FURTHER PERSPECTIVES ON COMPUTER DOCUMENTATION: SYSTEM DEVELOPER VS. TECHNICAL WRITER

Lorena F. Truett
Energy Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831-6207
voice: 423-574-4225; fax 423-574-3895
email: truettlf@ornl.gov

Elithe Truett Carnes
The University of Tennessee
Energy, Environment, & Resources Center
10521 Research Drive, Suite 100
Knoxville, Tennessee 37932
voice: 423-974-8392; fax 423-974-8448
email: ecarnes@utk.edu

Prepared for
Currents '96
February 17, 1996
Cecil B. Day Campus, Mercer University
Atlanta, Georgia

The submitted manuscript has been authored by a contractor of the U.S. Government under contract DE-AC05-84OR21400. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.
DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.
Further Perspectives on Computer Documentation:
System Developer vs. Technical Writer

Lorena F. Truett and Elithe Truett Carnes

Between the computer-knowledgeable "techie" and the technical writer is a chasm created by differences in knowledge bases and skills. Although this gap is widened by misunderstandings and misconceptions of system development roles, it is bridged by mutual need and dual appreciation. This paper explores the reasons that problems arise during the production of software documentation and discusses possible methods to span the gap between developers and their fundamental public relations agents, the technical editors/writers.

INTRODUCTION

Cooperation among the members of a software development team is critical, as is an understanding and appreciation for the roles and tasks of other members of the team. Otherwise, the software may be very good but the documentation inadequate and ineffective. In actuality, the composition of a software development team is diverse; however, for the purposes of this paper, a software development team consists of two types of software professionals: developers and documenters. The term software documentation in this paper includes printed manuals, reports, and pamphlets that describe the system and/or assist the end-user or system maintenance personnel. It also includes hypertext and on-line help files but excludes programming comments embedded within the code.

The relationship between team dynamics and the production of computer software documentation has received inadequate attention. Budgets and schedules emphasize code development processes and include documentation activities almost as an afterthought. This project management blunder initiates the drama that exists between software developers (programmers/writers of code) and software documenters (technical writers).

SOLID GROUND AND THE CREATIVE SPIRIT

The developer of software code is normally a mathematician or computer scientist, a person firmly entrenched on solid ground: logical, reasonable, analytical. The writer, on the other hand, is usually a person of letters, of languages, of a creative spirit. And so arises the conflict between the "left brain" and "right brain" members of the team.

Engineering and artistic temperaments disagree on the appropriate document content as well as on the presentation style. They have different goals: that is, the developer is most concerned with software functionality and meeting the technical requirements; the documenter is most concerned with documentation accuracy, completeness, readability, and usability. According to Weiss (1), neither technical experts working alone nor a technical communicator working alone can produce usable user documentation. Effective, reliable, maintainable documentation "demands participation of both technical and communication experts" (1, p. 7).

Typically, the ratio of programmers to writers on a software development team is 9:1. Therefore, the technical writer provides documentation for programs produced by nine other individuals. This task is impossible without assistance from the developers, but user reviews of the draft documents are not always effective. Because programmers frequently believe that the code is critical and that the documentation is just "fluff," the technical team members dislike reviewing the documentation. The writer hears such comments as "It's not my job." The point of documentation review by the technical staff is to uncover technical flaws; however, often the technical reviewers are unable to read the document from this perspective and will make comments on format and totally ignore errors in logic. If this occurs, the review does not accomplish its purpose.

*Based on work performed at Oak Ridge National Laboratory, managed by Lockheed Martin Energy Systems under contract DE-AC05-84OR21400 for the U.S. Department of Energy.
THE CHICKEN AND THE EGG PROBLEM

Two approaches to software documentation exist: (1) determine a good document structure and make the documentation for all code fit into that format, no matter what the code is written to do; (2) produce programming code as defined by requirements and then write appropriate documentation to fit the code.

Most documentation for software developed for the military conforms to the first approach. Military guidelines establish a specific list of required documents and a document format for each. Unfortunately, the formats were set in stone several years ago, and technology has progressed far beyond these predefined limits. For example, a system's maintenance manual for a client-server, distributed database design is almost impossible to describe using the military format and outline which was well suited to mainframe, flat-file architectures.

The second approach is also flawed. In this approach, the technical writer is "behind" from beginning to end because the writer joins the team after the programmers are well into system development. They are involved with development tasks and do not want to "waste time" discussing fundamentals with the documenter. The writer may be excluded from technical discussions because it is assumed that he/she would not understand these discussions anyway.

The best documentation is produced by a flexible approach led by the technical writer and characterized by persistence. The writer does not need to understand all of the technical details of the software, but he/she must understand the system's functionality and must have a clear understanding of the user community. Similarly, the developer does not need to understand the writer's emphasis on format and design, but the "techie" must communicate the purpose of the software to the writer. Cooperation is critical, communication is essential, persistence is required, and patience makes the whole process more pleasant.

