

SUMMARY

- ◆ Describes the usefulness of applying the principles of business process re-engineering to online documentation
- ◆ Presents the benefits of user-centered design, iterative user and task feedback, and an interdisciplinary design team

Re-engineering Online Documentation: Designing Examples-based Online Support Systems

MARTIN D. TOMASI AND BRAD MEHLENBACHER

INTRODUCTION

In this article, we argue that three developments in the design and evaluation of online support systems are reshaping traditional design efforts. First, we begin by questioning the long-term benefits of using linear, paper-based approaches to the design of online support systems that assist users in learning sophisticated computer software to accomplish complex tasks; instead, borrowing from principles of re-engineering, we outline an alternative user-centered approach to online documentation design that draws on the metaphor of information as tool. Second, although many principles of effective minimalist design pervade current practice in designing support materials (see Carroll 1990; van der Meij and Carroll 1995), we argue for the design of materials that support both procedural instruction and conceptual, performance-based learning. Third, tailoring support materials (in effect, re-engineering them) to aid customer performance through example-based interaction represents a potentially useful approach to designing effective online documentation, especially for supporting software where experienced users engage in non-linear task sequences to solve complex problems.

Despite the innovations produced by computers and information technology, perhaps no fact is as telling as the mere 1 percent productivity gain attributed to the use of computing technologies (Constant 1993). The large investments in computers and related technology that started in the late 1960s have run over \$100 billion annually since 1991, yet since the early 1970s, productivity gains have averaged less than 1 percent in the U.S. (Landauer 1995). It seems that although organizations have faced significant challenges in their struggle to restructure and retool their businesses for the 21st century, their attempts at improving productivity and performance through investments in information technology have often delivered disappointing results.

This small increase in productivity results from the fact that companies still tend to use technology to automate old ways of doing business, and so existing processes are left intact, and computers are used simply to increase their speed (Hammer 1990; Turnage 1990). The lack of substantive productivity increases results from businesses' incorrect implementation of new technology.

To address this "productivity gap," many companies have tried to start over and "re-engineer" their businesses. Organizations have found that simply speeding up a business process does not address its inherent performance deficiencies. Re-engineering—in the classic, business school sense of the term—refers to the use of modern information technology to radically redesign business processes in an attempt to realize substantial improvements in performance (Hammer and Champy 1989).

Business processes and structures are often outmoded and obsolete or have not kept pace with changes in technology and business objectives. Most of the organizational structures, work flows, and control mechanisms used by businesses came of age before organizations considered computer automation, and these processes were designed to support efficiency and control. New technology and new business objectives have left companies yearning for improvements in the performance of their processes, whether in speed, service, or quality (Hammer 1990).

Re-engineering online documentation

Authors of online documentation face the same problem that companies face in improving the performance of users, yet the concepts of re-engineering are not often applied to

Manuscript received 16 August 1998; revised 13 October 1998; accepted 15 October 1998.

online documentation development; rather, online documentation is frequently seen as a simple extension of the printed manual (Brockmann 1990; Horton 1991; Reece and Scheiber 1993) and researchers are still compelled to compare and contrast paper versus online documentation to justify the rapid contemporary movement of documents online (Barnett 1998; Henke 1998; Smart, DeTienne, and Whiting 1998).

Minimalist design maintains that designing systems that enhance the user's ability to quickly accomplish a task is a critical goal for online documentation designers (Carroll 1990), yet this focus may limit designers to providing skills-based training rather than imagining how to support long-term user performance. Perhaps not surprisingly, Gery (1991) asserts that many online documentation systems have "done little more than speed information searches" (p. 23); not only do users continue to have problems with their application systems, but they also have problems with the accompanying online help (Duffy, Palmer, and Mehlenbacher 1993; Pratt 1998). For this reason, van Dam (1987) warns, "Don't copy old bad habits; think about new organizations, new ways of doing things, and take advantage of this medium" (p. 3).

Many hypertext efforts have largely ignored this call to action and simply "automate" the printed documentation of a system or application by moving paper-based manuals online. In doing so, they do not address the need to integrate information with task performance (Norman 1993); instead, they simply transfer inefficiencies inherent to paper documentation online. The tasks presented in the online document must match the performance of the user. And at the same time that some sources project that workers will become increasingly more expensive because of the need to train and retrain them (Bibus 1990), the demands to deliver higher productivity affect the developers of online documentation as well.

The tasks presented in the online document must match the performance of the user.

Unfortunately, just as researchers have shown that performance improvements do not always occur where computing technology is applied to business environments (Turnage 1990), researchers have highlighted the failure of online information to aid user task performance (Gery 1991; Hughes 1997; Norman 1993). Business processes and online documentation share more than these parallel results of technology implementation: often online information supports computer applications using methods that

are based on old business models. One might conclude that organizations that have failed to take advantage of new processes in implementing their business systems have also not implemented online information to take full advantage of the capabilities afforded by hypermedia.

