Consequences of the Engineering Approach to Technical Writing

Bob Waite
Dept. 52X
IBM Corporation
Highway 52 and 37th Street NW
Rochester, MN 55901
rwaite@us.ibm.com

Abstract

In a series of publications, Edmond Weiss describes the change in both programming and technical writing from an artistic craft to an engineering discipline. The change brought numerous benefits to the users and readers, to the programmers and writers, and to the companies they work for. In exchange for those benefits, Weiss says, writers had to give up control over the content and format of the documentation. They also lost the esthetic pride of creating a work of art and the personal satisfaction of teaching and protecting the reader. Weiss anticipates a future in which technical documentation is generated automatically and the technical documentor's job is merely to feed information databases. In the conclusion, I note a few differences of opinion, but overall I agree with Weiss's view.

I.7.1 Document Preparation—*document management*

Keywords*: documentation development, project management, collaboration, teamwork*

Introduction

In the Technical Communications course that I teach in the Graduate Programs in Software at the University of St. Thomas in St. Paul, Minnesota, I have used Edmond Weiss's book *How to Write Usable User Documentation* (1991) as a text for the past ten years. Parts of the book—especially those dealing with online information—are dated, but the concepts are not only still valid but crucially important and convincingly explained. Weiss's notion of documentation engineering is an obvious extension of principles taught in software engineering courses. For students impatient to get beyond my required course to the "good stuff" (programming courses), the engineering metaphor makes my course palatable and, for some, even enjoyable.

Having read Weiss's book several times, I am familiar with many of its ideas. (Note: Weiss has published several books, but *How to Write Usable User Documentation* is the only one mentioned in this article, so I refer to it as "the book" or "Weiss's book" merely as a convenience.) After encountering some of those ideas again in more recent articles by Weiss—"Of Document

ACM Journal of Computer Documentation, 2002,26:22-26
 ISSN 1527-6805

Databases, SGML, and Rhetorical Neutrality" (1993) and "The Retreat from Usability" (1995)—I immediately noticed that his current article, "Egoless Writing," is another reexamination of three recurring topics in Weiss's writing:

- The similarities between programming and technical communication
- The contrast between the artistic approach and the engineering approach to writing documentation
- The implications of the engineering approach.

In this article, I focus on the last of those three topics. I describe how Weiss's vision of the future for technical communicators has become more skeptical over the past decade. I conclude that, with a few minor quibbles, Weiss's view is essentially correct.

How to Write Usable User Documentation

In 1991, Weiss's perspective on the engineering approach was positive. His book describes the new notion of a document and names several benefits that it provides: "Nowadays, a document, or a piece of a document, is actually a paper view of a digitally stored entity. And digitally encoded entities are far easier to reproduce, revise, interconnect, and otherwise manipulate than any traditional form of communication" (Weiss, 1991, p. 8). These benefits accrue to the writers, to the document designers, to the writers' employers, and to the readers/users:

> A documentation process that learns the lessons of software engineering will achieve the five goals listed below:
> - It will improve the "fit" between user documents and the needs and convenience of the users...
> - It must reduce the skips, jumps, and detours...
> - It will allow writers to work in teams and in parallel...
> - It will enhance the clarity, readability, and reliability...
> - It will generate publications and products that are maintainable and modifiable... (p. 44).

The most important stakeholders, the readers/users, benefit significantly. The engineering approach produces "documents that compel readers to find what they need, in the most efficient sequence, and with a level of effort that neither discourages them nor lowers their productivity" (p.16). Guaranteed success.

The writers also benefit. For one thing, their use of a definable, repeatable process leads to predictable results. Such consistency is the essence of standards-based quality. Furthermore, the engineering approach enables writers to quantify the value of their contribution to the product: "The usability of documentation (that is, how appropriate, accessible, and reliable it is) can be defined and measured" (p. 7). Since our society values science and technology, the engineering approach should "enhance the power and professionalism" of technical communicators (p. viii).

