5 Paradox of the Active User[1]

John M. Carroll and
Mary Beth Rosson

1 Introduction

One of the most sweeping changes ever in the *ecology* of human cognition may be taking place today. People are beginning to learn and use very powerful and sophisticated information processing technology as a matter of daily life. From the perspective of human history, this could be a transitional point dividing a period when machines merely helped us do things from a period when machines will seriously help us think about things. But if this is so, we are indeed still very much within the transition. For most people, computers have more possibility than they have real practical utility.

In this chapter we discuss two empirical phenomena of computer use: (1) people have considerable trouble learning to use computers (e.g., Mack, Lewis and Carroll, 1983; Mantei and Haskell, 1983), and (2) their skill tends to asymptote at relative mediocrity (Nielsen, Mack, Bergendorff, and Grischkowsky, 1986; Pope, 1985; Rosson, 1983). These phenomena could be viewed as being due merely to "bad" design in current systems. We argue that they are in part more fundamental than this, deriving from conflicting motivational and cognitive strategies. Accordingly, (1) and (2) are best viewed not as design problems to be solved, but as true paradoxes that necessitate programmatic tradeoff solutions.

A motivational paradox arises in the "production bias" people bring to the task of learning and using computing equipment. Their paramount goal is throughput. This is a desirable state of affairs in that it gives users a focus for their activity with a system, and it increases their likelihood of receiving concrete reinforcement from their work. But on the other hand, it reduces their motivation to spend any time just learning about the system, so that when situations appear that could be more effectively handled by new procedures, they are likely to stick with the procedures they already know, regardless of their efficacy.

A second, cognitive paradox devolves from the "assimilation bias": people apply what they already know to interpret new situations. This bias can be helpful, when there are useful similarities between the new and old information (as when a person learns to use a word processor taking it to be a super typewriter or an electronic desktop). But irrelevant and misleading similarities between new and old information can also blind learners to what they are actually seeing and doing, leading them to draw erroneous comparisons and conclusions, or preventing them from recognizing possibilities for new function.

It is our view that these cognitive and motivational conflicts are mutually reinforcing, thus exaggerating the effect either problem might separately have on early and longterm learning. These paradoxes are not defects in human learning to be remediated. They are fundamental properties of learning. If learning were not at least this complex, then

---

[1]This chapter was published in *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, edited by John M. Carroll, Cambridge, MA, MIT Press, pp. 80-111. We are grateful to John Whiteside and Phillis Reisner for critiquing an earlier version of this chapter. We also received helpful comments from our lab's reading group (John Black, Rich Catrambone, Bob Mack, Jean McKendree, and John Richards)

designing learning environments would be a trivial design problem (Thomas and Carroll, 1979). Our discussion is based on studies of the learning and routine use of word processing systems, and we speculate on potential programmatic tradeoff solutions to the paradoxes in this domain and generally.

2 The Active User

A colorful, and apt, image of the world of the new user of a computer system is found in the often quoted phrase of William James: "a bloomin' buzzin' confusion." People in this situation see many things going on, but they do not know which of these are relevant to their current concerns. Indeed, they do not know if their current concerns are the appropriate concerns for them to have. The learner reads something in the manual; sees something on the display; and must try to connect the two, to integrate, to interpret. It would be unsurprising to find that people in such a situation suffer conceptual—or even physical—paralysis. They have so little basis on which to act.

And yet people do act. Indeed, the typical pattern we have observed is that people simply strike out into the unknown. If the rich and diverse sources of available information cannot be interpreted, then some of these will be ignored. If something *can* be interpreted (no matter how specious the basis for this interpretation), then it will be interpreted. Ad hoc theories are hastily assembled out of these odds and ends of partially relevant and partially extraneous generalization. And these "theories" are used for further prediction. Whatever initial confusions get into such a process, it is easy to see that they are at the mercy of a diverging feedback loop: things quite often get worse before they get better.

Designers of computer systems and training technology surely would have liked things to have been different. The easiest way to teach someone something is, after all, to tell them directly. However, what we see in the learning-to-use-a-computer situation is that people are so busy trying things out, thinking things through, and trying to relate what they already know (or believe they know) to what is going on that they often do not notice the small voice of structured instruction crying out to them from behind the manual and the system interface.

A similar picture appears in considering the more experienced users of computer systems. Here, the best characterization is not one of striking out into the unknown. Rather, it is one of relying on the *known* to get things accomplished. Users tend to make use of the functions they know about to get a result, regardless of the efficacy of the method entrained. Designers of reference and help systems count on users to recognize opportunities for new methods, and to search out the information needed to implement them. Instead, users often figure out how to use what they already know to achieve new goals. They have little desire to explore new function, or to search out information, if they can use methods they are already comfortable with to achieve the same goal.

What is wrong? We would argue that the learning practices people adopt here are typical, and in many situations adaptive (Scribner, 1984). The problem in this particular learning situation is that learners are in the extreme. Each feature of a computer system may indeed have a sensible design rationale from the viewpoint of the systems' engineer, but this rationale is frequently far beyond the grasp of the new user, or indeed even a user familiar with the basic function of the system. "Word processor,": so far as we know, is not a natural concept. People who do not know about word processors have little, possibly nothing, to refer to in trying to actively learn to use such things. Innocence turns reasonable learning strategies into learning problems (Carroll and Mack, 1984).

3 The Production Paradox

It is good to want to get something done. One would only ever *want* to learn to use a new tool if one wanted first to get something done. But wanting to get something done can also be a problem, if one lacks the prerequisites: you have to learn to do in order to do. Merely wanting to use a new tool may be necessary but it is not sufficient.

