Charney, D.H., Reder, L.M. & Kusbit, G.W. (1991). Improving documentation with hands-on problem solving. *Proceedings of "Documentation: The First Conference on Quality"* sponsored by the Centre for Professional Writing. University of Waterloo, Canada.

Improving Documentation With Hands-On Problem-solving

Davida H. Charney
The Pennsylvania State University

Lynne M. Reder
Gail W. Kusbit
Carnegie-Mellon University

Davida H. Charney

Dr. Charney is Assistant Professor of English at Pennsylvania State University and a member of the Graduate Faculty. She earned a Ph.D. in Rhetoric from Carnegie Mellon University and was a Post-Doctoral Fellow there, researching the design of instructional texts. She teaches courses in Technical Writing, Advanced Technical Writing and Editing, and Research Methods in Composition, as well as assuming advisory duties at both the undergraduate and graduate levels. She is an Associate editor of *Technical Communication,* the Society for Technical Communication's quarterly journal, and co-editor of *Constructing Rhetorical Education: From the Classroom to the Community.* Her many publications and collaborations reflect her interest in document design and writing quality. She is co-author of *Evaluating Technical Writing, Studies in Elaboration in Instructional Texts* (which was named the NCTE's Best Formal Research Article in Technical and Scientific Communication for 1989), and *The Role of Elaborations in Learning a Skill from and Instructional Text.* Currently in press is *The Impact of Hypertext on Processes of Reading and Writing.* Dr.Charney has been a speaker at numerous conferences, workshops, and colloquia in Canada and the United States.

Paper presented at "Documentation: The First Conference on Quality," sponsored by the Centre for Professional Writing, University of Waterloo, October 3-4, 1991. The work reported here was sponsored by the Office of Naval Research, Contract No. N00014-84-K-0063, and in part by Grant BNS-03711 from the National Science Foundation to L. Reder.

The Cognitive Demands of Documentation

Computer documentation—no matter how well written or designed—imposes heavy cognitive demands on its readers, demands of a specific kind. In order to improve documentation, in order to make it easier for readers to use, it is crucial to begin by understanding what these cognitive demands are and find ways to reduce them. The clearest way to illustrate the cognitive demands imposed by computer documentation is to contrast how readers use documentation, a form of instructions that we will term "skill-oriented" with the way they read and use other types of instructions, which we will call "project-oriented."

Project-oriented instructions

Project-oriented instructions are probably the most common type of instructions. These include all the typical do-it-yourself instructions, such as instructions for assembling a model airplane, putting up vertical blinds, or setting the time and date on a VCR. To get a sense of how readers use project-oriented instructions, consider an idealized case of a parent who buys a children's bicycle and reads the instructions for putting it together. It is immediately obvious that this is an idealized case because we are assuming that the parent actually does start by reading the instructions! In any case, the reader's goal is relatively unambiguous: she wants to end up with a functional bicycle. In fact, we can easily imagine that anyone else who bought the bicycle would have the same goal of ending up with the same bicycle. The scope of the task is thus very tightly constrained and well-specified. The bicycle-buyer starts out with a package containing an assortment of hardware and a set of instructions, usually including a picture of what the fully assembled bicycle will look like. She generally can assume—unless there has been some kind of packaging error—that all the parts needed to assemble the bicycle have been supplied in the package and that if any other tools or parts are needed, the instructions will specify what they are. In this situation, the reader's basic strategy for using the instructions can be fairly straightforward: she reads one of the directions and then tries to carry it out. When one step is correctly completed (a condition that may take a good deal of interpretive and practical effort), she moves on to the next step in the sequence. She repeats this process, moving back and forth between the text and the task, until there aren't any more steps, there aren't any leftover parts, and the bicycle works. Admittedly, this idealized version corresponds only roughly to reality, but the basic goals and strategies are easily recognizable.

An important aspect of project-oriented instructions is that they don't involve much long-term learning. Once the bicycle has been assembled, the reader usually has little if any interest in keeping the instructions—let alone trying to remember them—unless she anticipates having to fix the bicycle or take it apart again. The next time she has to put a bicycle together, she will obtain and follow another set of instructions. Admittedly, the practice she has acquired at putting bicycles together might help her follow the next set of instructions more easily. However, she probably does not aspire to assembling bicycles on her own without looking at any instructions. In cognitive terms, the reader is content to rely heavily on an external store of information, the instructions.

Skill-oriented instructions

Obviously, the situation of someone picking up a computer user's manual or other "skill-oriented" instructions is quite different and in every way more complex. Whereas the instructions for assembling a bicycle pertain to a single, well-defined task, a computer manual provides instructions that are generic and multi-purpose. For any given spreadsheet program, for example, one user may want to set up a marketing forecast, another to calculate his income taxes, and a third to analyze the data from an experiment. Even users who are doing the same kind of task will be operating under extremely different constraints and with different goals. But these readers may all draw on the same set of basic procedures to achieve these very different goals. Any single computer user, in fact, may use and reuse some procedure to perform a variety of tasks at different times. Because of the wide-ranging applicability of many functions on a computer, it is difficult or impossible to determine in advance why a computer user is reading any particular part of the manual.

As a consequence of this fact, the writer of documentation has a very difficult task. As compared to the writer of instructions for assembling a bicycle, the documentation writer can assume very little about the readers' goal, about the initial problem state, or even about the "objects" the reader has to work with. The procedures must be stated in *abstract* terms. The documentation for the spreadsheet program, for example, cannot describe its functions and features only in terms of setting up a budget, but as general procedures that can accomplish a variety of functions. This level of abstraction, in turn, has other consequences for the reader, who has to figure out how to apply—or particularize, or *instantiate*—the general procedure in his or her particular situation. The computer user's basic strategy for gathering information from the text is therefore quite different from the parent putting together a bicycle. Rather than simply moving back and forth between a list of numbered steps and a well-specified set of objects, the computer user has to figure out which command or commands to use, in what sequence, and how to apply them.

To some extent then, the documentation reader's goal is not simply to solve some particular problem (though that may be the primary goal); a documentation reader also *learns* the procedures in a way that the reader of project-oriented instructions never wants to learn the individual steps in a set of instructions. It's not so much that anyone is really interested in learning individual computer commands *per se*—and in fact some computer users strenuously avoid learning anything beyond what they absolutely have to in order to do their jobs. But all computer users must compile a repertoire of procedures for performing day-to-day tasks. And, in contrast to the reader of project-oriented instructions, computer users expect to work fairly independently of the manual. While every computer user will need to refer to the manual once in a while to find or remember some arcane command or learn a new one, she does not want to have to look up every individual command every time she uses it. In other words, there is a core of basic commands that the computer user must internalize in order to become proficient at using the computer.

Because of the way in which computer users read and learn from documentation, the most important function of documentation may be to teach users to recognize the situations in which a given command or procedure or option may be useful and provide enough specific guidance for how to apply it to their particular situation. On the basis of several years of research on documentation, we have concluded that the most effective way to help users is *not* simply by providing extended examples in the manual, *not* by providing on-line step-by-step tutorials, and *not* by leaving users to discover procedures on their own. The best approach we have found is to incorporate problem-solving into documentation. In this paper, we will describe the studies we conducted that led us to this conclusion.

The general strategy for our research has been to produce several versions of a computer manual that differ in systematic ways. We asked participants in the studies (generally college students) to read the manuals and then demonstrate what they had learned by performing a set of tasks on the computer. We observed how many tasks the participants completed successfully and how long it took them to do so. By comparing the performance of participants who read the different versions of the manual, we inferred which characteristics of the manuals seemed to lead to better performance. We also employed a variety of readers: experienced computer users as well as computer novices, readers who open a manual with a particular task in mind as well as readers who have no particular agenda. By comparing the performance of these groups of readers, we learned what kind of manual works best for different kinds of readers.

How Much Elaboration?