But establishing what it means to take full advantage of hypermedia is complicated by its historical interaction with print-based development methods. Thus, emerging research on media differences plays an important role in identifying both the strengths and the weaknesses of the alternative media:

- ◆ Online libraries of documents allow users to locate the appropriate online book more quickly, but information within individual print manuals is often, though not always, located more quickly than in online manuals (Barnett 1998; Landauer 1995), encouraging a careful examination of the reading strategies that users bring to each medium (Kaindl and Snaprud 1991; Mings, Geisler, and Rogers 1993; compare Redish 1988).
- ◆ Hardcopy books use conventions that are more familiar to users and rarely require "system" knowledge or special skills of navigation routinely required by online documents (Mitterer, Lungsu, Carey, and Nonnecke 1992; Reece and Scheiber 1993; Zimmerman, Tipton, Bilsing, and Green 1993).
- ◆ Hardcopy books facilitate user tasks and goals in ways that are sometimes different from online documentation, although the research often conflicts on which medium is preferable for which user situation (Barnett 1998; Grice 1991). Measuring user performance and analyzing results from usability testing, then, may not be easily comparable (Grice and Ridgway 1993; Mehlenbacher 1993).
- ◆ Online documentation exhibits physical and rhetorical characteristics that differ from hardcopy books, in terms of resolution, display area, aspect ratio, presence, organization, navigation, and contextual structuring (Selber, Johnson-Eilola, and Mehlenbacher 1997; Spyridakis and Isakson 1991).

Whether or not distinct rhetorical differences exist between online and print documentation, most researchers agree that we must assess how well they support the tasks, activities, and overall performance of their users. Clearly a re-engineering perspective toward online documentation can inform such an assessment, given its goal of redesigning work processes and products to heighten user performance and productivity. Importantly, these potential differences raise the possibility that new ways of thinking about the development of online documentation are in order. And these differences sometimes undermine deeply ingrained habits and rhetorical strategies, a fact that suggests that we may need to explore alternative methodologies for developing online documentation.

Reflecting on the potential of re-engineering

The basic concept on which this article builds is manifested in van Dam's (1987) urging designers to think of "new ways of doing things." The essence of business re-engineering involves completely starting over and breaking away from the rules and notions that underlie existing activities. Hammer and Champy (1989) argue that to achieve the performance improvements required for businesses to remain competitive, it matters little how work was previously performed. Instead they contend that, given the demands of global markets and the power of current technologies, systems can be organized to create significant benefits.

Companies that embarked on the "extreme measures" of re-engineering realized results that, as Hammer (1996) asserts, "succeeded far beyond anyone's expectations" (p. xii). Throughout the early 1990s, many organizations achieved the "dramatic improvements in performance" that re-engineering promised (Champy 1995). Businesses made significant headway in overcoming process problems; unnecessary tasks were eliminated; tasks were either combined or reordered; and improvements in speed, accuracy, flexibility, quality, service, and cost savings were realized.

Similar to the re-engineering of business processes presented by Champy (1995) and Hammer (1996), other disciplines might likewise face crises of change. In the pursuit of solutions to online documentation problems, technical communicators have not yet begun to re-examine old approaches; we are applying sophisticated technology to old frameworks, old processes, and paper-based methods of design. With online information, the challenge to existing rules is not to be found in the technology itself, but in what we are expected to do with it and in the processes we use to create it. Given performance expectations, our current ways of developing online documentation may be limited, incomplete, and ripe for re-engineering.

A CASE STUDY OF A NEW APPROACH TO ONLINE DOCUMENTATION

To understand how the hypermedia design process can shift from adapting paper-based methodologies to applying re-engineering principles to re-think the process and the product, it is difficult to find model design situations to examine. The concept of re-engineering does not provide specific development models that can be applied to hypermedia, and as a result, few design principles currently exist. Instead, approaching the task requires a "fundamental rethinking and radical redesign" of the processes used to produce online information specific to an individual case (Hammer and Champy 1989). The best way to understand how re-engineering principles can be applied to hypermedia development, then, is to examine an actual application in a case study, a recent online documentation project conducted by a large software firm (referred to here by the pseudonym "SoftCo").

The concept of re-engineering does not provide specific development models that can be applied to hypermedia, and as a result, few design principles currently exist.

The business problem

SoftCo produces a software package called *App Builder*, a tool that helps programmers develop complex software applications (similar to other tools such as Microsoft's *Visual Basic*). *App Builder* has a typical graphical user interface that supports standard ease-of-use features such as point-and-click and drag-and-drop. Developers construct interfaces for their applications by selecting ready-to-use objects from a palette and dropping them on a window. Programming logic is then added to the window to enable the objects to communicate with each other.

The integrated development environment of *App Builder* can generate some code and provide other user assistance features to developers, but at some point, a knowledge of software development is required. The users of *App Builder* are savvy software developers who may understand object-oriented programming techniques but need to learn specific concepts to use *App Builder*. In 1996, a small team of SoftCo technical writers set out to create user support information in any form that would help applications developers quickly become more productive with *App Builder*.

SoftCo had had previous success with providing "examples books" to users of their other products. An example in one of these books consisted of computer code listings with accompanying documentation in marginal notes, while more complicated concepts were explained in greater detail through related chapters. Examples books dealt with linear concepts, and programmers who read them could construct examples by entering code line by line. The basic use of an examples book, then, was to read the text, copy the code out of the book, then watch the example execute. There was a single path to successful implementation. A book, with its inherently linear features, proved appropriate for this type of information. The information development tasks required to produce an examples book roughly took the following form:

- ◆ Review the product and define information requirements.
- ◆ Identify examples to show certain features of the product.
- ◆ Write the information for each example.

- ◆ Test the examples for clarity and completeness.
- ◆ Revise and rewrite as necessary.
- ◆ Publish the book.