3.1 Problems for New Users

Training and reference materials often are designed under the assumption that people who need to learn something will be willing to read about it, to practice skills in a sensibly structured sequence of exercises, and finally to assemble these conceptual and skill components into a mature competence and understanding. Further, it is assumed that when people seek to learn more about a domain they will again be willing to engage in these activities to develop and refine their expertise. But these assumptions are empirically unsound. Learners at every level of experience try to avoid reading. In structured practice, they often accidentally or deliberately get off track and end up in lengthy and complex tangles of error recovery or self-initiated exploration. (For details, see Carroll and Mazur, 1985; Mack, Lewis, and Carroll, 1983).

New users are not 'blank slates' for training designers to write upon. Indeed, the most accurate way to think about new users is as experts in other, non-computer domains. Secretaries trying to learn to use a word processor are not starting from ground zero in any relevant sense. They are experts at routine office tasks. (Unfortunately, but in all likelihood, they are far more expert than the designers of their word processing system!) The same point can be made for any other class of new users as they come to learn an application system designed to be a tool for them in their work. When a domain expert tries to use a tool designed specifically to support his or her work activities, the orientation is to do real work, *not* to read descriptions and instructions, or to practice step-by-step exercises.

New users tend to jump right in when introduced to application systems. If an operation is referred to in their training materials, they want to try it out at once. Rote descriptions and practice are resisted, and even when complied with, prove difficult to follow and assimilate. In a training system studied in Carroll and Mazur (1985), there is a list of icons in the training guide identifying the applications available, but users are not allowed to try the applications represented. After an hour or so of training, one learner complained: "I'm getting impatient. I want to do something, not learn to do everything." Half an hour later, he exclaimed: "I could have typed 3000 words by now!" Users become very frustrated when training "introduces" them to function but expects them to refrain from using it to perform a real task. Another learner balked when instructed by an on-line tutorial to read a passage but not do anything, exclaiming "I'm tempted to do it anyway and then see if I can get out."

Often, users respond to these desires to try things out, to get things done. But jumping the gun like this, and relying on exploratory learning strategies instead of the step-by-step rote structure of a manual or on-line tutorial, can be costly. Carroll and Mazur (1985) described a learner who explored a Wastebasket function by throwing away one and then another of the applications available on the system. This hypothesis testing approach did in fact enable her to correctly induce the Wastebasket operation, but at a fairly high price: she could not restore these applications. In other cases, the heuristic reasoning strategies users bring to bear do not even produce the correct conclusions. Another learner began drawing conclusions about work diskettes as soon as he saw the term: "Work diskette. Does that mean it won't work without a work diskette?" Later, he got an error message—"Work Diskette needs recovery; Use recovery task."—and confidently

concluded that he had initially placed the diskette in the wrong slot of the disk drive—which was a totally irrelevant consideration in this case. Loosely reasoned hypotheses are of course frequently wrong, yet they are attractive to users in that they allow rapid (albeit reckless), learner-directed progress on real tasks.

3.2 Problems for Experienced Users

For more experienced users, the Production Paradox is more subtle. It is not merely a matter of an urgent and yet premature need to produce, but rather a matter of balancing investment of time in learning versus throughput. The issue is one of whether it is worth the time to suspend throughput via already-learned, but perhaps inefficient methods, to engage in learning, which only in the long run might facilitate greater throughput.

Most computer systems and their reference libraries are designed with an inherently rational view of users in mind. They provide a range of function from basic to advanced, under the assumption that with experience, users will naturally acquire the procedural knowledge that most effectively fulfills their needs. Indeed, much of the early work in system evaluation has been guided by this assumption—that the asymptotic level of behavior is one relatively free of errors and inefficient strategies (e.g., Card, Moran and Newell, 1980; 1983). However, this assumption is called into question by recent work examining routine text editor use (Rosson, 1984a, 1984b).

In this work, users were both surveyed about their use of a text editor and monitored during their daily use of the system. The users varied in their experience with the editor, as well as in their experience with other editing systems, and in their job type. This editor provides a number of advanced functions for streamlining use, and one of the issues of interest was the extent to which such function is picked up by experienced users.

In general, we found that many users were not discovering and using function which could have made their jobs easier. A good example comes in the analysis of *macro* use; macros are stored programs which allow for extensions and modifications of the basic editor function. Many macros are available on a public disk, and one thing we examined was the usage of this “extra” function. We discovered that a large number of users had not incorporated any of the macros into their daily activities; this was true despite the fact that the most popular macros were ones providing very basic extensions to general editing functions, not routines serving special-purpose functions that could be viewed as appropriate only for sophisticated users. Thus, although there appeared to be a general need for this added function (analysis indicated that use of the macros was indeed associated with more rapid editing activity), the less sophisticated users (in this case, these were secretarial and administrative users) made virtually no use of these public macros. We speculated that this was due to the number of steps required to find out about and use the extra function, steps that might well seem too much trouble to a user focussed on generating end products.

Additional evidence that users fail to become experts is the appearance of “pockets of expertise” in user populations (Rosson, Gould and Grischkowsky, 1982; Draper, 1984): instead of becoming generalized experts themselves, users learn a basic set of knowledge, presumably relying on local experts to help them out when special needs arise. In some situations, this sociological phenomenon may work out nicely—where a work situation is structured such that specific users are assigned topics to master, and other users are made aware of when and whom to consult. However, in the general case, its success hinges on users' willingness to take the time to find and consult an appropriate expert when a particular need arises rather than making do with their more primitive skills. Unfortunately, we have no reason to believe that users will take the time and effort to find and consult human experts any more than they would a reference manual.