The first question our research addressed is how much and what kinds of elaborative information a computer manual should include. The value of elaboration in computer manuals has been a controversial issue for some time. The parties in the debate tend to fall into two quite distinct camps: the "expounders," who believe that instruction should be as complete and explicit as possible and the "minimalists," who believe that instruction should above all be brief and that it should leave much to the learner's own exploration.

The expounders' view is the more traditional: an instructional manual for novice learners should be as complete as possible; it should assume little if any prior knowledge and it should provide detailed exposition of all relevant points. This view addresses the problem of "skill-oriented" instruction by attempting to anticipate the various situations a reader might encounter and address them explicitly. Robert Tausworthe, for example, outlines several levels of detail for documenting computer software. The highest level of detail is called for in what he labels "Class A documentation," which he describes as follows: "Class A documentation is the most detailed; it contains specific definitions and detailed descriptions of every significant factor or item within the software specification. . . . This level of detail probably finds its most applicability in user manuals, and rightly so: The writer of a user manual is generally unavailable for consultation, so the user needs the extra detail" (158-159).

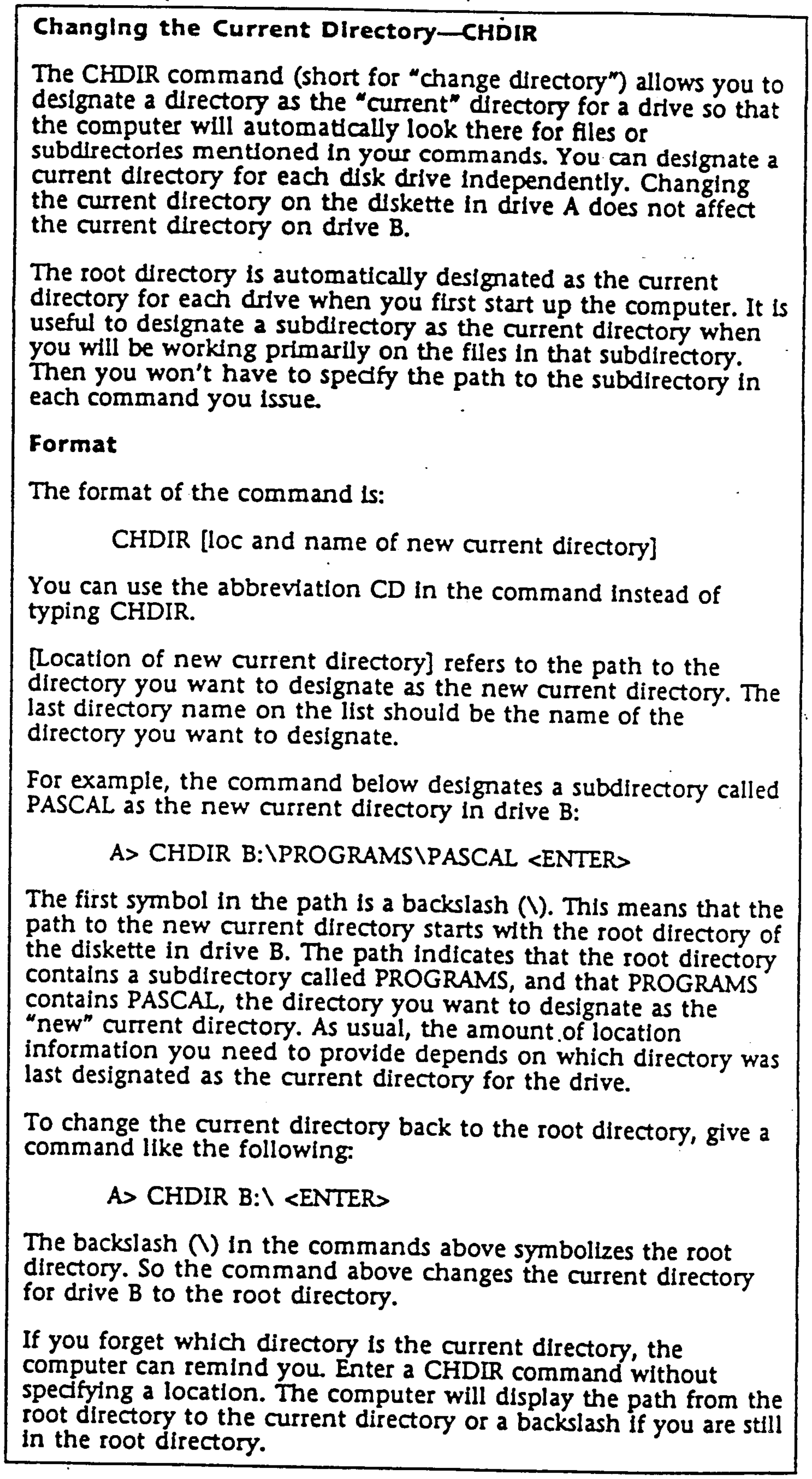
As commonly observed, however, there are drawbacks to complete and explicit documentation. In particular, users resist reading commercial manuals written according to the traditional guidelines, even when the relevant passages are easy to locate. People seem to prefer to figure things out on their own, or to ask someone for help (see, for example, discussions by Wright, Scharer, and Carroll). From this perspective, providing complete and detailed instruction seems to be of little practical use to the learner. Designers of so-called "minimalist" training materials proceed on the assumption that willingness to read a manual is inversely related to the manual's length, that people in general want to start doing things instead of reading about them and that therefore, instructional materials should actively encourage exploration by providing as little prose as possible (Carroll, et al., "The Minimal Manual").

Proponents of both approaches obviously have the readers' needs in mind, but disagree on how best to serve them. In our first studies, we investigated which approach to elaboration actually worked best for novice and experienced computer users.

Study 1

In our first study (reported in detail in Reder, Charney & Morgan), we prepared two versions of a computer user's manual that described the basic commands for the Disk Operating System (DOS) on the IBM Personal Computer (IBM-PC). One version of the manual, the Elaborated version, contained numerous definitions, analogies, examples, overviews, and other elaborations (see excerpt in Figure 1). The other version of the manual, the Unelaborated version, omitted all these elaborations and was about one-third as long—3500 words as opposed to 11,000 (Figure 2).

The participants in our study were 40 college students who were inexperienced at using computers. We gave each participant one of the two manuals to read for forty-five minutes. Then we took away the manual and asked the participants to perform a set of basic tasks on the computer. The participants were aware that the manual would not be available to them after the reading period was over.



Changing the Current Directory—CHDIR

The CHDIR command (short for "change directory") allows you to designate a directory as the "current" directory for a drive so that the computer will automatically look there for files or subdirectories mentioned in your commands. You can designate a current directory for each disk drive independently. Changing the current directory on the diskette in drive A does not affect the current directory on drive B.

The root directory is automatically designated as the current directory for each drive when you first start up the computer. It is useful to designate a subdirectory as the current directory when you will be working primarily on the files in that subdirectory. Then you won't have to specify the path to the subdirectory in each command you issue.

Format

The format of the command is:

```
CHDIR [loc and name of new current directory]
```

You can use the abbreviation CD in the command instead of typing CHDIR.

[Location of new current directory] refers to the path to the directory you want to designate as the new current directory. The last directory name on the list should be the name of the directory you want to designate.

For example, the command below designates a subdirectory called PASCAL as the new current directory in drive B:

```
A> CHDIR B:\PROGRAMS\PASCAL <ENTER>
```

The first symbol in the path is a backslash (\). This means that the path to the new current directory starts with the root directory of the diskette in drive B. The path indicates that the root directory contains a subdirectory called PROGRAMS, and that PROGRAMS contains PASCAL, the directory you want to designate as the "new" current directory. As usual, the amount of location information you need to provide depends on which directory was last designated as the current directory for the drive.