SoftCo's information development group faced a different challenge with *App Builder*. Examples in *App Builder* were small applications that illustrated a concept, a development technique, or a method of solving a development problem. The use of the product required object-oriented development techniques, a fact that meant that developers using *App Builder* could approach programming tasks in a variety of ways. While software developers must interact extensively with the tool to set interface attributes and other application parameters, the course of action that a developer takes depends on a complex set of related tasks. Finally, the concepts surrounding the *App Builder* product were not concrete, so they too depended on situations and events within the development process.

All of these issues seemed to preclude the use of a book as the medium for presenting examples; there was no single, linear path to present to users, and the interaction between the primary application and the secondary support materials needed to be direct and flexible. Thus, the project team focused on producing an online product.

Many examples-based programming books (from SoftCo and other publishers) also include a CD-ROM that contains a simple collection of code that readers of the book can use. The code on the CD-ROM is identical to the code that is contained in the book, with the exception that the CD-ROM code is ready to use without requiring additional keyboarding time. The team, with its mandate to "help applications developers quickly become more productive with *App Builder*," did not feel that this method of putting examples online would substantially improve user performance.

After several discussions, the team agreed that the online medium could offer more than just example code on a CD-ROM. To achieve user performance improvements, the online information could be integrated into the *App Builder* product. Therefore, the results of initial planning for the *App Builder* examples product were that

- ◆ A linear presentation of a typical examples book would not work with the *App Builder* product.
- ◆ Because of the goal of improved user productivity, the information product should be integrated with the software product.
- ◆ Integration with the software required skills that the initial technical writing team did not possess.
- ◆ Integration with the software required that the information developers understand the actions taken by developers in using the *App Builder* product.

The bottom line was simple: the previous process used to produce SoftCo's examples books could not be successfully migrated to the online medium where there was an

increased focus on improving user competency and productivity. Instead, the information development team needed to re-think its approach.

Task analysis

Before asking for additional resources with the skills required for software integration, the team decided to more clearly define the measurement of productivity improvement by gaining a better understanding of the tasks an *App Builder* user performed. In the development of previously published examples books, writers tried to answer one major question before beginning the project: What examples would best demonstrate what one can do with this product? With *App Builder*—and the unique problem set it presented—there were two additional questions:

- ◆ How do users of *App Builder* use the product to do their job?
- ◆ How do they use examples to improve their understanding of how to use the product?

To answer these questions, the project team performed an informal analysis of seven users who worked with the product. Specifically, writers sat with applications developers and watched as they used *App Builder* to create software programs. The developers spoke aloud, describing their actions as they worked with the interface. The writers asked various spontaneous questions to ensure that they understood what the developers were doing during each action. The answer to the question "How do users of *App Builder* use the product to do their job?" was varied, although that finding was anticipated. The task analysis supported the assumptions about *App Builder* that users would take different routes to accomplish the same general task.

The next aspect of the analysis focused on how *App Builder* users worked with examples of programs that could be developed using the software. The project team sought to learn more about how *App Builder* users explored examples to learn how to perform tasks. The team watched and interviewed nine developers in an intermediate-level *App Builder* training course as they used examples of *App Builder* applications to create their own programs. (The developers all had a basic level of competency

The task analysis, however, focused the design of the prototype on one major issue: how to present multiple paths of discovery for a single example.

with the product and, presumably, the object-oriented concepts necessary to be productive.) Two project team members recorded notes from the interviews, which were also conducted in an informal, conversational manner.

The results of these observations led to a new way of looking at examples. Three of the participants examined an example and “learned to do” tasks by following the steps necessary to create the example. Five developers felt they needed something different. While the steps to create an example helped them, they all expressed a desire to learn by asking “How do I?”-type questions, with the answer provided within the context of the steps of an example (as opposed to a more general, documented procedure). Yet another developer wanted to see how the different pieces of the example fit together, particularly from an object-based perspective.

Prototyping and examples

Equipped with the results of this task analysis as well as the acknowledgment that the development team required additional skill sets, the first phase of the design process was initiated by gathering a group of people from various disciplines (for example, technical writers, graphic designers, software developers, user interface specialists). The group conducted a “summit” on interface design and information presentation issues. The entire team reviewed the task analysis along with market research information about the types of developers who used *App Builder* and feedback sent to SoftCo about *App Builder* (via e-mail and postcards). The group’s objective was to design a prototype that demonstrated how *App Builder* examples could be delivered using online delivery technologies. The task analysis, however, focused the design of the prototype on one major issue: how to present multiple paths of discovery for a single example.

This concept of “multiple paths of discovery” arose from the fact that examples seemed to serve different purposes for different types of users. Or, conversely, users expected examples to do different things for them. Multiple paths of discovery would enable users to look at the example from the way that they perceived the problem and to get to relevant information through a route that made sense to them. In addition to designing a usable interface, the team’s primary goal was to provide these multiple “views” or “entry points” to the same kind of information in the examples. This objective, fueled by the information gathered in the task analysis stage, reinforced the belief that the information could not be appropriately presented in book format. Instead, the team required the capabilities found in online delivery technologies.