4 Approaches to the Production Paradox

As we stated in our introductory remarks, we do not see the Production Paradox as a just a problem to be solved *tout court*.. We do see several approaches to the paradox, but each is limited; indeed the three approaches we describe below are actually inconsistent with one another if taken to logical conclusions.

One approach is to ease the focus on tangible end products for users. While it may be natural for users of computing equipment to adopt an end-product motivational orientation, this may not be inevitably monolithic. Other motivational sets may be suggested or induced. A second approach is to minimize the consequences of an end-product focus by reducing the motivation necessary for learning. Learning that requires less motivation of the user might occur successfully even for users focussed principally on generating end products. As a third approach, we can try to design computing systems to better support the end-product focus: we can give the people what they want.

4.1 Attacking the End-Product Focus

At an extreme, the end-product focus can have the effect of subjugating *intrinsic* sources of reward (achievement, satisfaction of curiosity, control of the environment) to *extrinsic* sources of reward (printed output, hits in a data base query). Nevertheless, it is known that intrinsic rewards—when they can be made salient to people—can be far more potent motivators than extrinsic rewards (e.g., Greene and Lepper, 1979). Thus, one approach to the Production Paradox is to make the intrinsic rewards of successfully learning and using a computing system more salient to users.

Computer games have been investigated from this perspective. Malone (1981a,b) argues that these games can be effective learning environments for children by stimulating curiosity, fantasy and challenge as intrinsic sources of reward. One might imagine incorporating aspects of a game environment into the interfaces of ordinary application programs. For example, a version of the system could be made available for "playing;" learners would receive points according to their ability first to accomplish tasks at all, and second, to accomplish them in an optimal fashion. Carroll and Thomas (1982) suggested that routine applications could be presented under multiple interesting cover stories: the operator might be interacting with a flight simulator but in doing so actually managing a process control application.

Intrinsic motivation might also be effectively stimulated by incorporating more abstract elements of game environments into interface designs. One way that games motivate participants is by conjuring a world in which uncertainty is acceptable. In the well-known game of Adventure, players attempt to navigate a complex underground cave filled with assorted treasures and dangers. At the very outset though, they do not even know this, and as they go along they constantly encounter new and unexplained elements in the game. They remain in a discovery mode, never quite sure whether they are making progress or hopelessly lost. The game's interface dialog is structured to instill the attitude that this is all right, that uncertainty, discovery, and risk are inevitable. There is a rationale for this: people prefer activities whose outcomes of success and failure are uncertain (Weiner, 1980), and outcome uncertainty has been found to maintain greater interest in an activity (Berlyne, 1967). All this is in sharp contrast to typical application interface dialog, which implicitly projects an end-product focus to users, ruling out uncertainty, discovery and risk (Carroll, 1982a).

McKendree, Schorno and Carroll (1985) are currently experimenting with these ideas in a management setting. One version of their Personal Planner system provides prompting dialog that merely challenges the user to try things out. Other versions of the system provide more conventional prompting dialog that identifies correct and incorrect user

responses. We expect that providing increased feedback and specifically increasing the extent to which feedback is contingent on user goals and explicit behavior will increase achievement satisfaction.

Attacking the end-product focus directly has apparent limitations. When one ponders the proposals that have been made it seems evident that they will not work for all cases: some procedures might be too intricate for dialogs that rely exclusively on piquing the user's curiosity; some users might find it difficult to relax their end-product orientation. Moreover, the strategy can backfire: there are good effects of an end-product focus, and these too may be undermined by suggesting alternate motivational orientations—users might think of the system as a toy, or construe their real tasks in needlessly non-directive ways. In sum, it seems that other approaches will also be required.

4.2 Mitigating the Effects of an End-Product Focus

A second approach to dealing with the Production Paradox would be to make learning a less motivationally demanding task. There are two ways we might reduce the motivational "cost" of learning: make the learning safer and more risk-free, or make the relevant information easier to find. If trying out a new function is perceived as risk-free, a learner may be more willing to try it; it is less likely to interfere with the goal of producing something. Several design approaches have been taken in promoting the "safety" of systems during training. These fall into two classes—controlling the consequences of any given action by the user, and controlling the actions available to the user.

An extreme example falling into the first class is the "reconnoiter mode" proposed by Jagodzinski (1983). Here, users would be able to enter a mode that simulates the results of some proposed actions, but none of the activity has any permanent consequence for the task. The problem here, of course, is that the simulated activity does not move the user toward a goal in any real sense—it only allows him or her to "try out" something that when repeated outside of reconnoiter mode might have the desired effect.

Another approach to controlling the consequences of actions requires that each operation have an obvious inverse. Thus, if there is a command "drop" a file, there should be a complementary command to invert that operation and "pick up" a file—and the names of the commands should make this opposition salient (Carroll, 1982b). In such an environment, a person can try something out at a very low cost: if the result is not what was desired, the operation can be reversed, leaving no lasting consequence. A generalization of this approach is the so-called "undo" command, which is intended to be an inverse to any operation (or sequence of operations). Of course, this sounds simpler than it is: What is the appropriate "grain" of undo (a typed character, a command, a user task)? What state should you be in if three undos are executed in sequence? (See Gordon, Leeman, and Lewis, 1984). Currently there are only approximations of undo available.