DOCUMENTATION STAGES

Several stages make up the documentation process. We have classified these as "Circling," "Bold Thrust," and "Iterative Progression." These stages are described fully in an earlier paper (2) and are briefly described below.

The Circling Theory assumes that a software development team is comprised of individuals with differing skills (such as tools usage), experiences, knowledge bases, and goals. The theory states that the team members will attempt to establish a hierarchy of importance. To advance their own positions, team members will emphasize their own strengths and/or point out the failures and inadequacies of others.

Circling diversions waste a lot of time. Because of impending deadlines (which were almost certainly established without input from the technical writer), the documenter produces a first draft based on his/her interpretation of what the software does. (We call this "Bold Thrust!") The review of this first draft is usually extreme: either too little or too much. Coordinating software development and software documentation is mandatory, but difficult to orchestrate until after the initial Bold Thrust. Bold Thrust can result in significant changes to the team dynamics. For example, the writer may be suddenly included in meetings and discussions, or there may be increased willingness among the developers to respond to questions posed by the writer, or the writer may simply quit the team.

The third stage in the documentation process, Iterative Progression, is the longest stage from a time perspective. From a task perspective, this stage accomplishes the most. This phase in the process is centered around cooperation and communication among the members of the software development team. According to Price and Korman (3), only three drafts of a document should be circulated for review. We agree that three drafts (four, if the initial "Bold Thrust" is counted) are sufficient if the writer is free to interact with technical personnel on miscellaneous details throughout the development cycle.

Toward the end of the Iterative Progression stage, when deadlines, budget crunches, and the eternal revision cycle are all factors, it is important that programmers and writer cooperate fully. The developer accepts the role of the technical writer as the public relations agent for the code/software. The technical writer accepts the reality of constant change (i.e., revisions and newer versions -- both to the code and to the documentation). This joint cooperation promotes team unity, and a good team produces a good product.

BRIDGE OVER TROUBLED WATERS

Acknowledging that there are differences between developers and documenters is the first step in solving the problem. If the technical writer comes on board after the developers, the project manager should hold a meeting to introduce the technical writer and explain the documentation role. Because code is the
responsibility of the developer and producing documentation is the responsibility of the technical writer, the writer must establish his/her leadership role for format and style of the documents. The writer should avoid Circling distractions and get involved quickly: review old documentation or memos, log in to the system as an end user and review screens, attend meetings, and/or discuss the requirements with other team members. Prior to Bold Thrust, the writer might organize a meeting to explain the technical review process and distribute a technical review handout. This handout can take many forms, but at the minimum it requests a rating of the technical correctness of the document, gets specific information on what is wrong and how to correct the error, and obtains the name of the reviewer. After the review by the technical staff, the writer should be prepared to accept the comments gracefully (all help is appreciated). The documentation effort must be viewed as a joint production, and the writer should not consider the review as a personal attack. Throughout Iterative Progression, the documenter should attempt to involve appropriate reviewers as much as possible but not to the point of interference with code development; three reviews should be sufficient. If at all possible, the project manager should set aside sufficient time for technical personnel to do the documentation reviews. Above all, the technical writer should remember that the input of the entire team is necessary, and a sense of humor is always helpful.

CONCLUSION

Effective teamwork produces a comprehensive package of software and documentation, as well as a more effective and efficient software development team. Through Circling maneuvers, a Bold Thrust, and Iterative Progression, the final product (though fleeting, because the next version is right behind it) arrives.

Computer documentation, like any formal writing, is never completed in "one fell swoop." The process of drafting and revising is continuous. Computer documentation differs from creative writing, however, because it is a collaborative effort by individuals with different skills using different tools, from different backgrounds and experiences, with different knowledge bases, and having different goals. Although good documentation cannot make bad code any better, the technical writer's skill certainly elicits the end-user's appreciation of good code.

REFERENCES


BIBLIOGRAPHY


AUTHOR INFORMATION

Lorena F. Truett
Computer Specialist, Energy Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831-6207 USA
phone: 423-574-4225
e-mail: truettlf@ornl.gov

Elithe Truett Carnes
Research Associate
The University of Tennessee, Energy, Environment,
& Resources Center
10521 Research Drive, Suite 100
Knoxville, Tennessee 37932
phone: 423-974-8392
e-mail: ecarnes@utk.edu

Lorena F. Truett holds bachelor's and master's
degrees in mathematics. She began work at Oak
Ridge National Laboratory in 1975. She has spent
the past several years managing a project to design,
develop, and integrate computer systems for
worldwide management of military cargo.

Elithe Truett Carnes holds a bachelor's degree in
English and history and a master's degree in
English literature. She is currently pursuing her
Ph.D. in English. She is a technical writer/editor with
the University of Tennessee, Pellissippi Research
Institute; she also develops hypermedia companion
files for software written at the Institute.