To change the current directory back to the root directory, give a command like the following:

```
A> CHDIR B:\ <ENTER>
```

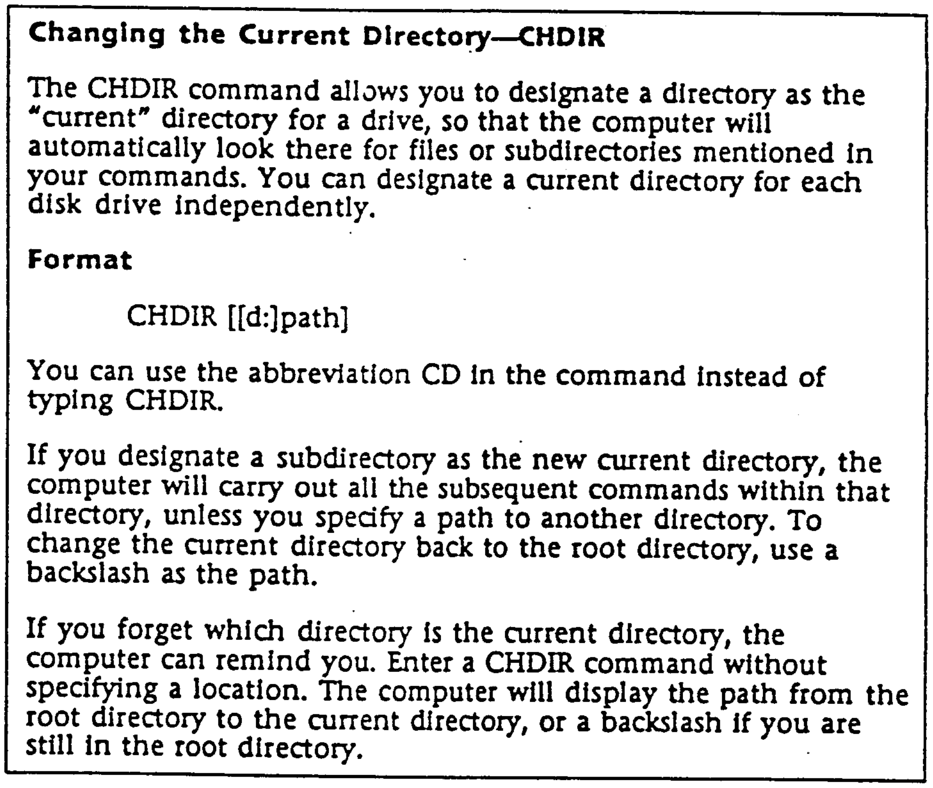
The backslash (\) in the commands above symbolizes the root directory. So the command above changes the current directory for drive B to the root directory.

If you forget which directory is the current directory, the computer can remind you. Enter a CHDIR command without specifying a location. The computer will display the path from the root directory to the current directory or a backslash if you are still in the root directory.

Figure 1. Excerpt of Elaborated PC-DOS Manual Used in Study 1 (Equivalent to Manual with Rich Procedural and Rich Conceptual Elaborations in Study 2)

In addition to varying the amount of elaboration in the manuals, we also simulated two common situations in which people read manuals. Sometimes people turn to a manual when they have a specific problem or goal in mind and are looking for information relevant to solving the problem. At other times, people turn to a manual to learn a new skill with only a general idea of how they might make use of what they learn. We simulated these two reading situations by dividing our participants into two equal groups. We gave one group advance information about the tasks they were going to perform before we gave them the manual to read. We assumed that participants in this "Task Orientation" group would then read the manual with the specific tasks in mind. The other participants (the "General Orientation" group) had no idea as they read the manual what kind of tasks we would ask them to perform. Within these two groups, half of the participants read the elaborated manual and half the unelaborated manual.

The tasks that the participants performed called directly on procedures described in the manuals: renaming files, creating subdirectories, copying and deleting files, and so on. As the participants worked at the computer, it recorded the commands and the time at which they were entered. We measured how well participants performed the tasks by counting how many tasks they were able to complete and how efficiently they worked (i.e., how much time they took and how many commands they had to issue to the computer).



Changing the Current Directory—CHDIR

The CHDIR command allows you to designate a directory as the "current" directory for a drive, so that the computer will automatically look there for files or subdirectories mentioned in your commands. You can designate a current directory for each disk drive independently.

Format

```
CHDIR [[d:]path]
```

You can use the abbreviation CD in the command instead of typing CHDIR.

If you designate a subdirectory as the new current directory, the computer will carry out all the subsequent commands within that directory, unless you specify a path to another directory. To change the current directory back to the root directory, use a backslash as the path.

If you forget which directory is the current directory, the computer can remind you. Enter a CHDIR command without specifying a location. The computer will display the path from the root directory to the current directory, or a backslash if you are still in the root directory.

Figure 2. Excerpt of Unelaborated PC-DOS Manual Used in Study 1 (Equivalent to Manual with Sparse Procedural and Sparse Conceptual Elaborations in Study 2)

The results summarized in Table 1 show that all participants completed about the same number of tasks successfully with either version of the manual. We infer from this result that both versions were adequate for learning the basic information. However, we also found very different trends for how efficiently the Task Orientation and General Orientation groups completed the tasks. The Task Orientation group performed about equally efficiently with either the elaborated or unelaborated version of the manual. On the other hand, the General Orientation group performed much better with the longer, elaborated manual—they needed significantly less time to complete the tasks and issued fewer commands than participants who used the unelaborated, more "minimalist" manual.

Measure	Task Orientation		General Orientation	
	Elaborated Manual (n=10)	Unelaborated Manual (n=10)	Elaborated Manual (n=10)	Unelaborated Manual (n=10)
Percentage of Tasks Correctly Completed	.80	.80	.85	.76
Average Time to Complete All Tasks (in minutes)	33.5	36.1	29.4	40.2
Average Number of Commands Issued To Complete All Tasks	95.8	94.2	76.8	101.8

Table 1. Average Performance at Test as a Function of.Version of Manual and Prior Orientation to the Tasks (Study 1)

Even though we found no advantage for elaborations for the Task Orientation group, the results in general support the expounder's case. The Task Orientation group was not seriously impeded by the inclusion of the elaborations. But omitting the elaborations led to significant declines in the performance of the General Orientation group. Since documentation writers cannot assume that all learners will come to the manual with such clearly defined goals as the Task Orientation group, the safest course seems to be to include the elaboration.

This conclusion conflicts with the findings of Jack Carroll and his colleagues at IBM who found advantages for a minimal manual over a commercially produced, fully elaborated manual for a word processing program ("The Minimal Manual"). Carroll et al. found that participants (mostly secretaries) who worked through the minimal version of the manual learned the basic information more quickly than those who used the full version. Furthermore, when participants went on to study more advanced topics, those who had initially worked with the minimal manual learned more new techniques more quickly. Why did our study produce such different results from these? We suspected that the conflict was mainly caused by differences in the kinds of manuals we compared and the kinds of information that was included or excluded. The elaborated manual in Carroll et al.'s study included a great deal of information (both main points and subordinate detail) that the minimal version omitted completely. In our study, the basic facts in both versions were carefully kept constant; we simply varied the amount of supporting information. It is possible that some kinds of elaborations are useful and others are not. Another important difference in the two studies was that both versions of Carroll et al.'s manual were tutorial in the sense that readers were expected to try things out on the computer as they read about them. In contrast, the participants in our study simply read the manuals and then tried to apply what they had learned on-line.[1] In our subsequent studies, we investigated the effect of both factors—the type of elaboration and the value of hands-on activity.

[1] We also note that the participants in Carroll et al's study were secretaries who were very familiar with the performance task-letter writing. They may therefore have been more like the Task Orientation group in our study who derived little or no benefit from the less elaborated manual.

What Kind of Elaboration?

To delve deeper into the issue of what kind of elaboration is of value to users, we began to consider what kinds of things people have to learn in order to acquire a skill (Charney and Reder). Our analysis led us to isolate three components to skill learning. In order to perform a new skill well, a learner must:

- Appreciate the meaning of novel concepts and the purpose of novel procedures. For example, proficient typists who have never used a word-processor must learn what things can and cannot be done on the computer. They must appreciate, for example, both the availability of automatic margin and tab adjustment and the impossibility of underlining by overstriking characters.
- Execute the procedures correctly. Learners must remember such details as where to position the cursor, in what order to type the arguments of a command, and whether a carriage return is required.
- Use the procedures at the appropriate times. Learners must remember to use the procedures they have learned and know how to choose the most appropriate procedure for a particular situation.