A second class of solutions moves the control up a level, so that the options available are controlled, rather than their consequences. So, for example, one can design or retrofit an interface so that advanced functions and/or potentially errorful troublespots are unavailable to beginners (or more generally to users diagnosed as not yet ready for them). This is sometimes called a "staged" interface. Staging the presentation of function can limit the range of errors that inexperienced users can fall into, and therefore make experimenting with the system less risky. An example is a system that refrains from displaying parameter options when a workable default value is available (e.g., Smith, Irby, Kimbal, Verplank, and Harslem, 1982). While such "progressive disclosure" restricts the range of activity available to the user, it restricts in parallel the number of error conditions that can occur.

Work in our laboratory has developed a "training wheels" approach which combines the two classes of solutions (Carroll, 1983). A training wheels interface displays all of the function choices of a full function system, but disables advanced and provocative incorrect choices during the early stages of a user's learning. Making one of these choices merely

elicits a disablement message indicating that the selected function is not available during training, leaving the user in the same system state. Thus, while the learner is allowed to make an "error" (choosing an incorrect option), the consequences of the error are minimized.

We have carried out several experimental evaluations of the training wheels system. We asked learners to use either the training wheels system or the complete commercial system to learn to type and print out a simple document. The results of these studies were quite encouraging: learners using the training wheels system got started faster, produced better work, and spent less time not only on the errors that our design blocked, but on the errors we did not block—indicating a generalized facilitation of learning. Moreover, the magnitude of these advantages increase over the course of the experiment. Finally, the training wheels learners performed better on a system concepts test we administered after the experiment. (See Carroll and Carrithers, 1984).

An important complement to making learning safe is to make information about new function easy to find and understand. Users focussed on a particular task may be much more likely to enter into "learning mode" for a time, searching for new knowledge, if they believe that the knowledge will be easy to come by. One way to encourage this perception would focus on the reference materials available to a user. In the system studied in Rosson (1984a), the reference material (manual, reference card, and help screens) is organized in a linear, alphabetical fashion. This is true despite the fact that a large proportion of the 140 commands are never used interactively; they are issued only from within macros or stored procedures. As a result, a user looking for information on commands useful to him or her at the terminal must wade through a number of commands unlikely to ever be used by other than a very sophisticated user. A relatively inexpensive improvement to such materials would be to take into account actual usage patterns, rather than simply alphabetical ordering, in organizing the material.

A more expensive approach would be to use the computer as an active partner in learning. One might imagine a system that is able to determine the most effective path for reaching any given goal. When users recognized limitations in their current knowledge, they could query the system for a better method. An even more radical approach would be to allow the system to take the initiative, thus removing the requirement that users be motivated to look for learning opportunities (e.g., Shrager and Finin, 1982). Of course, such methods assume considerable intelligence on the part of the system, at a level yet to be provided in any real system. But even partial systems of this sort—a system that recognizes and understands only a limited set of common inefficient methods, for example—could contribute considerably to the goal of making the discovery and use of new function easier.

The various approaches to reducing the motivation required of the user that we have reviewed must be qualified in terms of their potential and current limitations. Many of these proposals have yet to be implemented for serious testing; reconnoiter mode and undo do not really exist in nonapproximate forms. And although some work has been done on systems able to observe users and make suggestions, at this point the domains studied have been rather restricted. Further, the proposals that have been implemented have generally not been studied very thoroughly in terms of their behavioral concomitants. While progressive disclosure makes a priori sense, we know of no empirical work examining its effectiveness. The training wheels approach has only been studied for a single computer application (word processing) and a single interface style (menu based control).

A host of behavioral questions remain open. How can systems transit between stages so as to most facilitate transfer of old knowledge and incorporation of new knowledge? What effects will the blocking of error consequences have on learning in the long run? Preliminary results from Carroll and Kay (1985) suggest that certain types of protective interfaces may have a negative effect on subsequent learning, indicating that the nature of

the "block" will be critical. If not presented carefully, a system that volunteers suggestions for improvement may be so disruptive as to wipe out any benefits to learning. And of course, there is the danger that by making learning too easy, we will make it too passive. If all problems are automatically analyzed, and suggestions for improvement directly provided, users' motivation to learn may be reduced even further, due to lack of challenge. Clearly, the issues here are complex, and it is unlikely that a single approach will be sufficient.

4.3 Designing for the End-Product Focus

We need not take the learner's focus on tangible products to be the problematic aspect of the Production Paradox. As a complement to designing around the end-product focus, that is, by making the system itself more intrinsically interesting or more safe to navigate and easy to learn, we can directly exploit the user's desire for a product by using it to drive learning. We can take the production bias as our starting point, and attempt to design systems and learning environments which actually depend on such an orientation.

An example is the Guided Exploration cards studied by Carroll, Mack, Lewis, Grischkowsky, and Robertson (1985). This training approach challenges the assumption that a linearly structured training manual format is the most appropriate training tool. The cards are more task-oriented than manuals in that each card addresses a particular functional goal that users can understand on the basis of their understanding of office tasks (irrespective of computers). The cards are designed to keep learners focussed on and involved in the learning task by being intentionally incomplete, often relying on hints. The cards are also more simply organized than manuals. Each card attempts to address its functional goal without reference to material covered on other cards. Finally, each card includes specific checkpoint information (to help learners detect and diagnose errors) and error recovery information (to help them get back on track). In addition, there is a general "What if something goes wrong?" card that described remedies that applied anywhere.

