Redefining Written Products with WWW Documentation: A Study of the Publication Process at a Computer Company

Gary Beason
Purdue University

SUMMARY
This article examines how writers used the World Wide Web to create and change documents, and how changing documentation challenged the production models and processes at a supercomputing company.

Analysis of electronic writing would be worthwhile to both technical communication researchers and practitioners, particularly when that analysis shows how a specific technology can simultaneously improve and hinder work. For instance, Shoshana Zuboff examined how computer technology has changed work environments both positively and negatively: even though a network improved the efficiency of office communications, employees found that the computers increased barriers to personal communication and even created an unfriendly environment (1988).

For almost 2 years, I have been conducting similar research at a supercomputer company, while working there as a technical writer. As part of this research, I asked the question, “What is a final product for documentation published on the World Wide Web?” and examined this specific aspect of writing to see how it affected a particular writing culture. In this study, I have tried to follow up in a specific setting on the idea that Jay David Bolter (1991) and others have proposed: “In place of the static pages of the printed book, the electronic book maintains text as a fluid network of verbal elements” (p. 5). My study shows that online writing allowed writers to take advantage of and to redefine a dynamic “written product,” but not without repercussions. My research further analyzes the notion of written product and the consequent changes in production and writing processes. For example, defining the “written product” precisely is complicated by the multiple file formats (PostScript, MIF, and HTML) and the maintenance of the World Wide Web site for a collection of documents:

- Boundaries between individual books began to disappear as documentation became more of a searchable database.
- Tracking and distinguishing among production

This article has been peer reviewed.
and "post-production" files and copies became increasingly complex.

The study also examines how the ongoing redefinition of written products has affected daily writing decisions and practices.

**Methodology and Project Background**

I had selected Kent Computer Company (a pseudonym) as a site to conduct research for my dissertation as a participant-observer at a company where writers were beginning to produce online documentation. I began my research in May 1994. I had applied for an intern position with Kent so that I could work while conducting research. I had explicit agreements with and cooperation from the writing managers and their management for this project.

After I started work at Kent, almost all the writers knew that I was there to conduct research on their uses of computer technologies. In March 1995, I was offered a full-time position, which I accepted. Although Kent did not pay for the research reported here, my agreement to work there included provisions for released time so that I could conduct and analyze the findings from my research.

This particular study of World Wide Web use arose from the opportunity in February 1995 to work with hardware writers to put documentation for I/O (input/output) devices (for example, disk drives, controller cards, network connections) onto the World Wide Web. My assigned role in this group of writers was to help the others learn about HTML, tools, and other World Wide Web issues. This project began with three writers: Dave, Wayne (both pseudonyms), and me.

During this time, I collected data from

- personal observations and experiences
- informal interviews of the writing group managers, the I/O documentation writers, and other writers
- recorded meetings and discussions
- text analysis

Dave was new to the writing group, though he was one of the first Kent employees and had worked on the manufacturing floor and in the field for several years. He was brought into the writing group in October 1994 because of his familiarity with Kent products and customers, but he did not have an extensive writing background. He assumed sole responsibility for I/O documentation in November 1995. Wayne is an experienced technical writer who has worked for Kent for several years, having previously written documentation for the military and for defense contractors. These writers and I worked together from February to September 1995 on the I/O documentation for the Web; during this time, we worked with HTML and prepared documentation for the Web. After that time, I participated less as a writer and more as a peer reviewer. Because of rising hardware documentation demands elsewhere, Wayne left the I/O Web documentation group in November 1995.

---

**Kent competes in a rapidly changing field where technological changes often drive organizational changes. For example, Kent recently set a goal to improve its computer performance 200-fold by the year 2000.**

---

Kent Computer Company is a supercomputer manufacturer in the Southwest that employs some 800 people worldwide. Kent has a flat structure: as few as three levels of management separate the company president and a hardware developer or writer. Kent competes in a rapidly changing field where technological changes often drive organizational changes. For example, Kent recently set a goal to improve its computer performance 200-fold by the year 2000. In such a market, where customer expectations change almost as rapidly as the technology, Kent emphasizes the ability to change and learn quickly. To do this, Kent has developed two important cultural elements:

1. Kent works with a rapid release, or iterative, production model. Multiple versions of computer products are developed and sold, the assumption being that Kent can keep up with a changing market and improve its products with quick changes to the product and feedback from customers.