Elaborations in a computer manual may touch on any of these topics: what concepts and procedures are involved, when they are relevant, and how one applies them. It is possible however that users need elaboration of only some of these kinds of information but not of others. If we are right, then expounders may indeed include unnecessary information by giving detailed treatment to all points and not just the ones that need it. Conversely, minimalists may under-specify some points when learners would benefit greatly from more elaboration. In our initial study we used manuals that either elaborated on all of these points or on none of them. In our next study, we systematically varied which kinds of elaborations were included.

Study 2

In order to test the possibility that different components of skill learning require different elaboration, we classified the elaborations in our IBM-PC manual into two categories (for a complete description, see Reder, Charney, and Morgan). Elaborations were classified as "conceptual" if they concerned basic concepts, such as the purpose of the "Rename" command, or conditions for application, such as when it's a good idea to rename a file. That is, conceptual elaborations dealt with both the first and third components of skill learning. Elaborations were classified as "procedural" if they concerned the second component, learning to issue commands correctly. Procedural elaborations included examples of correct commands, details about notation conventions, and so on. We then tested all possible combinations of the conceptual and procedural elaborations by producing four versions of the IBM-PC manual with the following combinations of elaborations:

- Rich Conceptual & Rich Procedural Elaborations
- Rich Conceptual & Sparse Procedural Elaborations
- Sparse Conceptual & Rich Procedural Elaborations
- Sparse Conceptual & Sparse Procedural Elaborations

The version that was rich in both types of elaboration was equivalent to the elaborated manual in our previous study (Figure 1) and the version that was sparse in both types was equivalent to the completely unelaborated manual (Figure 2). The other two combinations are presented in Figures 3 and 4.

The procedure for this experiment was similar to that of Study 1. Participants were given time to read a version of the manual and then were asked to perform a series of tasks on the computer. In this study no participants were given advance information about the tasks they would perform—all were in a "General Orientation" condition. However, we did vary previous computer experience. Our participants included 40 novice computer users and 40 experienced computer users, none of whom had ever used an IBM-PC.[2] We expected that the novices might need elaborations of both kinds, while the experienced computer users (who were already familiar with basic computer concepts) might only need the procedural elaborations.

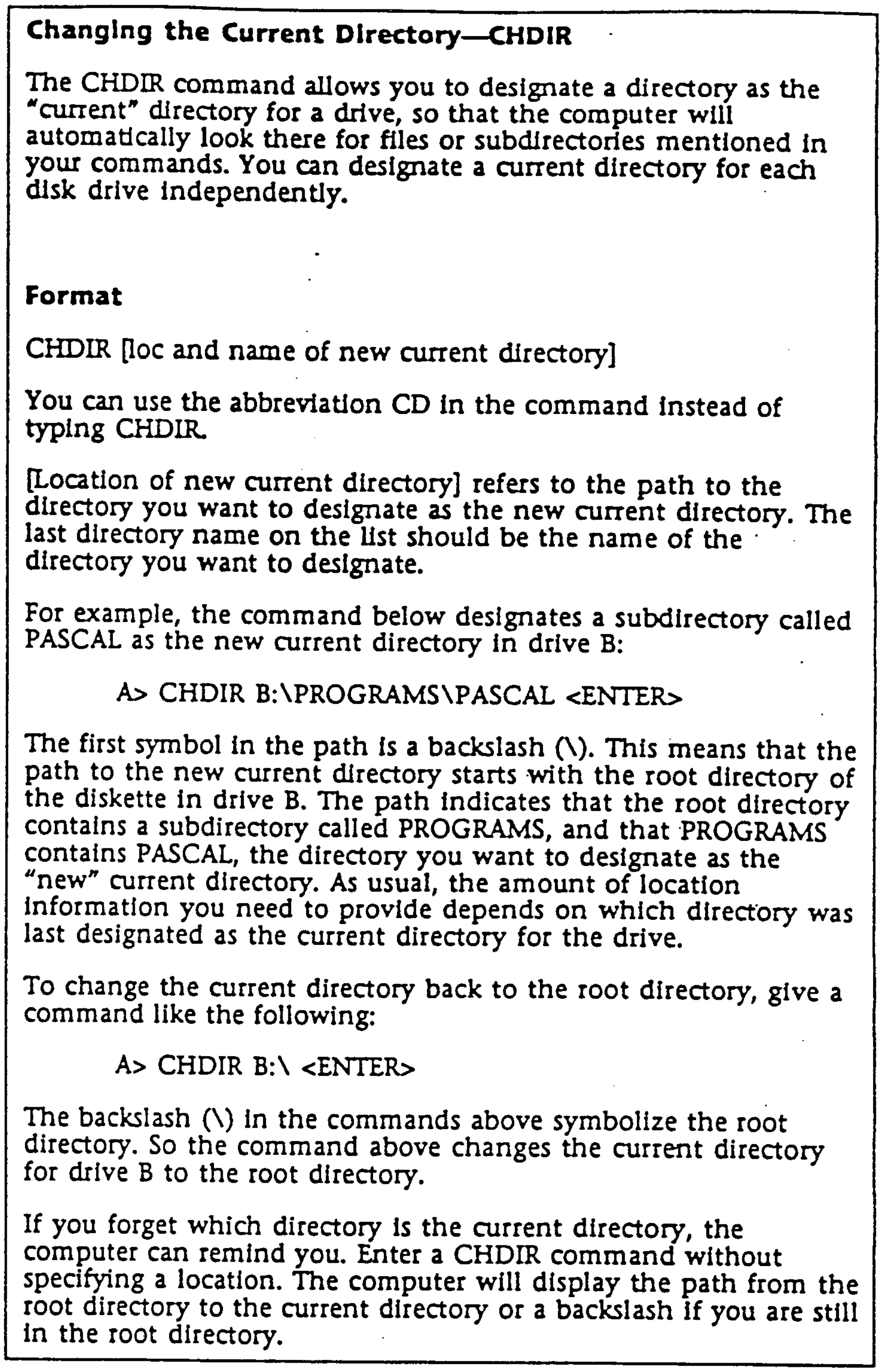
The results of this study (Table 2) indicated that the most effective manuals were those containing procedural elaborations, as shown in the leftmost two columns in Table 2. Participants who had read either of the manuals with rich procedural elaborations finished the tasks in about 37 minutes and issued about 72 commands. In contrast the participants who read manuals with sparse procedural details (the right hand columns) needed about 44 minutes and 90 commands, significantly worse performance. Surprisingly, we found no benefit at all from including the conceptual elaborations. Adding rich conceptual elaborations to manuals with rich procedural elaborations produced no better performance than having the procedural elaborations alone. Similarly, manuals with only rich conceptual elaborations produced performance no better than those having no elaborations at all.

The most surprising aspect of our results was the similarity of the patterns for novice and experienced computer users. Although the novices performed less well in general than the experienced computer users, they did not benefit any more than the experienced users from the conceptual elaborations. On the other hand, they did benefit just as much from the procedural elaborations. It is possible that the type of conceptual elaborations we included were not exactly what the novices needed. Nystrand conducted a study that is consistent with this possibility. He found that "high knowledge" and "low knowledge" computer users asked different kinds of questions when trying to use computer documentation. Low knowledge participants most often asked for "categorical definitions" while high knowledge participants most often asked for "further specifications." Nystrand characterized the differences in the questions as requests for topic elaborations (categorical definitions) or comment elaborations (further specification). He also found that adding the appropriate kinds of elaborations reduced the questions asked by both groups.

The conclusion to be drawn from our two studies is that one cannot produce an effective manual by being either a pure minimalist or a pure expounder. While conceptual elaborations may be dispensable, computer users—both novices and experienced users—clearly benefit from another kind of elaboration—those that illustrate how to carry out general procedures.[3]

[2] We evaluated computer experience with a questionnaire which probed what kind of computers and software applications the participants had used as well as their previous programming experience.