We performed experimental evaluations of Guided Exploration cards and state-of-the-art self-study training manuals. Learners using the Guided Exploration cards spent substantially less time yet still performed better on a transfer of learning post-test than learners using the commercially developed self-study manual. Taking learning efficiency to be achievement per unit time, we found that the cards were nearly 3 times as efficient as the manual. Moreover, qualitative analysis of learning protocols shows that the Guided Exploration cards worked as they were designed to work: they increased attention to the task, they encouraged learners to explore (and to do so successfully), they helped learners recognize and recover from errors, and they provided a better understanding of learning goals.

While this work was carried out for novice user groups, many of the objectives and techniques of the work should apply to more experienced users as well. Indeed, some systems now include "job aids" cards which are in many ways a generalization of Guided Exploration cards. Other possibilities can be imagined. For example, one might design reference material for advanced function in a text processing system that is organized according to real-world goals, rather than system function. In such an environment, a user's first exposure to a macro facility might come in discovering how to update a bibliography, with the result being a better association to actual needs. Work such as this is underway at University of California (e.g., O'Malley, Smolensky, Bannon, Conway, Graham, Sokolov, and Monty, 1983). The approach there has been to request goal-oriented comments from users during their interactions with a system, and to base recommendations for structuring reference materials on an analysis of these goals.

A separate area, now barely beginning, is the design of advice-giving systems—systems which are designed to help the user better articulate his or her own goal (Coombs and Alty, 1984). While this approach shares some similarities with the intelligent

help systems described in the previous section, a distinction exists in the level at which suggestions are made. Instead of offering advice about ways to "tune" a method for achieving a given result, these systems would attempt to assist a user in developing and organizing task-oriented goals from the start.

Like the other approaches we have considered, attempts to design for the end-product focus carry limitations. The proposals that have actually been implemented and studied all rely on the user to have appropriate goals. But clearly this assumption may not hold. If the user of Guided Exploration cards has a defective analysis of what he is trying to do, or if the analysis is at a different level than that provided by the card titles, this training approach may fail. And while we have suggested that intelligent problem-solving aids may contribute to this piece of the process, it is not at all clear that such systems can truly be developed. Finally, even if we assume the availability of appropriate goals, end-product approaches may ultimately impair breadth of learning. An organization of knowledge by particular task procedures may produce isolated modules of understanding that will be difficult to combine when novel tasks are encountered.

5 The Assimilation Paradox

If we knew *nothing* at all, it is difficult to imagine how we could make any progress at learning *anything* at all. Almost every new idea we learn comes to us through the graces of things we have already mastered. Nevertheless, this unavoidably good learning strategy cannot be the whole story—or indeed we would *never* learn anything truly new (Bartlett, 1932; Piattelli-Palmarini, 1980).

5.1 Problems for New Users

In discussing the Production Paradox, we made the point that even new users should be thought of as experts, albeit not in the computer domain. As such, their natural approach to a new tool is to try to use it—not simply to learn about it. As experts, new users also know a lot, though what they know is not necessarily relevant to the computer domain. Nevertheless, even a little knowledge can be a dangerous thing, particularly in a situation that invites the inference that it is relevant when it is not. This is a typical problem for new users of computer systems.

New users of word processing applications often try to understand their systems by reference to what they already know about typewriters (Carroll and Thomas, 1982). The metaphor of a typewriter can be useful in this context. But it can also lead to trouble. New users are often surprised that the Space Bar, Backspace key, and Carriage Return can change the format of their documents (including deleting characters and inserting special format characters into the document), as well as performing their more familiar typewriter functions. Learning is of course facilitated by the typewriter metaphor, but those places where the metaphor breaks down strain this advantage (see also Douglas and Moran, 1983).

We have seen similar problems in studying learners interacting with systems based on the "desktop" metaphor, where users are invited to make the comparison of an office system's directory objects to physical desktop objects like stationary, stationary pads, and folders. In one extended episode a person tried to create some test documents and then to store them. He started with a "folder," and was somewhat unsure what to do with it. He found operations for "making a stationary pad" and "tearing off stationary," which seemed consistent with his original goal, and tried them. Unfortunately, the interface metaphor strains a bit here: what he got was a stationary pad of folders from which he tore off a folder. This never quite sank in and for almost a hour he labored, selecting and making stationary pads, tearing off stationary, but never creating and storing a test document. The

episode finally produced only a confused question: "Why can I sometimes make a stationary pad and not tear off stationary and other times I can tear off stationary but not make a stationary pad?"

Another aspect of this problem is that by relying on metaphors, learners impair their ability to correctly anticipate the function of the system they are learning. For example, if one conceives of an office work station as a super-typewriter, then it is doubtful that one will tumble to an anticipation of its capabilities for formatting alternate fonts or for integrating text functions with graphics, database, and spreadsheet functions.

5.2 Problems for experienced users.

Again, for experienced users the problem is somewhat more subtle. Experienced users, by definition, either have experience on another system or systems, or they have prior experience on the current system. In both cases, they have established patterns of behavior and understanding that can interfere with the establishment of new patterns. Thus, people who have some experience with traditional half-duplex systems (which require pressing an Enter key to send buffered interactions to a host processor) may have trouble adjusting to full-duplex systems (in which each keystroke is an interaction with the processor, and in which Entering is not necessary): they expect to need Enter (much like a novice application user might expect word processing functions to behave like the typewriter's Space Bar and Backspace). In the survey work of Rosson (1984b), one of the most frequent classes of responses to the question "What things about <the editor> were especially difficult to learn?" described functions in the editor similar to, but slightly different from, those available in a previously used editor.