The team focused on defining the possible paths a user might take through an example. It was proposed that users must have a specific question or problem in mind before they

The team focused on defining the possible paths a user might take through an example.

explored a path or view. For each approach a user might take, an interrogative statement was framed. The questions associated with the paths a user could take through an example included

- ◆ *How do I put this example together myself?* This view represented a step-by-step approach that listed the tasks associated with building or re-creating the example. The team agreed that previous examples books had appropriately accounted for this type of view.
- ◆ *What is this example comprised of, and how is it held together?* This approach seemed to encompass several smaller questions, such as *What does this entire example look like underneath?* or *What pieces do I need to build this example?* Its purpose was to solve the problems of those users who needed a global view of how elements of the example would or could interact. The view addresses inputs and outputs used in the example and the determination of whether those inputs and outputs were required or optional. From an object-oriented perspective, such a question could also cover the objects required by a program and the “widgets” (or visual objects) used in any interface the example might demonstrate.
- ◆ *How does this example work?* Developers working in the object-oriented model required by *App Builder* needed to know what happens behind each step, or if the sequence of steps is important, or which step causes a certain result. Such a question seemed a logical extension to the step-by-step approach, but it also appeared to function as the entry point to an example as well.

These questions served as input into the actual design of the prototype. Designs quickly focused on a presentation that included a “Step-by-step” section, a “Structure” or “Object map” section, and a “Q and A” section. The prototypes led the user/reader to the same (or similar) topics via these different sections that corresponded to the identified discovery questions. Design meetings then focused on storyboarding the examples the team had seen the developers work with in the training seminar and task analysis conducted earlier.

Once the team developed a complete working prototype, SoftCo performed several usability tests. One test involved participants in an *App Builder* training class, who

used the product to develop the small applications required in the course exercises. SoftCo was able to capture valuable information as to how users expected to find information in the interactive environment. Other tests revealed the kinds of examples users expected and how the connectivity between information and product needed to occur, in addition to more paper-oriented design questions such as colors and layout. A final test solicited feedback from a large user group meeting where users completed extensive surveys.

At first glance, the solution to providing examples documentation seemed elementary; the final version was delivered in CD-ROM format and consisted primarily of HTML files ready for viewing with common Web browsers. If the examples documentation was nothing more than this, it could have just as easily been a book “dumped online.”

However, the Web browser was actually integrated into *App Builder* in what a user would perceive as a standard tabbed dialogue. It was nearly impossible to determine that the information was actually in a format that could also be delivered as a Web page. So instead of dumping a book online, the project team

- ◆ Extended the information to target specific ways developers use examples
- ◆ Provided a facility for more entry points that enables “multiple paths of discovery”
- ◆ Integrated the information within the software product itself

ANALYSIS OF THE CASE STUDY

The final information product (referred to here as *App Builder Examples* or *ABE*) described in the case study of SoftCo makes the re-engineering concept more tangible in terms of how the hypermedia product was developed. However, further analysis of the results outlines how the re-engineering answered the critical question: *Does the resulting hypermedia product provide dramatic improvement in user performance?* This section provides such an analysis by examining the features users can access in the online information and the extent to which those features improve their performance.

Examples and how they work

The *ABE* product is installed as an optional add-on to *App Builder*. Developers can access it through *App Builder's* Project Navigator or from the Help menu. The *ABE* interface is based on other tabbed dialogs used in *App Builder* to improve its integration with the product. Standard navigation features within the tabbed dialogue enable access to five types of information or “views” for each example:

- ◆ **Overview** Describes what an example does and how it might be used. Any requirements for using an example, such as required software components, are

listed here. A thumbnail representation of how the complete example appears is also displayed to provide a visual cue for users who may be seeking a particular example.

- ◆ **Q and A** Describes how an example works or how specific features and functions operate. It is designed to provide a way for users who approach examples asking “how does this example work?” Answers to the questions contain references to the other sections, such as procedures in the Step-by-Step section or object attributes in the Objects section. In this way, *ABE* implements the multiple paths of discovery concept.
- ◆ **Structure** Presents two views of the same information: how the pieces of an example fit together. For users who benefit from visual representations, a diagram provides a pictorial view of an example, including the objects used in the interface. Users can follow hyperlinks on the diagram to see detailed information. The other structure view provides a “parts list” perspective for users who may benefit from an inventory of what is required. Again, hyperlinks on the parts list items lead to greater detail.
- ◆ **Objects** Presents two views similar to the Structure section. Instead of showing the parts necessary to build an example, this section is limited to the user interface components. As with the Structure view, both a diagram and a parts list show the objects used in the interface, and the hyperlinks provide access to details. Users benefit from having this section separated from the Structure view since the interface can be rapidly changed to achieve a different look to an application while preserving the underlying structure.
- ◆ **Step-by-Step** Details the steps required to create the application described in an example. Most steps have links to other detailed information, and some of these links access the same details that users can access from the Q and A, Structure, and Object sections.

In addition to using these views of an example, users may also choose to “download” examples directly into the *App Builder* product. The download feature places the complete example (including the code) in the *App Builder*

All 12 participants commented on how much easier *App Builder* became once they had access to *ABE*.

environment, where developers can modify the example to meet specific needs or copy some of the features directly into their applications. If users choose to make no modifications, the example still operates as a fully functional software application.

At a basic level, *ABE* was implemented as little more than files coded in HTML, delivered on a CD-ROM, and viewable through a standard Web browser. In this way, *ABE* has many similarities to standard online documents. The difference is that the supported degree of interactivity is greatly increased. Because the presentation of an *ABE* example mirrors the way different types of users work with examples, users are likely to find more useful information or to discover how to perform specific tasks. In addition, the integrated interface enables users to move quickly from interacting with the example within the *App Builder* product. Not only are the examples authentic and illustrative, but they are also useful, so *App Builder* users can then integrate the example (or parts of the example) into their work.