[3] Other researchers have also found differential benefits for particular types of elaboration. For example, Kieras found that people learning to operate a mechanical device derive little benefit from detailed information about the internal workings of the device. He found that such "how it works" information is only useful if the learner must *infer* the operating procedures (rather than memorizing them or looking them up) and if the "how it works" information is specific enough to enable such inferences.



Changing the Current Directory—CHDIR

The CHDIR command allows you to designate a directory as the "current" directory for a drive, so that the computer will automatically look there for files or subdirectories mentioned in your commands. You can designate a current directory for each disk drive independently.

Format

CHDIR [loc and name of new current directory]

You can use the abbreviation CD in the command instead of typing CHDIR.

[Location of new current directory] refers to the path to the directory you want to designate as the new current directory. The last directory name on the list should be the name of the directory you want to designate.

For example, the command below designates a subdirectory called PASCAL as the new current directory in drive B:

```
A> CHDIR B:\PROGRAMS\PASCAL <ENTER>
```

The first symbol in the path is a backslash (\). This means that the path to the new current directory starts with the root directory of the diskette in drive B. The path indicates that the root directory contains a subdirectory called PROGRAMS, and that PROGRAMS contains PASCAL, the directory you want to designate as the "new" current directory. As usual, the amount of location information you need to provide depends on which directory was last designated as the current directory for the drive.

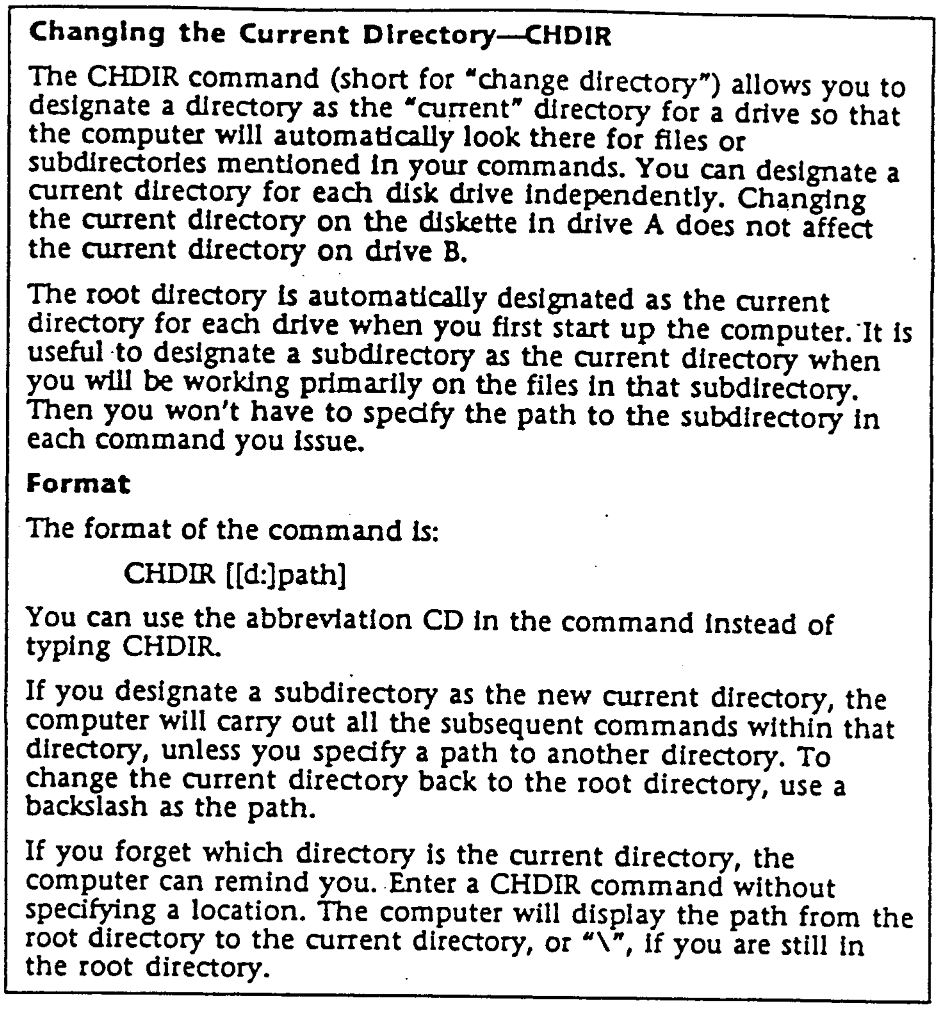
To change the current directory back to the root directory, give a command like the following:

```
A> CHDIR B:\ <ENTER>
```

The backslash (\) in the commands above symbolize the root directory. So the command above changes the current directory for drive B to the root directory.

If you forget which directory is the current directory, the computer can remind you. Enter a CHDIR command without specifying a location. The computer will display the path from the root directory to the current directory or a backslash if you are still in the root directory.

Figure 3. Excerpt of Manual Illustrating Rich Procedural and Sparse Conceptual Elaborations Used in Study 2



Changing the Current Directory—CHDIR

The CHDIR command (short for "change directory") allows you to designate a directory as the "current" directory for a drive so that the computer will automatically look there for files or subdirectories mentioned in your commands. You can designate a current directory for each disk drive independently. Changing the current directory on the diskette in drive A does not affect the current directory on drive B.

The root directory is automatically designated as the current directory for each drive when you first start up the computer. It is useful to designate a subdirectory as the current directory when you will be working primarily on the files in that subdirectory. Then you won't have to specify the path to the subdirectory in each command you issue.

Format

The format of the command is:

```
CHDIR [[d:]path]
```

You can use the abbreviation CD in the command instead of typing CHDIR.

If you designate a subdirectory as the new current directory, the computer will carry out all the subsequent commands within that directory, unless you specify a path to another directory. To change the current directory back to the root directory, use a backslash as the path.

If you forget which directory is the current directory, the computer can remind you. Enter a CHDIR command without specifying a location. The computer will display the path from the root directory to the current directory, or "\", if you are still in the root directory.

Figure 4. Excerpt of Manual Illustrating Sparse Procedural and Rich Conceptual Elaborations Used in Study 2

Why were the procedural elaborations so useful? The benefit we found is akin to the widely known strategy of learning from examples. Psychological studies of concept and skill learning alike have shown that studying worked-out examples to a problem (often in conjunction with an abstract rule or procedure) can help people identify categories of problems and appropriate solution strategies for each (Ross and Kennedy; Sweller and Cooper; Tarmizi and Sweller; Nitsch; Tennyson, Woolley, and Merrill). Studies have also shown that learners use worked-out examples as scaffolds for constructing solutions to new problems: they retrieve the example from memory and replace the terms specific to the old problem with terms relevant to the new one (Anderson, Farrell and Sauers; Ross). Examples, then, provide computer users with two ways to meet the cognitive demands imposed by skill-oriented instructions. First, as indicated by the results of Study 1, computer users need to know something about the contexts in which a general procedure may be useful. Seeing an example of a command being applied in a particular situation provides information about typical situations, or categories of problems. Second, as indicated by Study 2, computer users need to learn how to apply a general procedure to a specific situation—having the abstract rule for the computer command was not sufficient even for experienced computer users. An example, by providing an appropriate instantiation of a rule for a particular situation, provides a scaffold with which learners can construct new instantiations.