This type of learning problem is called *negative transfer*, the inhibition of later learning by prior learning of related material. The classic demonstrations of the phenomenon have been in the context of unrelated word lists (e.g., Underwood, 1957), and the effects are much less robust in real-world situations. However, it is clear that there is some disruption caused by the mapping of new command names or function keys to identical or similar function goals. Fortunately, while the interference can be frustrating during initial learning, it tends to be a short-lived problem, one that disappears if the learner discontinues use of the old system. And in general, its negative effects are more than compensated by the positive transfer that occurs for more general system concepts (Singley and Anderson, 1985).

There is another component of the assimilation paradox, however, that can have long-lasting effects. Prior knowledge not only can mislead users about how a system will work, but also it can in a sense put blinders on them, preventing them from fully anticipating the function available. This negative effect of prior knowledge can be especially debilitating, because often a learner may be completely unaware of the problem.

Evidence of these limiting effects of prior knowledge is seen in the routine editing behavior analyzed by Rosson (1984b). So, for example, one feature of the editor used by individuals in this study is a set of program function (PF) keys that can be used to speed up editing activities through assignment of frequently used functions to single keypresses. We discovered that larger function key repertoires were in fact associated with faster work rates; however, there was no tendency for key repertoire to be greater for the users most experienced with this particular system. Instead, use of the keys appeared to be a function of users' experience with other editors—specifically, experience with other *screen-oriented* systems seemed to encourage use of a greater number of PF keys. We believe that users' experience with other systems that rely heavily on function keys allowed them to recognize similar possibilities in this system. In contrast, users familiar only with more traditional command-driven systems were blinded to the many opportunities for extending their editing methods that these programmable keys provide.

6 Approaches to the Assimilation Paradox

As in the case of the Production Paradox, we will describe three different approaches to the Assimilation Paradox. One approach is to attack the assimilative tendency, to try to get people to deal with the system on its own terms, as it were, and not "as if" it were similar to something else they already know about. A second approach tries to compromise, simplifying the assimilation processes required of (or offered to) users, and therefore hopefully mitigating the force of the paradox. Finally, a third approach manipulates the assimilative opportunities for the user, deliberately highlighting inconsistencies between the current situation and the prior knowledge engaged in order to stimulate new learning.

6.1 Attacking Assimilation

As we remarked above, we doubt that the tendency for people to try to learn assimilatively can be altered. Nevertheless, effort could be directed at limiting the apparent similarity of a new experience to some prototype of past experiences. If we think of this relativistically we might expect that the trackball as a pointing device for text applications would be less susceptible to inappropriate assimilation than are step-keys for cursor pointing. The latter work very similarly to typewriter pointing via the Carriage Return, Space Bar, and Backspace. As a result, users might be less likely to begin with misconceptions about how a trackball works, and might be more likely to imagine, and to learn, novel aspects of its function.

One could also take an instructional approach, specifically directing the learning experience, advising against assimilation at least in certain cases. Interface metaphors, like the desktop and typewriter comparisons mentioned earlier, have only recently been incorporated into system training and documentation. Perhaps we need to qualify these explicit invitations to assimilative learning strategies. Halasz and Moran (1982) assume this, suggesting that users be provided with explicit models of the system, highly accurate and arbitrarily complete descriptions, usually in some abstract format, like a flow-chart or a graph (e.g., du Boulay, O'Shea and Monk, 1981; Moran, 1981; and Young, 1981). Indeed, they suggest that the user be warned against apparent metaphors and analogies in favor of this more literal conceptual model.

One thing that explicit system models have in common with analogical models is that both focus on concepts users need to master in order to have "understood" the system—as opposed to the operational procedures users must execute in order to use the system. This distinction suggests another approach to attacking assimilation, namely to forego attempts to encourage "conceptual" models at all—literal or analogical. Instead, user training and interaction could be aimed strictly at the procedures a user must execute, minimizing the conceptual material incorporated into these descriptions. An example is the teaching of recursive programming to students learning LISP. Pirolli, Anderson and Farrell (1984) found that students learned recursive programming much more quickly if directly provided with the programming procedures, than if provided with a conceptual model of recursion.

There are apparent limitations to approaches that try to eliminate assimilation. For example, can a designer really provide the user with a conceptual model that does not evoke assimilation, as advocated by Halasz and Moran? Carroll and Mack (1985) argue that when such a model is codified in any way (e.g., on paper as a graph or a chart), as it would have to be in order to function as an instructional tool, its interpretation will require prior knowledge about such representational formats and their characteristic interpretation. To the extent that this process is not automatic and determinate, it will be assimilative. Thus, there is no sharp dichotomy here and no way to eliminate assimilation *in toto*.

It is also not clear that strictly procedural materials can really be developed. The examples available so far either provide implicit conceptual content (e.g., trading on

understood conventions for flow-charts and graphs) or confound conceptual content *per se* with difficulty of material (Pirolli et al. reduced conceptual content by eliminating conceptual material on recursion, but in doing so eliminated a notoriously difficult concept as well). Perhaps the most severe limitation is that non-assimilative materials—if they can be developed—may be inappropriate for real users. Detailed descriptions and step-by-step directions, such as those one might find in a state-of-the-art self-instruction manual, are often close approximations to the explicit model approach of Halazs and Moran or to the pure procedural approach, but they are inconsistent with the propensities and capacities of actual learners trying to master computing systems (the Production Paradox).