2. Kent is a field-based company. Sales representatives and field engineers work extensively with customers not only to help with installation and maintenance, but also to provide feedback and direction for future product versions.

As a by-product of this culture, Kent also values informality because it grants employees autonomy and the opportunity to work quickly. Hence,
knowledge at Kent is widely distributed and localized, often associated with a particular person.

The I/O Web project began with discussions among Wayne (who was writing I/O documentation at the time), I/O developers, and the writing managers. The problem with the I/O documentation at that time was that it was not delivered to the customers when they needed it. In part, this delay owed to the quick release of I/O products (one every 2 or 3 months), the 2 weeks required to print, an over-taxed technical illustrator, and the information-gathering process (which required finding out all the relevant part numbers, getting installation procedures and product specifications, consolidating technical review comments, and keeping up with the latest bug reports and fixes).

The Web was chosen as an appropriate medium for several reasons that reduced dependency and improved on-time delivery:

1. Kent needed to change from a “push” system (in which Kent shipped documents to customers, sometimes even before they had the products) to a “pull” system (in which the audience had to retrieve what they wanted). Because Kent “pushed” its documentation, customers sometimes lost their documentation or were inundated with new information.

2. Publishing on the Web offered a quicker means of distributing documentation without needing to work with a printing company.

3. Writers could scan photographs of the products instead of using line art, thus reducing the dependency on a technical illustrator.

4. Field engineers were the primary documentation users; customers installing devices themselves were the exception. All field engineers had access to the Kent Web servers.

Redefining Writing and Product at Kent

The idea that a book is actually several books, or versions, or that even a “final” product changes is not a new insight at Kent: the written product is not only updated from release to release but is also restructured. However, these multiple instances of the same book have been successive, like editions. Multiple versions of Web documents, on the other hand, are simultaneous and demand explicit recognition of this fact and of the possible problems that this fact causes for writing usable documentation.

The idea that a book is actually several books, or versions, or that even a “final” product changes is not a new insight at Kent: the written product is not only updated from release to release but is also restructured.

In the following sections, I show how Web documentation is dynamic and how file location, Web writing tools, a Web production process, and conceptual rethinking of Kent books have made an explicit definition of the “final product” ambiguous and have multiplied the document into a set of documents.

Redefining through File Location

A written product, particularly for the Web, is not only the words and graphics that appear on the page; it is also the file source and structure. As we worked on documents for the Web, we changed the location of files to share a work space. Whereas this shared space consolidated formerly dispersed private work areas, it also created space for multiple versions of the documentation. We recognized the existence of multiple versions of the same document and tried to develop tools and procedures to manage the documents.

The writers at Kent work in a networked X Windows environment with FrameMaker 4.0, a publishing tool that allows word processing, page designing, illustrating, and book building. Most writers use “diskless” workstations booted from a remote machine. Because of the different architectures and projects, most writers work on several machines. When this project first began, writers had common directories that could be mounted from most of the machines where writers might work. Primarily, writers had two directories: one for work in progress (/work) and another for documents that were completed and had been published (/publish). These directories are illustrated in Figure 1.

Even though they were encouraged not to, writers sometimes worked in other directories, even their home directories (sometimes because the /work and /publish file systems were full or nearly full), as shown in Figure 1 with Wayne’s /book1 directory. The I/O Web team at first worked in several
directories, which had previously been created for a print production process.

With FrameMaker, each book is actually a collection of many individual files; generally, these files are located in a unique directory. The /publish directory was intended to be an archive for books that had been published and sold to customers: these books were more or less "frozen" in that these copies were not edited further. The /work directory was for documents currently in progress: they were the "living" copies that writers modified. The /home directory (for example, /home /wayne) was intended only for UNIX and X Windows configuration files, electronic mail, and so forth—not for storing book files. However, no part of the production process required that the book files be located in /publish or /work. To further distribute the book files, the /work and /publish directories were directories located on a network drive, accessible from various writer workstations.

This use of the /work and /publish directories reflects a print mentality. The idea that the books were located under a writer's directory in /work or /publish also reflects the Kent print model of writing. Because printed books had only "soft" links—references or quotes—to other books, the writers worked individually on their books. As the writing manager remarked, writers feel an ownership of the books, each writer thinking this was her or his book, even if the book also involved an illustrator, engineers, users, support personnel, the printer, an editor, other writers, and sometimes even marketing personnel. In Kent's print production, writers "owned" the responsibility for books: the writers had to manage the schedule, get input from peers and engineers, and request illustrations and reviews. The use of /work and /publish reflected, perhaps even promoted, this relation between writer and book. Writing for the Web, especially by a group of writers, required that we change this model.