Measure	Rich Procedural Elaborations: Rich Conceptual Elaborations (n=20)	Rich Procedural Elaborations: Sparse Conceptual Elaborations (n=20)	Sparse Procedural Elaborations: Rich Conceptual Elaborations (n=20)	Sparse Procedural Elaborations: Sparse Conceptual Elaborations (n=20)
Percentage of Tasks Correctly Completed	.73	.80	.73	.73
Average Time to Complete All Tasks (in minutes)	37.4	37.7	43.5	45.9
Average Number of Commands Issued To Complete All Tasks	71.7	73.7	88.7	92.4

Table 2. Average Performance at Test for Manuals Containing Four Combinations of Elaborations (Study 2)

Reading examples, then, is one way to help computer users learn about the range of problems that various procedures can help them solve, as well as how to execute those procedures. But is reading an example in a manual the most effective learning paradigm? In our next two studies we addressed the issue of learning by doing, the claim that people not only prefer to try things on their own, but also that they learn more that way than by reading manuals. In particular, we investigated the relative effectiveness of having learners read manuals, follow tutorials, solve a set of problems, and explore the features of a system on their own. Each of these formats is consistent with the notion that computer users need examples of contexts in which procedures are used, but each makes different assumptions about the types of examples they need to see and how they process them.

Learning by Reading, Following, Exploring, and Problem-solving

Study 3

In this study, we compared the relative merits of reading a manual (the approach used in our previous two studies) with two new paradigms: following a tutorial and solving problems. Of the two, tutorials in which a user follows step-by-step instructions to enter commands that demonstrate the computer's features, are by far the more familiar. On-line tutorials (augmented in some cases with audio tapes or on-line simulations) have become a staple of much computer documentation. Tutorials would seem at first glance to be simply an enhanced, active form of reading a manual with examples. Both set up hypothetical situations in which the computer procedures may be appropriately applied. The tutorial simply adds the physical activity of carrying out the instructions on the computer and allows the user to observe the computer's prompts and feedback messages at each stage along the way. Given the similarities between reading examples and the guided activity in tutorials, one might expect that people using tutorials would learn at least the same amount, and probably more than, people who simply read manuals that contain examples.

FIGURE 5. Sample Training Problem in Problem-Solving and Tutorial Form (Used in Studies 3 and 4)

1a. Problem-Solving Form
Alphabetize the names, by putting the rows containing Steele and Stewart further down in the appropriate spots.

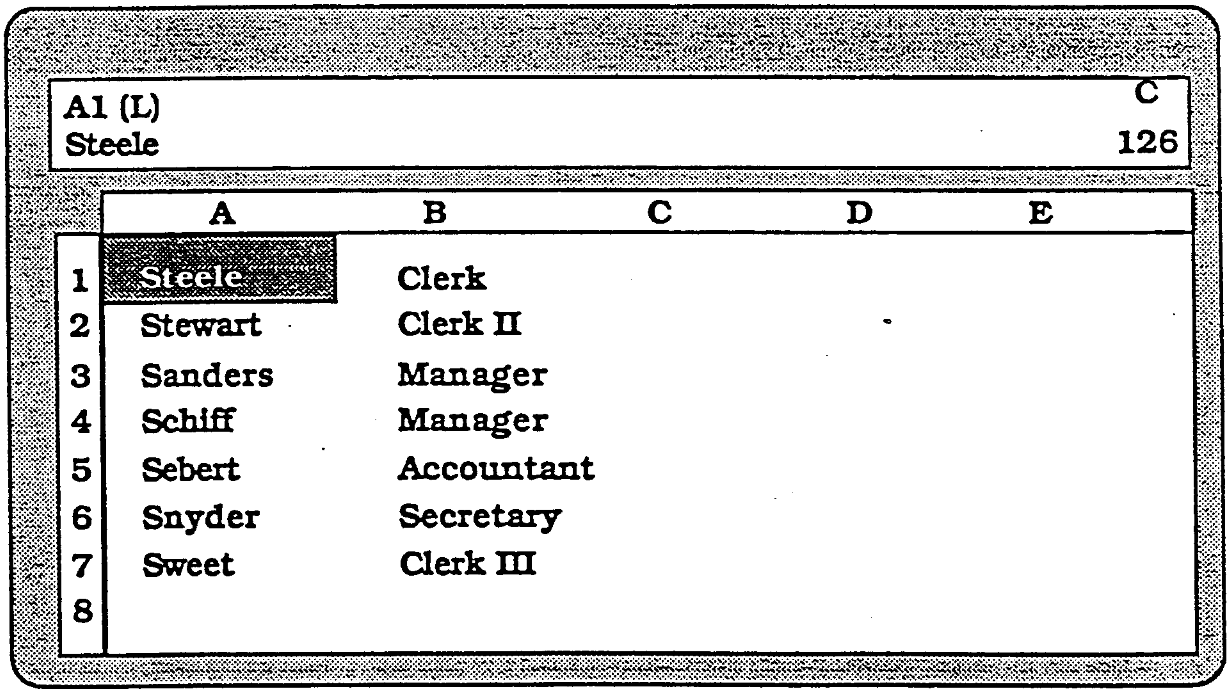
Feedback appearing on the following page:
You could have used the following sequence of commands (starting with cell A1 as the current cell) to solve the preceding problem:

```
/M . A7 [RETURN]
/M . A7 [RETURN]
```

1b. Tutorial Form
To alphabetize the names, by putting the rows containing Steele and Stewart further down in the appropriate spots, type the following sequence of commands (starting with cell A1 as the current cell):

```
/M . A7 [RETURN]
/M . A7 [RETURN]
```

VisiCalc display as it appeared on the computer for all versions of the problem



```
A1 (L)                                  C
Steele                                126

      A          B            C      D      E
1   Steele     Clerk
2   Stewart    Clerk II
3   Sanders    Manager
4   Schiff     Manager
5   Sebert     Accountant
6   Snyder     Secretary
7   Sweet      Clerk III
8
```

Solving problems as a learning strategy grows out of the tradition of chapter-end exercises in math or science textbooks. Like the reading examples and tutorial approaches, this approach starts with a text that contains problems prepared in advance for the learner. However, instead of studying or carrying out a solution provided in the text, the learner must arrive at a solution independently by retrieving candidate procedures from memory, selecting an appropriate one for the given problem, generating a correct instantiation of the procedure, and executing it correctly. In addition, learners may receive feedback on the solution, in the form of answers in the back of the book. In this analysis, solving problems should provide some of the same benefits as studying worked-out examples (or perhaps tutorials) by providing information about the types of situations in which specific procedures may be useful. Solving problems may do more, however, because it forces learners to actively consider the possible procedures and to attempt to execute the chosen procedures independently. Sweller and Cooper dispute this claim, arguing that learners who simply solve problems may spend time fruitlessly on incorrect solution paths and thereby fail to acquire good models of solutions.

In this study (described in more detail in Charney and Reder, "Designing Interactive Tutorials") participants learned how to use the VisiCalc electronic spreadsheet. The study was conducted in two sessions consisting of a training session and a testing session two days later. In the training session, 44 participants (mainly undergraduates) learned 12 VisiCalc commands by reading brief manual entries describing the purpose of each command and how to execute it. In addition to the entry for each command, the manual contained problem sets, consisting of three problems for each command. In all relevant respects, the problems were equivalent to examples of concrete situations in which a procedure could be used to achieve some particular goal. However, some problems were presented with explicit solutions—an exact sequence of keystrokes—and others were presented as unsolved problems with the correct solution presented as feedback on the following page (Figure 5). Notice that both versions of the problem presented exactly the same information to the participant, but at different times. We systematically varied what kinds of problems participants saw and what they did with them.

Read Only Group

One group of 14 participants was assigned to the "Read Only" group. These participants received a manual in which all the problem sets contained worked out solutions. In effect, these participants followed a learning paradigm similar to that of our previous studies, in which they simply read the manual and were later tested trying to carry out the procedures on-line for the first time.

Interactive Instruction Group

The remaining 30 participants were assigned to the "Interactive Instruction" group. These participants all used the computer to try out the commands on-line as they encountered the problems. However, they received different kinds of problems for different commands. Any given command may have been accompanied by three kinds of problem sets (Pure Tutorial, Pure Problem-solving, and Mixed Practice). Since there were 12 commands in the manual, each participant in the Interactive Instruction group learned four commands with each of the three kinds of problem sets.