At the same time that the writers' role becomes more respected, their actual work becomes easier. The designer gives the writer a model (already reviewed and approved) of the entire document. In addition, for each module that the writer is responsible for, the designer provides a headline, a thesis passage, and one or more exhibits. All the writer has to do is generate 200-700 words of missing details in the text of each module. In short, says Weiss, "Anyone who will not write under these conditions – especially when told that grammar doesn't count – probably would not write under any conditions" (p. 134). Under those conditions, it is often possible to get the subject matter expert (the product developer) to write the first draft. In that case, "the result, though awkward in style, is likely to be accurate. And it also liberates the professional writers to do what they do best: correct, clarify, and improve the writing of the draft" (p. 135).

Weiss does note that, if the design of a structured document is fully specified and frozen, "then the writing of the first draft is hardly like what is ordinarily thought of as 'writing' at all" (p. 134). Instead, "the writers are implementing, not creating" (p. 132). But this consequence is not said to be a problem. The only concern Weiss expresses is that, in the special case where one person single-handedly designs and writes an entire book, implementing one's own design might be boring:

> Developing documentation in this structured style reduces the interest of the first draft. Instead of being the most complicated, demanding, and fascinating part of documentation, writing the draft becomes the least interesting part. (Remember, most of the art and intellect has been shifted to the design phases.) Be warned, then, that even though books written by one person still benefit from the structured method, a

> professional writer will find it boring to carry out his or her own design and may be tempted to wander off onto artistic sidetracks (p. 132)

Weiss does not seem to regard this boredom as a serious problem because two pages later he says, "Even when there are only one or two authors, though, the benefits of the modular design are impressive" (p. 134).

The main benefit that structured documentation gives writers is greater likelihood of success in meeting the requirements of the job because the job is broken down into small, manageable pieces. It offers similar benefits to the document designer and to the writers' manager. For example, "designers are able to predict the size and cost of publications at the same time they are testing them for readability and accuracy" (pp. 50-51), and, by manipulating assignments, the manager can keep the project on track. "The more thoroughly enforced the modular design, the greater the opportunities to employ project management tools, honor budgets, and shorten the 'critical path' of the production" (p. 137). The result is greater efficiency, productivity, and profit.

The price of increased efficiency (doing the thing right) can be increased effectiveness (doing the right thing, or at least deciding what the right thing to do is). In the engineering approach, writers do surrender decision-making control to the document designer and to the writers' manager, who determine the content and schedule for the writers' work. Nevertheless, he declares, "the era of the artist-documentor is over" (p. 42). Far from lamenting its passing, he asserts that "the artist should yield to the engineer" (p. 41). In the final module of his book, Weiss argues that the issue for technical communicators is not ego but backbone: "Documentors have all the tools and techniques they need to produce outstanding work. What they often lack, however, is will and assertiveness. In the '90s, documentors should insist on certain standards of corporate conduct" (p. 228).

Of Document Databases, SGML, and Rhetorical Neutrality

Two years after the second edition of his book appeared, Weiss published an article, "Of Document Databases, SGML, and Rhetorical Neutrality," that questions the conclusion he arrived at in the book. He describes the new approach, in which writers create information units (variously referred to as articles, modules, chunks, or objects) and store them in a database. These units can be organized and linked in many combinations, but "all documents are 'virtual,' existing potentially in the database but not in any ordinary physical reality" (Weiss, 1993, p. 59).

It is important to note that the people who select the information units and combine them into documents are not the writers, but the readers. "The sender remains rhetorically neutral; the readers persuade and instruct themselves" (p. 59). Weiss describes this new communication culture as "exhilarating, and even somewhat visionary" (p. 59), at least from the readers' perspective. Readers have nearly endless freedom to determine the content and format of the document, responsibilities that have historically belonged to writers.

Further limiting the writers' ability to define the nature of the interaction with the readers is the growing acceptance of SGML, an ISO standard that uses a dictionary of document type definitions (DTDs) to specify the look and feel of documents.