Dave, Wayne, and I quickly realized that, even if we were to work in the "proper" directories, we would still each be working in different places. This situation would cause confusion:

- Who has the latest draft?
- What changes have possibly been lost?
- Who has permissions to write to which files?
- How will we combine two versions of a collaboratively written document?

It would also make creating cross-references difficult because FrameMaker opens and edits the document to which a cross reference is made when a cross reference is created.

To prevent these problems, we developed a common work area and changed our write /read permissions. The new work area, while reducing the number of copies of a file, did allow for the existence of multiple versions of the file in FrameMaker MIF format, in HTML, and in PostScript. In short, our authoring and conversion tools defined a document as document versions (as explained in the section "Redefining through Writing Tools" below). We also considered how to minimize the needless repetition of information by creating shared files, such as graphics and a preface, that would reside in /html (in HTML format) and /templates (as FrameMaker files). Figure 2 shows the organization of our shared work area.

The idea was that the /frame files should match the content of the corresponding /html files. One of the most significant challenges we faced was maintaining this correspondence, which I discuss in the "Redefining through Production Process" section.

As we responded to these concerns, we moved away from the print model and came closer to adopting a model resembling the software development process. In software development, developers work on parts of a code, but they have a "work tree" into which their individually written pieces of code are merged together as a single source. Using software like ClearCase to control versions and
processes, software developers tightly control the parts and the whole of an application’s code: there is a formal check-in, check-out, and freezing of the code.

By recognizing the need for process change and implementing a common work space, we were adapting to the needs of a new publishing process and redefining the written product at Kent: it was no longer a collection of individual books but a changing “database” of information. This change of file space also had an attitudinal impact: we consciously tried to stop using the term “book,” an effort I describe in more detail below, because we realized that the “book” concept no longer adequately described our work.

Redefining through Writing Tools

Corresponding to the shared work space that we had created, the writing tools for online production also multiplied. For print, we had primarily worked with FrameMaker; for the Web, we continued to use FrameMaker and several new tools. These tools sometimes produced new file formats for the same information that had previously been written in FrameMaker; other tools were used to check the links and to create a hypertext table of contents:

- **FrameMaker** was used to compose the documentation. It was a writing tool that we were all familiar with and could easily edit without having to learn all the HTML tags.
- **Mifmucker** was used to convert the FrameMaker file into a document with HTML tags.
- **html.map** was a file that defined the character and paragraph tags in FrameMaker as HTML tags.
- **xpaint** was used to supplement the FrameMaker graphic tools.
- **Photoshop** was used to edit scanned images.

A significant challenge was to keep the FrameMaker and HTML files in synch. We typically had to write the document in FrameMaker, convert it to HTML, and view it in a Web browser. Finding areas for revision, we would return to the FrameMaker source and edit it, repeating the conversion process again. The justification for using FrameMaker rather than an HTML authoring tool was that we still had to produce a printed version of the document. Although we could have printed pages from the Web, there were two obstacles: 1) we could not print the whole book (unless there was also a PostScript version) but only a “page,” and 2) we could not print a copy that met even minimal quality requirements for printed documents (for example, page numbering). Rather than try to create separate HTML and FrameMaker versions, we attempted at first to produce both from a single source. However, as we worked with Mifmucker, we found that we had to change some of the HTML tags. The FrameMaker
and HTML versions had to be formatted somewhat differently.

Redefining through Production Process

The previous sections have primarily shown how an online document is a set of documents or versions of the same document. The production process—how the information was published—helped define "final products" as documents in flux. We wanted a flexible document—one that we could publish on the Web as the information, even if preliminary, became available. Nonetheless, we expected to produce a

We wanted a flexible document—one that we could publish on the Web as the information, even if preliminary, became available.

completed, whole product. Our Web production process, however, helped keep the document in flux as well as define several versions. Because we had difficulty deciding what version to "freeze" and because we published and delivered the product ourselves, we gave up procedures that defined both a change of ownership and an end to writing. In short, we continued to revise because we could. This flux prompted significant challenges to the writers and readers.