6.2 Mitigating the Effects of Assimilation

A second approach to the Assimilation Paradox would be to accept assimilation as a given of learners and learning, and to try to design systems such that potential negative effects are minimized. One way to do this is to simplify the assimilative process by reducing the "assimilative gap." For example, one could take a strong view of metaphors in which the metaphor comparisons would have to be completely obvious and true: if the word processor appears to be "like" a typewriter, then it indeed would be operable in exactly the same way as a typewriter. Extending this idea, one could approach computer system design in general from the perspective of naive and intuitive expectations, designing the appearance and operation of systems so that they optimally accord with user expectations. What the user sees and predicts when introduced to the system is guaranteed to be correct by the designer.

One example of such an approach is the principle of *direct manipulation* described by Shneiderman (1983). He argues that wherever possible, the operations available to a user should be based on physical metaphors. So, for example, instead of issuing commands to modify textual material indirectly, word processors should allow users to move directly to text to be edited, and press buttons to produce the desired changes. This approach may not be extensible, however: it is by no means clear that there will be appropriate physical analogs to computer system function in the general case. Further, the approach is based on the fundamental assumption that a physical analog will indeed provide the best match to learners' naive expectancies about system function. The validity of this assumption has yet to be demonstrated.

A related approach incorporates naive expectations about system operation, but develops the understanding of users' intuitions through empirical observation, rather than through analysis and assumption. Mack (1984) provides an example of this approach, in his design of a word processing system based on a prior study of naive users' expectations about how such a system would work. In observing learners' interactions with the resulting system, however, Mack discovered that the goal of matching naive intuitions is a difficult one to meet. Intuitions are often very complex, inconsistent, even irrational. Further, not all users have the same intuitions, suggesting that designing an intuitive interface for the general case may be an impossible task. Mack's solution to this was to use behavioral observations as a starting point for his design, relying on an empirically-driven iterative process to point to modifications and additions not suggested by the initial corpus of naive expectations.

The problem of nonconvergent user expectations is not merely an issue of "early learning," something users outgrow. Mayer and Bayman (1981; see also Bayman and Mayer, 1984) asked students to predict the outcomes of keypress sequences on a calculator. All of the students were experienced users of calculators, but nonetheless their prediction responses varied considerably. For example, some predicted that an evaluation occurs immediately after a number key is pressed, some predicted that evaluation occurs immediately after an operation (e.g., plus) key is pressed, and some predicted that an evaluation occurs immediately after equals is pressed. The variability and the inaccuracy of

these predictions varied as a function of the student's prior training in programming, but it is open as to whether this is a smoothing effect of experience or of aptitude.

Other research has explored the possibility of addressing nonconvergent expectations by providing increased flexibility. Furnas, Landauer, Gomez, and Dumais (1984) analyzed naive users' intuitions about function names and found considerable variation among users. They developed a limited natural language facility with multiple levels of keyword synonyms for an information retrieval application (Gomez and Lochbaum, 1984). Good, Whiteside, Wixon and Jones (1985) used a similar approach in developing an electronic mail application. These interfaces could simultaneously meet the different expectations of different people.

A major limitation of approaches that seek to reduce the assimilative gap directly has been the size of the example systems developed. So, for example, all major examples of direct manipulation interfaces are small systems with relatively little function (experimental simulations or hobbyist personal computers for the most part). Intuitive design approaches have also addressed small-scale systems environments. There has yet to be a demonstration that this empirically-driven mapping between interface and intuitions will work for more complex real-world systems. Finally, there remains the possibility of a more general cost of eliminating the assimilative gap: if learners are no longer required to "work" for their new knowledge, they may fail to engage in the active processing critical to building a rich and flexible understanding.

6.3 Designing for Assimilation

A final approach to the Assimilative Paradox exploits the accommodation that can occur when assimilation fails. The terms assimilation and accommodation are associated with the theory of Jean Piaget (1954) in whose view the two are natural complements: learners assimilate experience into mental structures as possible, and then accommodate mental structures to experience as necessary. Computer interfaces and accompanying materials can be deliberately cast to stimulate direct comparisons between the current situation (the system itself so to speak) and whatever prior knowledge is engaged by the current situation, thereby highlighting key similarities and differences. These comparisons must be engineered to stimulate inferential processing, hypothesis testing, and active learning (Carroll and Mack, 1985; Whiteside and Wixon, 1985).

Consider the often referred to computer interface metaphor "a text editor is a super typewriter." Not all properties of a typewriter can be carried over to a developing concept of a text processor. Some can (the layout and character-transmission function of the keys); some cannot (character keys cannot straightforwardly be overstruck using a text editor); and some can be mapped from the typewriter base, but somewhat problematically (e.g., with respect to the storage of information, the tape recorder provides an alternate—and in some ways more accurate—metaphor). The comparison of a text editor with a typewriter carries *all* of these implications. The obvious similarities in function and form afford the metaphor in the first place: text editor learners almost never puzzle over what will happen when they strike a character key. In the context of such canonical and salient correspondences, the *dissimilarities* between the text editor and a typewriter become open questions—impelling further thought and leading then to further learning.

For example, keying two characters at the same location on a conventional typed page results in an overstrike. However, text editors do not produce overstrikes (in this way). They either insert (i.e., place the new character adjacent to the old one, and adjust the text line accordingly) or replace (i.e., place the new character where the old one was—deleting the old one). Conventional typewriters, of course, do not have an insert or replace capability; this is a clear dissimilarity in the metaphor. But this incomplete fit is not a functional a limitation on the metaphor. Salient dissimilarities—in the context of salient

similarities—stimulate thought and afford a concrete opportunity for developing an enhanced understanding of the electronic medium (e.g., the concept of dynamic storage).