Users can select an example to explore from a list that contains both a description of what the working example does and a thumbnail graphic representation of the main screen or window displayed in the example. Once an example is selected, the Overview is displayed, along with the standard intra-example navigation items available from all views of an example. If users want to see detailed information about an object, they select the object from a graphic map of the working application. The *App Builder* example documentation then displays a list of the attributes required to create that object in the example. Users have the option of using that example object directly in *App Builder* code, which accepts the example information and then automatically updates the actual *App Builder* environment to reflect the change in the primary application.

Performance evaluation and results

To determine whether *ABE* provided the kind of dramatic improvements re-engineering had offered, SoftCo evaluated *ABE* with four particular questions in mind:

- ◆ Does it improve user understanding of the *App Builder* product?
- ◆ Does it enhance a user's ability to develop an application?
- ◆ Is it technically reliable?
- ◆ Does it present a feasible opportunity for further development and extension of the integrated examples concept?

The evaluation process involved two parts. The first evaluation involved an explicit test conducted informally as part of the in-house training program for *App Builder*. Participants in the evaluation were 12 SoftCo programmers and software developers who were novice *App Builder*

users taking a course to build their skills. Standard course pre-tests and post-tests were administered as part of the evaluation. After taking the pre-test and attending the first 2 days of a 3-day course, participants were given *ABE* to use to develop their own applications. A post-test was given after the third day of the course. All 12 participants scored higher than the average score usually attained by course attendees who did not use *ABE*, and 10 of the 12 attained scores in the top tenth percentile. All 12 participants commented on how much easier *App Builder* became once they had access to *ABE*.

In short, this informal evaluation shows that the re-engineering of examples-based information led to a product that was perceived as more useful.

The second part of the evaluation involved an assessment of feedback provided by *App Builder* users, including interviews by SoftCo representatives and a review of comment cards returned by users. Some findings and observations of note include:

- ◆ Users expressed the overall sentiment that, while formal training and standard documentation are useful, *ABE* provided examples in context of the task they were performing.
- ◆ Users appreciated the ability to dissect an example to get instant access to the part of the example needed to complete a task.
- ◆ The seamless integration with the product proved valuable. Users commented that, at times, they did not know what was part of *ABE* and what were standard *App Builder* features.
- ◆ The breadth of examples did not cover enough typical programming situations.

In short, this informal evaluation shows that the re-engineering of examples-based information led to a product that was perceived as more useful.

Since *ABE* is based on hypermedia, SoftCo could also include the actual examples, a fact that saves users the time of building the applications. While examples could previously accompany books (usually shipped on a diskette or CD-ROM attached to the book's cover), users would have to take extra time to locate desired examples, copy them from the shipping media to the computing environment, and then manipulate the information. With *ABE*, the actual

example code is always available. Because of delivery in a hypermedia format, users receive this added value.

Assessing the re-engineered product

The focal point of the re-engineered effort is not the use of the hypertext medium, but rather the approach to its implementation. SoftCo focused on four aspects of information development:

- ◆ Determining how the user would use both the product itself and the supporting information
- ◆ Designing the information support product based on this usage
- ◆ Emphasizing new aspects of usability testing for a more user-centered development approach
- ◆ Forming extensive relationships between traditional hypermedia designers (that is, technical communicators) and other contributors

This approach contrasts dramatically with the original steps used to produce examples books.

In many ways, the most radical aspect of this change in process is the fourth item: relationships between technical communicators and other contributors. To re-engineer its approach to the information product, SoftCo enlisted the talents of more than writers and encouraged the interaction between *ABE* and *App Builder* itself. The development of such interaction between “documentation” and “product” required integrated programming skills to get the example to “talk” with *App Builder*. In the end, the *App Builder Examples* documentation team included writers, editors, graphic designers, programmers, video specialists, instructional designers—as well as project managers. SoftCo realized that, if the supporting information was to be successfully used in connection with the user performing the task, that support information must be integrated into the product itself. Instead of having documentation that makes a user access and read information apart from the activity of working with the product, the support information could help a user do his or her job.

There are several measurements of success for SoftCo’s *App Builder Examples* product.

1. *ABE* provides more support than the passive help that typically accompanies software applications: it provides multiple paths of discovery that enable users to interact directly with the product while exploring examples.

2. Evaluations of user performance show marked improvement and proficiency. (Additionally, an informal study shows that users, for the most part, recognize the benefits of *ABE* and are interested in using it.)

3. SoftCo shows that, by making extensive use of task analyses and usability tests, incorporating a wide variety of contributors, and using a different development cycle, new hypermedia development processes can succeed.

4. The project demonstrates new logistical considerations. The development, prototyping, and testing phases took 7 months to complete, a time that was substantially longer than the corresponding research and testing phases of SoftCo’s traditional book projects. However, the project team needed only 3 more months to complete the first version, which was substantially less time than would have been required to write a linear book.