- How would the audience know to check for new information?
- How would the audience know what information had changed?
- If the information could be revised after publication, when should the information first be published on the Web?
- How are field engineers notified of the changes?
- How do writers know when to stop writing?
- How are Web documents "frozen"?
- How are customers able to order documents if they do not know about the I/O Web site?
- How long should information be kept on the I/O Web site?

For printed books, writers followed a procedure that defined both when a book was considered final and who assumed responsibility for the distribution of the books.

1. Write the book in FrameMaker (presumably in /work).
2. Request a book cover.
4. Submit the book for peer and editorial reviews.
5. Submit the book and cover for a quality check.
6. Copy the book files to a product control directory by part number.
7. Create a PostScript version of the book.
8. Copy the PostScript version to a digital tape.
9. Sign off on the book, and give the tape and a printed copy of the book and the cover to the printer.

This is a formal process that writers were very familiar with. Still, in this model, the book exists in multiple versions (for example, draft, revised, and approved) and even in different media formats: printed as well as PostScript and FrameMaker binaries. However, the production process has two relevant points: The writer turns the "book" over to the printer and to product control for production and distribution, and by doing so, the "final" or published version of the documentation is defined.

Publishing on the Web changed this production process, mainly because the writers also control the delivery of the product: there was no need to hand the documentation to another party. Also, the documentation did not have a part number because of the complexities of determining what the final product was. Would it be the FrameMaker source? If so, would the HTML conversion and changes be considered simply "post-processing" that did not affect the content? In part because of the desire to allow flexibility of documentation content, this option was not possible because assigning the documentation a part number and turning over source to product control would mean that the documentation product could not subsequently change. At the outset, the writers and managers recognized the importance of never "freezing" the written product, but we decided to proceed without participating in the print production process until an online model could be developed. However, very recently (at the time this article was being written), the I/O writers decided to align their process more with the print production process by writing in FrameMaker and "freezing" the FrameMaker source.

To answer some of these questions, we developed (and continue to revise) the following production process.
1. Write the book in FrameMaker (presumably in /frame).
2. Convert to HTML and review the Web pages for content, style, consistency with other Web pages, graphics, and hyperlinks.
3. Submit the Web pages for a technical review.
4. Submit the Web pages for peer and editorial reviews.
5. Submit the Web pages for a quality check.
6. Copy the HTML and PostScript files to the technical support Web server.
7. Notify I/O developers, field engineers, support personnel, and sales of the changes.

Each time we change the content or add links, we repeat these steps. The printed books, though, go through the process only once. It seems that this production process, as much as the quality of the book, help define what is and is not a final product.

This production process answers some of the questions raised earlier. Other questions are answered differently, sometimes changing from case to case. To alert customers to changes, the I/O writers decided to include a summary of changes to each document: when the reader first selects a document title from the I/O documentation home page, a “time stamp” lets readers know the date of the latest version. Also, we provide readers a link to a revision history of the document, which notes any information, or content, changes. (We briefly considered an analogy to a list of “bug fixes” that would include navigational or behavioral changes. We decided against this approach, given the volume of changes and the risk of de-emphasizing the content changes.)

---

**To alert customers to changes, the I/O writers decided to include a summary of changes to each document: when the reader first selects a document title from the I/O documentation home page, a “time stamp” lets readers know the date of the latest version.**

---

We spent a great deal of time discussing exactly when to release even partial documentation. We first redefined “partial” to mean inclusion of certain sections (such as installation), not unverified information or omitted steps. We were less specific about this issue because, depending on the product and the audience, we might publish parts of the I/O documentation on the Web at different times. This approach was especially needed for “preferred customers” who participate in a partnership to test pre-production models and to offer design feedback. The I/O writers work with the developers to list the products and their beta and production release dates.

---

**Redefining through Concept**

Even though Wayne knew little about the specifics of online technologies, he had an idea of a documentation project that differed greatly from the books that the writers were accustomed to producing. He and others saw the need for an information database about I/O products: rather than searching through the entire collection of Web pages, a user should be able to specify her or his machine system and I/O device, retrieving only the relevant information. In fact, as we worked on the I/O Web documents, Wayne repeatedly emphasized the need to avoid calling these “books” and to call them something else, an alternative that we did not agree on at the time. (Eventually, though, we did achieve Wayne’s goal in that the writers and developers called it the I/O Web.) This idea of a “nonbook” surfaced in casual discussions and formal reviews of templates and information.