- Pure Tutorial. The three problems for a command all included the exact sequence of keystrokes that participants were to enter to achieve the goal. These problems were identical to those presented to the "Read Only" group—the difference is that participants actually entered the commands listed in order to solve the problems.
- Pure Problem-solving. The three problems for a command presented a specific goal but included no instructions for how to achieve it. After participants solved the problem to their satisfaction (or gave up), they saw feedback on the recommended sequence of keystrokes.
- Mixed Practice. The first training problem for a command was presented in Tutorial form and the remaining two were in Problem-solving form.

Two days after the training session, all participants were given a test consisting of 12 new problems to solve on the computer, one problem for each command. We evaluated their performance in terms of how many problems they solved correctly and how much time they needed to reach a solution.

Overall, the results indicated that working interactively with the computer was more effective than simply reading the manual. The participants in the Interactive Instruction group as a whole solved more test problems more quickly and accurately than participants in the Read Only group (Table 3). These results support the notion that active learning situations in which people apply procedures for themselves are more effective than those in which they simply study the procedures. Among the interactive conditions, problem-solving was clearly the most effective form of training. Participants solved significantly more test problems correctly for commands they had learned with Pure Problem-Solving or Mixed Practice than with Tutorials. The accuracy for Tutorials was nearly as low as for the Read Only group. Furthermore, participants did not need a tutorial before trying to solve problems on their own. In particular, training with Mixed Practice did not lead to better performance than Pure Problem-Solving. However, we did not find that the different forms of training influenced participants' efficiency, or how quickly they could solve a problem. To this extent, we found at least some benefit—in terms of facility at entering commands—to following a tutorial as compared to simple reading.[4]

	Interactive Instruction (n=30)		**Read Manual Only (n=14)**	
Measure	Pure	Pure Tutorials	Mixed Problem-Solving	Practice
Percentage of tasks correctly completed	.53	.66	.68	.48
Average time to correct solution (in minutes)	1.52	1.58	1.42	2.23

Table 3. Average Performance at Test for Manuals Promoting Different Kinds of Learner Activity (Study 3)

We believe that the problem with tutorials is that learners are apt to enter keystrokes mechanically, without thinking about the purpose of each action or even observing the screen. Therefore, even though tutorials can provide examples of how to instantiate general procedures in a range of realistic situations, learners do not process these examples well enough to derive lasting benefit. We believe that problem-solving is more effective than tutorials because it combines the provision of exemplary problem situations with active processing. The problems we found with tutorials are consistent with those of Carroll who found tutorials to be less effective than a Minimal Manual.

Carroll, however, does not argue for presenting learners with exemplary situations. Instead, he argues that the most effective form of training is to allow computer users to explore the system and discover procedures on their own. In this exploration-based training, learners experiment directly on a device or computer application, in effect creating their own example training problems and working out how to solve them. Carroll and his colleagues argue that problems set by the learners themselves are intrinsically more motivating and less susceptible to misinterpretation than problems set in an instructional text ("Exploring a Word Processor"; see also Kamouri, Kamouri and Smith). They compared a commercially produced tutorial manual for a word processing program to a set of guided exploration materials. In order to force learners to discover procedures through interactions with the computer, the guided exploration materials omitted procedural rules or any explicit, formal statements of how to execute the commands. The materials did provide hints about useful keys and menus,

[4] This conclusion must be tempered by the fact that we did not set up a direct comparison of Tutorial Only against Read Only. Participants' speed in the Inactive Instruction group may have been improved in part by their problem-solving practice.

checkpoints for assessing success, and remedies for recognizing and recovering from mistakes. The learners, who were experienced secretaries, experimented with procedures at their own initiative and formulated their own goals for applying them. Once again, tutorials were ineffective. The guided exploration learners trained more quickly, completed the criterion tasks more quickly, and made fewer procedural errors than the group trained with the tutorial. It is difficult to generalize from the results of this study, however, because the length, content, and presentation format of the instructional information varied widely in the two conditions.

Study 4

In this study, we compared the relative effectiveness of problem-solving and a form of exploration learning. The design and procedure for this study were similar in many ways to those of Study 3. The 65 participants again learned 12 VisiCalc commands during a training session and demonstrated what they had learned and retained in a test session two days later. In this study, however, we eliminated Mixed Practice condition—the 45 participants in the Interactive Instruction group saw only pure tutorial or pure problem-solving training sets. Further, instead of having a Read Only group, we assigned 20 participants to an Exploration Learning group. Participants assigned to the Exploration Learning group were provided with the same instructional manual as the other participants but without the training problems. They were told that there was no constraint on the order in which they could study or practice the commands. These participants experimented with the commands at their own initiative, looked back in the manual at any time, and freely created their own spreadsheets or modified the set of practice problem spreadsheets that were stored on-line. Two days after training, the Exploration Learning group took the same test as the other participants.

The test results for the Interactive Instruction group (Table 4) replicated the results of Study 3: participants were significantly better able to solve test problems—both in terms of correct completion and speed—when they had trained with problem-solving than with tutorials. Surprisingly, the exploration-based training produced no better results than the tutorials. Participants in the exploration group solved significantly fewer test problems correctly than participants in the problem-solving condition and their solution times were intermediate—neither significantly better nor worse than the other conditions. Overall, it is clear that problem-solving was the most effective form of training.

	Interactive Instruction (n=45)		**Exploration Learning (n=20)**
Measure	Tutorials	Problem-Solving	
Percentage of tasks correctly completed	.53	.71	.54
Average time to correct solution (in minutes)	2.23	1.60	1.80

Table 4. Average Performance at Test for Manuals Promoting Different Kinds of Learner Activity (Study 4)

We should note here that participants did spend significantly more time during training in the problem-solving condition than in the other training conditions. However, our statistical analyses argue against the possibility that the advantage of problem-solving was simply due to increasing the time participants spent on training (for a full discussion, see "Goal Setting"). Our analysis of the participants' behavior (as recorded on videotape) suggests that they spent more time in the problem-solving condition only on commands that were more difficult for them. Longer training times usually indicated repeated (and therefore unsuccessful) attempts at solving a training problem. Those who had difficulty on the training problems for a command also tended to have difficulty with that command at test.

Why did exploration-based training fare so badly in this study? As others have noted, in order for discovery learning to work, learners must succeed at discovering the "desired" principles (Anthony) or at recognizing the potential usefulness of a rule (Scandura). If exploration learners in our study were unable to invent appropriate situations for applying the commands, they would not have discovered what they needed to learn. The poor performance of our Exploration group, then, may be due to the participants' minimal prior knowledge of the domain. Carroll's successful use of exploration learning to teach word processing skills occurred with temporary office workers who knew a great deal about the goals and strategies involved in producing business correspondence. In contrast, our participants were students who had little experience with the goals and strategies for using a spreadsheet. By working the problems in the manual, the Interactive Instruction group (who saw training problems in both problem-solving and tutorial forms) surely learned a great deal about how spreadsheets are typically used and in what contexts specific commands may be useful. However, while the Exploration group had access to the spreadsheets used for the problems in the manual, they were not able to invent corresponding problems independently, nor did they always choose to use the spreadsheets that we had created. In short, the ability of the Exploration group to fully appreciate the functions of the various spreadsheet commands and when to use them may have been severely constrained.

Informal observations of the activities of the exploration learning group during training are consistent with this conclusion. To their credit, these participants seemed engaged in the task as a whole and were generally successful at generating correct commands. They tended to practice each command numerous times, whereas participants in the Interactive Instruction group were limited to only three opportunities to practice each command (as provided by the three training problems per command in tutorial or problem-solving form). However, the exploration-based learners did not practice (or did not discover) all the suboptions and ramifications of the commands; instead, they repeatedly practiced relatively simple applications. As a result, many failed to notice important consequences of the commands. For example, most exploration learners succeeded at splitting the display into two windows, but never attempted to scroll the windows independently. Similarly, most participants succeeded at using the replicate command to copy data in one set of cells to another location, but never used it to reapply mathematical functions to a new subset of data.