IMPLICATIONS AND CONCLUSIONS

The central argument of this article is that our accepted practices for developing online documentation, which are rooted in paper-based processes, may not be sufficient given the demands of changing technology and a changing workplace. The limitations of existing processes may provide a basis for an understanding of the problems confronting users of software applications and their support systems. From those newly framed problems, we can take a re-engineering approach to developing appropriate user-centered information. We conclude by outlining some of the implications of re-engineering hypermedia information and the necessity of further research that focuses on multidisciplinary topics.

Toward a reflective practice of re-engineering

Online documentation’s relationship to paper-based information has received extensive treatment in the research literature. Researchers have highlighted the physical similarities between paper-based books and online information, such as online searches being analogous to index items and hyperlinks functioning as “see also” items or footnotes (Brockmann 1990; Horton 1990; Mohageg 1992). Yet, as other research indicates and the SoftCo case study illustrates, non-traditional approaches—that is, approaches not rooted in paper-based methodologies—may produce creative hypermedia solutions (see Landauer 1995, an overview of 17 user-centered design projects that have led to “substantial improvement in user work efficiency,” p. 222).

To date, the focus in hypertext document research has been to take what is familiar about developing paper-based manuals and to move those characteristics online.

Print-based metaphors may be too restrictive, and metaphor is not just a matter of language but how human thought processes are conceptualized, structured, and related (Lakoff and Johnson 1980). Metaphor frames any attempt to understand one element in terms of another and, in this way, using metaphor always produces a kind of one-sided insight; that is, when we focus on particular interpretations, we tend to obscure others.

Thus, limiting the online medium to the static, paper-based metaphor may limit our ability to understand the actual problems that contemporary users face. In other words, “information is a book” may not portray the complete image that may also include “information is a tool,” “information is a safety net,” or “information is a support mechanism.” Perhaps by using different metaphors to understand the complexities (and paradoxes) of the problems users face and need to solve with online information, we might begin to design, organize, and present information in ways that we have not imagined previously. And by using different metaphors to view problems users face, we may better represent data, objects, tasks, and concepts required to assist users. Research in human-computer interaction (Norman 1993; Raybould 1995) supports this notion and shows that metaphors in user interfaces succeed when they are based not on the product (that is, a book) but on a tangible task (such as the checkbook metaphor used in personal budgeting software).

So how do technical communicators get to a point where they can appropriately frame the problems of users if these taken-for-granted ideas about information as books act as barriers? Donald Schön (1983) posits that there exists among professionals an epistemology of practice that places technical problem solving within a broader context of reflective inquiry in what he terms “reflection-in-action” (pp. 54–55). The reflection of a practitioner occurs when a problem arises that cannot be easily converted into a manageable problem. The practitioner may construct a new way of setting the problem that re-frames the situation. Schön (1983) further writes:

When a practitioner makes sense of a situation he perceives to be unique, he sees it as something already present in his repertoire. To see this site as that one is not to subsume the first under a familiar category or rule. It is, rather, to see the unfamiliar, unique situation as both similar to and different from the familiar one, without at first being able to say similar or different with respect to what. (p. 138)

The familiar situation functions as a precedent or a new metaphor for the practitioner.

In Schön’s reflective practitioner model, professionals have the capacity to see unfamiliar situations as familiar

The skills represented by the team members are more often associated with the development of software than with user support information.

ones that enable them to bring past experience to bear on the unique case. Following this concept, technical communicators can focus on user-centered issues by relying less on a specific model such as the book metaphor than on the kind of improvisation learned in practice. This user-centered focus, then, enables one to take a re-engineering approach to online documentation. By combining an expanded use of metaphors with a recognition of the reflection-in-action that supports our user support efforts, hypermedia may challenge the methods of linear, paper-based information development, just as re-engineering challenges the way businesses implement technology.

Hammer (1990) provides the essence of a shift in thinking about online documentation: “At the heart of re-engineering is the notion of discontinuous thinking—of recognizing and breaking away from the outdated rules and fundamental assumptions that underlie operations” (p. 107). To date, the focus in hypertext document research has been to take what is familiar about developing paper-based manuals and to move those characteristics online. Often the research examines the wrong problem; it is not a technological hurdle. The state of hypermedia technology today is such that hyperlinked documents are commonplace and navigation through browsers is almost a ubiquitous skill. So we do not necessarily require new technology; we just need new ways of thinking about it and of implementing it.

The SoftCo case study demonstrates how a focus on user performance can inspire new ways to use online information to fundamentally change how individuals use that information to complete a task or set of tasks. The case also shows how technical communicators can move from the single-view book metaphor, through reflective practices, and into an exploration of solutions that fulfill the definition of re-engineering. Faced with a problem set that seemed to prohibit the use of a paper manual, the ABE team found new ways of framing the situation to develop an information presentation scheme that worked in hypermedia.

One can argue that the team applied the concept of “reflection-in-action” to handle the challenge of building multiple paths of discovery, a new idea that not only further distanced the team’s direction from book methodologies but also benefited users substantially. To imple-

ment the design, the team used a development cycle completely independent from traditional book production and incorporated multiple usability assessments. Finally, *ABE*'s ability to achieve these user benefits affirms the success of the re-engineering effort.

What a re-engineering approach means for hypermedia

The SoftCo case study highlights several noteworthy issues. Most importantly, the *ABE* project shows that the significant performance improvements that should accompany re-engineering efforts can, in fact, occur. It is unlikely that the same results could have been duplicated had SoftCo used a traditional paper-based approach or “dumped” a book online. This result does not dismiss the usefulness of simply moving traditional information online; there are instances where online replication of hardcopy information can provide benefits such as speedier information access (as with reference texts) or mobility (such as the Portable Document Format that enables distributed printing capabilities).