Wayne was adamant about this change because he saw that the boundaries among books would disappear as readers followed links from, for example, a SCSI disk installation procedure to information about the operating system. He also saw the writers becoming more responsible for each other’s work. For example, he saw the stylistic and organizational differences between hardware and software documentation that could cause reading problems if the printed books were converted to online form and linked seamlessly online. We also began to change our ideas of ownership. Although Dave had written the first two books that we converted to HTML, I felt as much ownership of those documents as Dave did, for I had also implemented some of the organizational changes that we had all agreed on.

Calling the information a “Web” and not a “book” was simply a matter of terminology: Wayne also saw the need for us to change the way we write. We all agreed that this database model meant that we have to collaborate more and that several writers might be responsible for what once was a single printed book.
In fact, for the first three I/O products for which we produced Web documentation, Wayne, Dave, and I each worked on all three pieces of documentation: sometimes Wayne worked on the installation chapter, while Dave wrote and revised the other chapters, and I converted the FrameMaker files to HTML and "tweaked" the HTML. However, we did switch roles and responsibilities, especially as I was preparing to leave the team and begin work with another product team.

As a result of this conscious redefinition of the documentation from book to Web, we tried to find ways to work together and learn how to write online. For example, we changed the peer review process so that we met and made group decisions about what to change in the documentation. For print, a writer would typically depend on written comments from other writers. We also made conscious efforts to develop some style conventions for naming figures, writing numbered steps, and using consistent terminology throughout the documentation. Much of this consensus building was to create a consistent style for online use. For example, we had to be sure that if we called a product by its marketing name, we did not arbitrarily start using the part or order number. Some of these changes were necessary because of the format limitations imposed by HTML. We also began to define conventions for naming files so that they had a pattern; this convention was necessary to make creating links to other files easy and predictable. It also allowed us to develop templates specific to the I/O documentation. For example, we created a generic installation chapter that was typical for disk drives and another template generic for controllers. In these templates, we highlighted parts that were likely to change.

In these meetings (sometimes lasting four or six hours), we were redefining I/O books in very specific ways. Sometimes, we fought the inertia of information and format: frequently, we asked, "Why is this information here?" We found that the design and organization of the online books were modeled after previous printed I/O books. Using existing books as models was a frequent strategy in a situation where the writers had little access to customers, worked under very short deadlines, and were, as in Dave's case, new to the group. In talking about the organization of the online I/O books, we were addressing two issues:

1. The rhetorical questions (Who is the audience, what do they need to do, and what information is essential?)
2. The presentation of the material in chunks for online reading

Wayne wanted to address the first issue in part because he wanted more conformity among the I/O books and more agreement between himself and Dave, the two principal I/O writers. Wayne was also interested in stylistic conformity, for example, using the same vocabulary and references. Wayne was concerned that, with all the different part numbers, marketing names, and vendor numbers for every product, readers would be confused by inconsistent references.

Chunking and organizing the information was the dominant theme in these meetings. We recognized that chunking the information for presentation on users' screens was important: we had evaluated Web pages that were several screens long as too cumbersome for our readers. Procedures were particularly challenging because we agreed that including both input and output during software installation helped the audience keep track of where they were in the script. Yet some installation scripts included 20 or more steps. To provide screen samples sometimes made the node four or five screens long. We had defined usable Web page length based on both screen "fill" and unity of information, but procedures for installation scripts challenged that.

As a result of this conscious redefining of the documentation from book to Web, we tried to find ways to work together and learn how to write online. For example, we changed the peer review process so that we met and made group decisions about what to change in the documentation.
definition. We decided that unity took precedence over page length because, if users read the documentation online (as opposed to reading a version printed from the Web browser), they would be confused by having to connect to another node in the middle of procedures.

Also, we debated the value of procedural and conceptual information. For example, Dave had included in a manual for a SCSI device controller some background material about SCSI devices, such as the general interface architecture. He justified this approach because of his own experience installing products; he wanted to understand how they worked, not just how to install them, especially if he had to troubleshoot the products. Wayne, however, felt that this was unnecessary information and wanted a singular focus for the online documents—how to install and test a product. Dave argued that, if anything, writing online allowed the inclusion of non-essential, though helpful, information.