Those constraints on the writer lead Weiss to raise some thoughtful questions about the future of information development:

> Can writing be an intellectually, professionally rewarding occupation if we break the traditional link between the writer and reader? What professional writer would choose a career of putting thousands of chunks of information into a database stew? What pleasure is left in the process without a sense of the audience, the feeling that I am "speaking" words of my choice, in an order of my choice, to a person I recognize? (p. 61)

In the section of this article entitled "Implications for the Professional Communicator," Weiss suggests that the engineering approach is a mixed blessing for technical communicators. On the one hand, "the new technologies will probably create thousands of new, highly skilled jobs for technical communicators" (p. 61). On the other hand, "most professional writers still want to affect their readers in predictable ways, to practice their hard-earned craft, and to point to some products or creations that are more actual than virtual. Absent these small satisfactions, it is hard to imagine anyone wanting to be a technical writer at all" (p. 61).

On balance, Weiss seems to suggest that, in surrendering both control over their work and the personal satisfaction provided by their role as protector and teacher, technical communicators have struck a Faustian bargain.

The Retreat from Usability

In his 1995 article "The Retreat from Usability: User Documentation in the Post-Usability Era," Weiss describes the brave new world in which the technical communicators' role has changed from protecting and teaching the reader to feeding information databases. Predictably, many technical communicators dislike this change because it threatens "the intellectual independence and pride of authorship that, for many, were the best parts of the job" (Weiss, 1995, p. 16).

When Weiss extrapolates the change, he sees a discouraging future in which the technical communicators' role becomes increasingly mechanical: "we will reuse existing document objects with new local data, so that even the creation of these information units will be reduced to filling in the blanks. And thus, eventually, more and more of the units will be generated automatically" (p. 16).

Egoless Writing

In "Egoless Writing: Improving Quality by Replacing Artistic Impulse with Engineering Discipline," Weiss uses more extreme language to express his deterministic view of the future. He blames the "ego-obliterating" (Weiss, 2002, p. 7) engineering approach for the loss of "basic self-esteem" (p. 8) and the "enfeeblement of the technical communicators" (p. 8). The technical communicators' loss of autonomy proves that their "talents and uniqueness are being devalued" (p. 9).

Ironically, the engineering approach has succeeded too well. As a result of that approach and related usability techniques, "the technical problems of user documentation have been largely solved" (p. 8). Therefore, "the industry now understands the problem of user support and can offer adequate (if not brilliant) documentation and training" (p. 8). Weiss cites Microsoft Press and Maran Communications for their ability to routinely turn out "publications that could be implemented as 'best practices' or even templates by all software writers" (p. 8).

Once a war has already been won, territory captured by heroic soldiers can be adequately patrolled by raw recruits. Similarly, in the aftermath of the usability struggle, "the current epoch of egoless technical communication could be staffed with people far less talented than the current cohort of North American technical communicators" (p. 9). In such an environment, Weiss concludes, technical communicators have only three choices:

> They can continue to resist egoless methods, swim against the tide in pursuit of personal satisfaction. Or, they can aspire to become the central cadre of inventors and designers, learning to delegate the "writing" to the miners, here or off shore. Or they can become "knowledge workers," people whose job is less the generation of information than helping people to find it, use, it, adapt it, or deal with the frustrations of a workplace rich in data but poor in insight (p. 9).

Conclusion

What should we make of Weiss's changed attitude toward the consequences of the engineering approach?

First, I conclude that his attitude changed because the world he describes changed during the last decade. Weiss's book is a major reason that most of the usability problems with technical documentation have been solved. When he wrote the book, he could not have declared victory. Eleven years later, he could. His book is the Before picture; "Egoless Writing" is the After picture.