Another interesting feature of the SoftCo case is the composition of the *ABE* team, which included technical communicators, instructional designers, graphic designers, programmers, multimedia/video specialists, and user interface specialists. The skills represented by the team members are more often associated with the development of software than with user support information. So long as integration with the product leads to improved user performance, a team consisting of individuals with varied skills may be what is required. However, to consider hypermedia as “just software” is to misunderstand its intent and the important link it has with improving user performance within the supported product. Taking full advantage of the medium, at least in *ABE*, means re-engineering information in a way that results in something that may look and feel more like software than a book.

Therefore, though re-engineering does not provide an explicit methodology that hypermedia developers might follow, SoftCo's *ABE* project may offer some ideas that can serve as attributes of a user-centered design effort. Such attributes might also provide metrics for a re-engineering approach to hypermedia:

1. *Maintain a context with the software application or product.* User support may be intrinsic (inherent to the system and integrated with its use) or extrinsic (integrated with the system, but requires users to invoke it—for example, wizards or *ABE*), but it might not work as well if it is external (such as online help).

2. *Reflect real-world work situations.* *ABE* provides real-world examples that users might actually use, revise, and explore in completing their tasks.

3. *Provide alternative search and navigation mechanisms.* This attribute is derived from the multiple paths of discovery found in *ABE*. While multiple discovery paths may be germane to examples (or, more specifically, object-oriented programming examples), it seems important that users have choices based on task type, learning style, particular information goal, and so on.

4. *Provide contextual feedback.* *ABE* users always know where they are within an example and within a task, and this information is available in alternative ways—for example, through lists, maps, and graphical representations.

5. *Structure tasks in a logical progression.* This attribute seems necessary to help users keep track of where they are. In *ABE*, a Step-by-Step section provides this feature.

6. *Collect and integrate information about users and their tasks.* The *ABE* project involved ongoing, iterative efforts to understand, explicate, and anticipate users as they attempted to accomplish real-world activities with the primary application.

These six attributes are an initial attempt to make explicit the goals of a re-engineering approach to online information. It is important to note that, while some of the characteristics such as reflecting natural work situations and providing logical procedures may work in a book, others such as an intrinsic context with the software applications and contextual feedback are simply not possible in a static document. What makes these attributes work, then, is how they work together in the online medium.

These six attributes are an initial attempt to make explicit the goals of a re-engineering approach to online information.

If the attributes we have listed here are incorporated into hypermedia designs, we are certain additional characteristics will evolve and take shape. With critical review, a general model for a re-engineered approach to hypermedia may emerge.

New research opportunities

Online information design presents a significant opportunity as well as a significant challenge for technical communicators. There is an opportunity for research in hypermedia that broadens its scope and extends beyond issues in paper-based documentation. While recent studies explore wizard development and other types of in-

egrated help, technical communicators now require research into the conscious and systematic design of support systems that provide on-demand access to all the resources individuals need to solve a problem, to perform a task, or, someday, to do an entire job. Examples-based online documentation may provide another area for study; it is worth investigating how an integration of examples with software might benefit products such as office automation tools or even custom data entry and reporting systems.

Case studies such as SoftCo's *App Builder* example documentation demonstrate radically different—and achievable—approaches to supporting and generating employee performance and knowledge via alternative design approaches. The SoftCo case also shows that an interdisciplinary approach to hypermedia can produce successful results. Research in instructional design, human-computer interaction, and information engineering offer yet other opportunities for discovery. Even business management, with its contribution of process re-engineering, may hold keys to future revelations about what we need to do to improve hypermedia products and design processes.

The challenges are equally consequential. Researchers must now confront the confluence of new technology, changing demands of the workplace, and new knowledge about how human beings learn and how performance might be improved. If hypermedia is a means by which viable solutions to user problems are provided, we need to improve our development efforts. Re-engineering, with its focus on performance improvement, provides a possible approach toward these improvements.

As companies continue to invest strategically in information technology, they will increasingly depend on improved user performance to realize productivity gains. One ultimate goal of technical communicators may be to help ensure that systems are designed, developed, implemented, and supported with user performance in mind. To reach this goal, we must re-evaluate our approach to online support documentation. More research into the online medium is certainly required. After all, we have hundreds of years of experience working with print-based tools but less than three decades in which we have explored hypermedia's potential. **TC**