This willingness to redefine I/O documentation as something other than a book continues. The I/O books were often short, typically 30–60 pages (compared with the 250 or more page reference guides, for instance). Still, they had the look and feel of other Kent books: each had a glossary, table of contents, list of figures, list of tables, and preface that explained the stylistic conventions of the book. The online versions of this documentation, however, did not require separate prefaces: there was more opportunity for shared information. Still, our initial online documents were longer than our later online documentation, with each product’s documentation having 20–40 nodes of information. Lately, however, Dave has been working alone on the I/O documentation, changing the format even more. Currently, there are fewer than 10 nodes in a “book,” and the information changes so that were it printed, it would look more like a release notice than a book.

**Considerations for Changing Products**

I have tried to show how “final product” is actually multiple documents or versions, and how documentation on the Web is in flux. Much of this ambiguity was intentional so that we could better serve customers as part of Kent’s rapid release development process. I have also noted some of the challenges of dynamic documentation. We continue to examine issues, however.

- What graphics, text, and other information can several “books” share?
- How can we create increasingly collaborative “books” or databases (with many levels of hypertext) that can easily be maintained yet can also work as printed documentation?
- How can we write documents so that they are not focused on a product but on tasks, especially for customer sites that use combinations of different products?
- To what extent should the written product continue to be flexible? How do we define a “frozen” document?
- How can product developers be persuaded to accommodate documentation needs?
- To what extent should we continue to consider the latest Web developments and try to implement them?
- How do we develop style guides that keep pace with Web changes?

Some of our responses worked; others failed, partly because of the audience’s reading habits and because of changes in the information. More specifically, by putting documents on the Web, we successfully switched from a “push” delivery system to a “pull” system. Even though we “pushed” out news of information changes, users still needed to retrieve it. As I have found with my own use of the Web, I am less likely to use the Web to check for information as I once was. So may be the case with our audience.

Our most significant regret was the production process, for we found that, because I/O documents used a different production process from other Kent documents, I/O documents were not given the same focus or resources that others were given. The managers admitted this problem and are currently working with the I/O writers to design a formal strategy that states the goals, requested resources, processes, risks, and dependencies.

For the most part, the navigation and organization of the Web documents succeeded (insofar as it met our goals and inasmuch as we heard from users). Still, the presentation and behavior of the Web documentation is continuously questioned, in part because of other sources:

- Kent technology
- Web technology (HTML enhancements, browser enhancements and format developments—such as Adobe’s Portable Document Format)
• personnel changes
• other online documentation efforts at Kent (for example, SGML and PDF)
• other Kent Web servers
• Kent’s competition’s Web servers

These sources currently affect the development of the I/O documentation in too many ways to describe here, but the point is that the I/O Web documents are not isolated from other online efforts.

As I have shown, publishing on the Web redefined Kent I/O documentation. Much of the change was planned, even desired. Yet Dave, Wayne, the managers, the I/O developers, and I recognized that the I/O documentation needs further change. Even though we were told that all the field engineers had access to the I/O Web, we heard from two of them that this was not always the case. (Even though we sent out questionnaires and prodded for feedback, we were not able to get consistent feedback. In fact, learning more about how our audiences use our documents is a problem across the documentation group.) It is likely that the audience’s reading habits are changing; they are not necessarily using the Web as a reading environment but as a delivery or storage tool, printing out the PostScript versions.

As these solutions are pursued, though, a fundamental conflict is apparent: the desire to “freeze” knowledge that is constantly changing. Kent has identified its main skill as designing, revising, and producing computer technology very quickly. Kent moved from a vector-based architecture to RISC machines, from the high-end scientific market to the high-end server market, from the expert who wanted a highly specialized and configured machine to the system administrator who wanted a “shrink-wrapped” product.

Given this situation, writing of any sort, let alone production of a “final product,” can only attempt to present a snapshot of the technology, of a knowledge that is revised (with little hyperbole) daily. For example, project teams at Kent work by a rapid release schedule: produce a “patch” or an enhancement of the product every quarter and complete a major product release once a year. Such a model may seem to minimize the opportunity for a quality product; instead, the value is on the ability to revise and learn, revise and learn.

The challenge to the writer, then, is to create an interface, an interpretation, an explanation of such changing product. To represent this knowledge, the written product has to change as well. But that change cannot be unplanned, unexamined, or arbitrary. Writing is an act of capture, of producing a still life. This is the tension between the written product and the developing knowledge. And this is the irony: “product” suggests an outcome, some finality, but at Kent, the written and technological products are both outcomes and beginnings.

REFERENCES