Next, I note that beneath the change is a constant concern with the well-being of technical communicators. Weiss is not complaining about his own lot. He is not blowing off steam because of an isolated instance of unfair treatment of technical communicators. Nor is he trying to organize a union or foment a revolution. Instead, he is asking us to think seriously about the demoralizing and dehumanizing effects that a mechanistic, profit-oriented system has on sensitive, well-meaning individuals who do not want to be robots. We must applaud Weiss's moral principles as well as his intelligent analysis.

Nevertheless, technical writing is simply not art. Should we shed tears for the misplaced artist who would like the job to be something that it is not? At IBM, the technical writers' job title is software engineer. They do feed articles into an information database called the Information Center. Team leaders create a model (a hierarchy) of each set of articles, but the team

leaders do not provide the writers with a summary statement or exhibit for each article. The team leaders and the writers share responsibility for the information design. Within this structure, writers do more than connect the dots, and they have room to be creative. For example, writers can create advisors (programs that ask the user questions—for example, about installation and configuration—and, based on the user's responses, recommend an appropriate action), wizards (programs that perform actions for the user), finders (alternative views of reference information), and JavaScript.

My initial motivation for writing this article was to disprove Weiss's skepticism. Surely, I thought, the situation cannot be as bad as he says. But after weeks of trying, I have not found the evidence to refute him. The best I can do is to say, "Yeah, but..." and point to a few exceptions that only prove the rule. For example, he says, "Such technologies as SGML, HTML, and XML underscore the sudden abrupt change in the power of the technical communicator: from *perfect control over every element* of the page to provider of 'tagged input,'" a change that "*destroys any last vestige* of ego involvement by the author" (Weiss, 2002, p. 8; italics added). Perfect control? Over every element? What about the many corporate policies, local procedures, style guidelines, editing conventions, and grammar rules that prescribe a writer's choices? Many of the information developers at IBM would agree that SGML and XML are restrictive, but they get a degree of satisfaction from mastering the requirements to produce the correct results. That satisfaction may be more akin to the mental pleasure one gets from solving a puzzle than to the pride that comes from creating a work of art, but it meets their expectations.

A related concern is that Weiss sometimes makes generalizations about the computer industry as if it were monolithic. For example, he says that "the industry now understands the problem of user support" (Weiss, 2002, p. 8). No doubt a few people in the industry fully understand the problem and many people in the industry understand aspects of the problem, but I do not believe that the understanding is a binary, all-or-nothing condition. Thus, when he characterizes the industry as a whole—for example, when he states at the end of "Egoless Writing" that technical communicators have only three possible futures to choose among—we should bear in mind that the statement does not apply universally to all technical writers.

Although the three choices available to technical communicators at the end of "Egoless Writing" are presented as if they were all lose-lose alternatives, they are no different from the choices that most people face in any career field. The three choices are to be idealists who buck the system or invent their own system, to be leaders or decision-makers who master the system and succeed on its terms, or to be Dilberts who strive to cope with the system. What truck driver, clerk, teacher, athlete, doctor, lawyer, Indian chief—you name it—does not inhabit that same world? What makes technical communicators a special case?

With the exception of these few quibbles, I think that Weiss's interpretation of the consequences of the engineering approach to technical writing is correct. I know that his assessment of the past and present of technical writing is extremely insightful and fundamentally accurate. Although his vision of the future is not encouraging, I can find no reason to disagree. This fall, I will incorporate "Egoless Writing" into my Technical Communications course at the University of St. Thomas.

References

Weiss, E. (2002). Egoless writing: Improving quality by replacing artistic impulse with engineering discipline. *Journal of Computer Documentation*, 26 (1), 1-9.

Weiss, E. (1991). *How to Write Usable User Documentation* (2nd ed.). Phoenix: Oryx Press.

Weiss, E. (1993). Of document databases, SGML, and rhetorical neutrality. *IEEE Transactions on Professional Communication*, 36(2), 58-61.

Weiss, E. (1995). The retreat from usability: User documentation in the post-usability era. *Journal of Computer Documentation*, 19(1), 3-18.