Most importantly, we noted that participants in the exploration group tended not to invent spreadsheets that represented full-blown, meaningful scenarios for manipulating numerical data. While we provided participants with a typical spreadsheet at the onset of the experiment as well as a list of other available spreadsheets stored on-line, participants did not make great use of them. Even when they started with a blank spreadsheet, few participants attempted to create a checkbook register, or a monthly budget ledger. Instead, they tended to create ad hoc fragments, typing in a few columns of numbers chosen apparently at random. As a result, they tended not to create or recognize situations that motivated the use of particular commands. For example, if the participant experimented with the "Windows" command (which split the display into independently scrollable windows) while using an empty spreadsheet or one containing entries that easily fit on the display, she would not appreciate the value of the command for bringing distant columns or rows into view. In fact, she might not appreciate that the size of the display screen constrains effective use of a spreadsheet. In contrast, the training problems for the Windows command (used in the Problem-Solving and Tutorial training conditions) presented participants with the goal of bringing distant columns into view in an appropriate context (e.g., a column with year-end totals and a column representing data for the third month).

Exploration-based discovery learning seems to involve two distinct phases, problem formulation and problem-solving. In the *problem formulation* phase, learners decide to experiment and invent a task or problem. For learners unfamiliar with the domain, this phase may be difficult or problematic. These learners may have trouble fully exploring the domain because inexperience prevents them from setting appropriate problem goals. That is, unless they know something about the range of problem situations that may arise in the domain, they may not be able to invent typical or important tasks or problems. Further, novices may not invent situations that allow them to assess the appropriateness of one procedure over another. Both potential problems stem from a lack of knowledge of what is possible or desirable to do in the new domain and a lack of knowledge of typical problem situations that may arise.

In the second phase, *problem-solving*, learners work on the task they set for themselves, using domain-specific techniques of problem-solving. We might expect the problem-solving phase of exploration learning to share some of the same benefits as the problem-solving strategy discussed previously, particularly in providing practice at applying commands appropriately. However, it might not provide effective practice at selection since the learner may have a particular command in mind when generating a problem. In contrast, learners who are given a problem to solve do not know ahead of time which command is intended and therefore must review the available commands and select one that seems appropriate. Further, exploration learners may have more difficulty accurately evaluating their progress; nor is it easy to provide feedback to learners who set their own goals. Discovery learners may retain misconceptions that do not quickly produce salient errors. Further, learners may not be able to evaluate the quality of their solutions, to distinguish between makeshift solutions and more efficient or elegant ones. In contrast, problem-solving does provide learners with exemplary situations in which they must actively select and implement a solution to a problem; further, providing feedback gives learners access to solutions that they may not have thought of, or confirms the quality of a good solution. For these reasons, we believe that problem-solving is a more effective form of training than exploration.

Conclusion

We began this chapter with an analysis of how readers use documentation, an analysis which highlighted both the necessity for manuals to present general, all-purpose procedures and the difficulties that such procedures impose on users who must apply them in specific situations. Consistent with this analysis, we found that users learned most from manuals in which only certain classes of information were elaborated. Participants did not benefit from elaborations of general concepts (e.g., what is a disk drive) or from elaborations offering advice on when to apply specific procedures. However, they did benefit from elaborations on how to apply procedures, in particular, from well-chosen situational examples. We also found that hands-on practice produced even better results than simply studying a manual containing examples. The improvement from hands-on practice was only slight when examples were replaced with a highly directive hands-on tutorial. Both tutorials and reading alone were significantly worse than problem-solving exercises that forced the learner to independently apply the instruction offered. Finally, we found that problem-solving exercises were also superior to exploration-based learning, in which learners invent their own problems to solve.

References

Anderson, John R., Robert Farrell, and Ron Sauers. "Learning to Program in LISP." *Cognitive Science* 8 (1984): 87-129.

Anthony, W. S. "Learning to Discover Rules by Discovery." *Journal of Educational Psychology* 64 (1973): 325-328.

Carroll, John M. "Designing MINIMALIST Training Materials." *Datamation* 30.18 (1984): 125-136.

Carroll, John M., Robert L. Mack, Clayton H. Lewis, Nancy L. Grischkowsky, and Scott R. Robertson. "Exploring Exploring a Word Processor." *Human Computer Interaction* 1 (1985): 283-307. Reprinted in *Effective Documentation: What We Have Learned from Research.* Ed. Stephen Doheny-Farina. Cambridge: MIT Press, 1988. 73-102.

Carroll, John M., Penny L. Smith-Kerker, Jim R. Ford, and Sandra A. Mazur. "The Minimal Manual." *Human Computer Interaction* 3 (1987): 123-153. Reprinted in Effective Documentation: *What We Have Learned from Research.* Ed. Stephen Doheny-Farina. Cambridge: MIT Press, 1988. 103-126.

Charney, Davida, and Lynne Reder. "Initial Skill Learning: An Analysis of How Elaborations Facilitate the Three Components." *Modelling Cognition.* Ed. Peter Morris. Chichester: Wiley, 1987. 135-165.

Charney, Davida, and Lynne Reder. "Designing Interactive Tutorials for Computer Users." *Human Computer Interaction* 2 (1986): 297-317.

Charney, Davida, Lynne Reder, and Gail W. Kusbit. "Goal Setting and Procedure Selection in Acquiring Computer Skills: A Comparison of Tutorials, Problem-solving, and Learner Exploration." *Cognition and Instruction* 7.4 (1990): 323-342.

Charney, Davida, Lynne Reder, and Gail Wells. "Studies of Elaboration in Instructional Texts." *Effective Documentation: What We Have Learned from Research.* Ed. Stephen Doheny-Farina. Cambridge: MIT Press, 1988. 47-72.

Hermann, G. D. "Learning by Discovery: A Critical Review of Studies." *Journal of Experimental Education* 38 (1969): 58-72.

Kamouri, Anita, Joseph Kamouri, and Kirk Smith. "Training by Exploration: Facilitating the Transfer of Procedural Knowledge Through Analogical Reasoning." *International Journal of Man-Machine Studies* 24 (1986): 171-192.

Nitsch, K. E. *Structuring Decontextualized Forms of Knowledge.* Unpublished doctoral dissertation, Vanderbilt University, 1977.

Nystrand, Martin. *The Structure of Written Communication: Studies in Reciprocity Between Writers and Readers.* Orlando, FL: Academic Press, 1986.

Reder, Lynne, Davida Charney, and Kim Morgan. "The Role of Elaborations in Learning a Skill from an Instructional Text." *Memory and Cognition* 14 (1986): 64-78.

Ross, Brian. "Distinguishing Types of Superficial Similarities: Different Effects on the Access and Use of Earlier Problems." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15 (1989): 456-468.

Ross, Brian and Patrick T. Kennedy. "Generalizing from the Use of Earlier Examples in Problem-solving." *Journal of Experimental Psychology: Learning, Memory, and Cognition* 16 (1990): 42-55.

Scandura, Joseph. "An Analysis of Exposition and Discovery Modes of Problem-solving Instruction." *Journal of Experimental Education* 33 (1964): 149-159.

Scharer, Laura. "User Training: Less is More." *Datamation* 29 (1983): 175-182.

Sweller, John. "Cognitive Load during Problem-solving: Effects on Learning." *Cognitive Science* 12 (1988): 257-285.

Sweller, John, and Graham Cooper. "The Use of Worked Examples as a Substitute for Problem-solving in Learning Algebra." *Cognition and Instruction* 2.1 (1985): 59-89.

Tarmizi, Rohani, and John Sweller. "Guidance During Mathematical Problem-solving." *Journal of Educational Psychology* 80 (1988): 424-436.

Tausworthe, Robert. *Standardized Development of Computer Software, Part II.* Englewood Cliffs, NJ: Prentice Hall, 1979.

Wright, Patricia. "Manual Dexterity: A User-Oriented Approach to Creating Computer Documentation." *Proceedings of CHI '83 Human Factors in Computing Systems* (Boston, December 1983). Ed. Ann Janda. New York: ACM, 1983. 11-18.