observations are encouraging. Dozens of staff members of all levels have requested advance copies of standards; they know from their colleagues (who are authors) that particular standards are in development, and they want the current best version until the official standards are issued. There are routine "hallway" discussions in which authors or coordinators explain and defend the standards; this was seldom observed when standards were imposed from on high. Several authors who disagreed on a "C" language standard enthusiastically joined a quality improvement committee and resolved their differences. More than 100 suggestion forms for changes and upgrades have been submitted. These qualitative observations bode well for user ownership and enforcement. Genuine interest in standards at the level at which those standards are first applied indicates a major victory. The standards customer likes a user-friendly product as does any customer.

References