REFERENCES

- Barnett, M. 1998. "Testing a digital library of technical manuals." *IEEE transactions on professional communication* 41:116–122.
- Bibus, C. J. 1990. "Changing to user-centered documentation." *Proceedings of the International Technical Communication Conference*. Arlington, VA: Society for Technical Communication, pp. WE-45-WE-48.
- Brockmann, R. J. 1990. *Writing better computer user documentation: From paper to hypertext*. New York, NY: John Wiley & Sons.
- Carroll, J. M. 1990. *The Numberg funnel: Designing minimalist instruction for practical computer skill*. Cambridge, MA: MIT Press.
- Champy, J. 1995. *Reengineering management: The mandate for new leadership*. New York, NY: HarperCollins.
- Constant, D. 1993. "The productivity paradox: Why hasn't information technology fulfilled its promise?" *SIGCHI bulletin* 25, no. 4:42–44.
- Duffy, T. M., J. E. Palmer, and B. Mehlenbacher. 1993. *Online help: Design and evaluation*. Norwood, NJ: Ablex.
- Gery, G. 1991. *Electronic performance support systems: How and why to remake the workplace through the strategic application of technology*. Boston, MA: Wiengarten Publications, Inc.
- Grice, R. 1989. "Online information: What do people want? What do people need?" In *The Society of text: Hypertext, hypermedia, and the social construction of information*, edited by E. Barrett. Cambridge, MA: MIT Press, pp. 22–44.
- Grice, R., and L. Ridgway. 1993. "Usability and hypermedia: Toward a set of usability criteria and measures." *Technical communication* 40:429–437.
- Hammer, M., and J. Champy. 1989. *Reengineering the corporation: A manifest for business revolution*. New York, NY: HarperCollins.
- Hammer, M. 1990. "Reengineering work: Don't automate, obliterate." *Harvard business review* 90, no. 4:104–112.
- Hammer, M. 1996. *Beyond reengineering*. New York, NY: HarperCollins.
- Henke, H. 1998. "Are electrons better than papyrus (Or can Adobe Acrobat Reader files replace hardcopy?)." *ACM SIGDOC 1998 Conference Proceedings*. New York, NY: ACM, pp. 29–37.
- Horton, W. 1990. *Designing and writing online documentation*. New York, NY: John Wiley & Sons.
- Horton, W. 1991. "Is hypertext the best way to document your product? A essay for designers." *Technical communication* 38: 20–30.

- Hughes, M. 1997. "Online documentation in reference-based instruction: A practical model for integrating help systems into product training." *Technical communication* 44:58–64.
- Kaindl, H., and M. Snaprud. 1991. "Hypertext and structured object representation: A unifying view." *Proceedings of Hypertext '91*. New York, NY: Institute of Electrical and Electronics Engineers, pp. 945–949.
- Lakoff, G., and M. Johnson. 1980. *Metaphors we live by*. Chicago, IL: University of Chicago Press.
- Landauer, T. K. 1995. *The trouble with computers: Usefulness, usability and productivity*. Cambridge, MA: MIT Press.
- Mehlenbacher, B. 1993. "Software usability: Choosing appropriate methods for evaluating online systems and documentation." *ACM SIGDOC 1993 Conference proceedings*. New York, NY: ACM, pp. 209–222.
- Mings, S., C. Geisler, and E. Rogers. 1993. "Hypertext technology in information design." *Proceedings of the International Professional Communication Conference*. New York, NY: Institute of Electrical and Electronics Engineers, pp. 94–97.
- Mitterer, J., D. Lunga, T. Carey, and B. Nonnecke. 1992. "A reader-centered approach to online information." *Proceedings of the International Professional Communication Conference*. New York, NY: Institute of Electrical and Electronics Engineers, pp. 312–316.
- Mohageg, M. 1992. "The influence of hypertext linking structures on the efficiency of information retrieval." *Human factors* 34, no. 3:351–367.
- Norman, D. A. 1993. *Things that make us smart—Defending human attributes in the age of the machine*. Reading, MA: Addison-Wesley.
- Pratt, J. A. 1998. "Where is the instruction in online help systems?" *Technical communication* 45:33–37.
- Raybould, B. 1995. "Performance support engineering: An emerging development methodology for enabling organizational learning." *Performance improvement quarterly* 8, no. 1:7–22.
- Redish, J. 1988. "Reading to learn to do." *The technical writing teacher* 15:223–233.
- Reece, G., and H. J. Scheiber. 1993. "Designing for dual delivery: Online and paper." *Proceedings of the International Professional Communication Conference*. New York, NY: Institute of Electrical and Electronics Engineers, pp. 85–87.
- Schön, D. 1983. *The reflective practitioner: How professionals think in action*. New York, NY: Basic Books/HarperCollins.
- Selber, S. A., J. Johnson-Eilola, and B. Mehlenbacher. 1997. "Online support systems: Tutorials, documentation, and help." In *The computer science and engineering handbook*, edited by A. B. Tucker, Jr. Boca Raton, FL: CRC Press, pp. 1619–1643.
- Smart, K. L., K. B. DeTienne, and M. Whiting. 1998. "Customers' use of documentation: The enduring legacy of print." *ACM SIGDOC 1998 Conference proceedings*. New York, NY: ACM, pp. 23–28.
- Spyridakis, J. H., and C. S. Isakson. 1991. "Hypertext: A new tool and its effect on audience comprehension." *Proceedings of the International Professional Communication Conference*. New York, NY: Institute of Electrical and Electronics Engineers, pp. 81–85.
- Turnage, J. J. 1990. "The challenge of new workplace technology for psychology." *American psychologist* 45:171–178.
- van Dam, A. 1987. "Keynote address." *Proceedings of Hypertext '87*. Chapel Hill, NC: University of North Carolina.
- van der Meij, H., and J. M. Carroll. 1995. "Principles and heuristics for designing minimalist instruction." *Technical communication* 42:243–262.
- Zimmerman, D. E., M. Tipton, L. Bilsing, and D. Green. 1993. "Hype, hypertext, and reality: What research tells us about hypertext." *Proceedings of the International Professional Communication Conference*. New York, NY: Institute of Electrical and Electronics Engineers, pp. 71–85.