Consider an example from a computer system that is based on the metaphor of a desktop. In this system objects and their manipulations are represented concretely (at least on the surface): for example, to create a new document file, a user is prompted to initiate an action roughly described as "tearing off paper," in the context of an icon representing a pad of paper. One user we observed took the prompt quite literally. He tried to execute the action of "tearing" by sweeping the cursor across the icon representing the paper. In fact, the metaphor is misleading in this case because actions applied to objects like files (or applications) must be selected in a more conventional fashion, from menus which describe the actions. Was the metaphor a failure? In fact, the experience was informative: the user understood that the desktop metaphor has certain boundary conditions, but more importantly he had a specific insight into the concept of selection and the fundamental role it plays in this interface (see Carroll and Mack, 1985).

The cognitive engineering challenge that inheres in designing for assimilation is formidable. In this approach, we design not merely to enhance simplicity, but to manage the presentation of complexity, to stimulate and guide an active problem-solving orientation and thereby to elicit better learning and more skilled and fulfilled routine performance. Much evidence indicates how these processes can go awry when they are not guided effectively. Thus, Scandura, Lowerre, Veneski, and Scandura (1976) described a student who came to the conclusion that the equals and plus keys on a calculator had no function by observing that the keys caused no visible change in the display. Norman (1983a) described learners who superstitiously pressed the clear key on calculators several times, when a single a keypress would do.

A key problem with designing for assimilation is determing how and when assimilative opportunities should be provided to learners. A classic approach has been to provide learners with advance organizers (Ausubel, 1960) that engage and direct appropriate prior knowledge. The idea is that making relevant prior knowledge available at the outset allows it to be brought to bear on the variety of problems that the user actually encounters, hence increasing the chance that the learning will be meaningful. This approach has been used in studies of learning computing (Foss, Rosson, and Smith, 1982; Mayer, 1976). It is difficult, though, for the designer to predict if the prior knowledge engaged by the advance organizer will still be salient to the user when an opportunity for assimilation occurs.

7 Is Effective Learning Possible?

In couching our discussion in the language of paradoxes, we have not intended to project the connotation of hopelessness, just of complexity. A paradox, in this sense, is a problem utterly refractory to a simple, comprehensive, logical treatment. Human learning is in this sense paradoxical. We do not believe that there is a simple, comprehensive, logical treatment of human learning in the offing, now or ever. But if the problem is complex, it is surely not hopeless. We have raised a number of suggestions as to how the paradoxes of learning can be addressed. However, as we have pointed out along the way, these solutions themselves have problems.

A premise of our discussion has been that the paradoxes of learning must be taken seriously, not as defects or errors but as fundamental patterns of learning. One could question this premise, and clearly there are no demonstrative arguments either way, but it seems to us that the inevitability of both paradoxes is plausible. If learners were less focussed on action, the Production Paradox could be avoided. But the cost would be that the connection between knowledge and performance goals would be far less stable, far less

direct. If learners were to rely less on prior experience, the Assimilation Paradox could be avoided. But here the cost would be a far less stable and direct connection between prior learning and new learning achievement. Both the paradoxes point to a single—and perhaps disturbing—fact of mental life: *adults resist explicitly addressing themselves to new learning* (see also Knowles, 1973; Kidd, 1977).

If we are correct, the paradox of the active learner entails specific *a priori* limitations on how much we can accelerate learning—limitations that apply irrespective of design intervention. Our only course, however, is to address the paradox through design, resigning ourselves to inevitable trade-offs (Norman, 1983b). In our discussion of approaches to the Production and Assimilation Paradoxes, we have considered solutions from three often conflicting perspectives: direct attacks on the underlying learning tendency, ways to limit the effects of the tendency, and attempts to take advantage of the tendency in a creative way (see Table 5.1).

Table 5.1
Summary of the Active User Paradox

	Approach	**Example**
Production Paradox:	Users focus on end products at the expense of prerequisite learning	
Attack:	Make learning the system intrinsically rewarding	Systems as games Performance feedback
Mitigate:	Make learning the system easy	Training wheels Undo
Design for:	Exploit the user's desire for a product by using it to drive learning	Guided Exploration cards
Assimilation Paradox:	Users apply prior knowledge even when it does not apply	
Attack:	Repress potential connections to prior knowledge	Explicit system models Performance feedback
Mitigate:	Make or describe the system as truly similar to something familiar	Direct manipulation Natural language
Design for:	Exploit the accommodation that can occur when assimilation fails	Incomplete metaphors

In discussing the Production Paradox, we suggested that one solution might be to try to reduce learners' production bias by making the system more intrinsically interesting. But it is not clear that all systems can be presented in this fashion, and even if they could, we can not be sure that the effects of such an approach would be uniformly beneficial—users might well come to see the system not as a useful tool, but rather as a toy to play with on occasion. The other solutions have their own problems: if we try to get around learners' motivation to produce rather than learn, by reducing the cost of learning—perhaps through error blocking and guided discovery of function—we run the risk of making learning too passive, or of setting up learning situations that may not transfer to subsequent usage scenarios. And finally, if we accept learners' end-product focus, and try to design systems and training materials to take advantage of it, we risk

either guessing at inappropriate goals for users or relying on users to structure their own goals, a task they for which they may be poorly prepared.

Our analysis of solutions to the Assimilation Paradox also pointed to limitations in each case. If assimilation is attacked directly, through designs too novel to assimilate, or through explicit instructions intended to eliminate assimilation, any learning that does take place may be quite fragile, due to its lack of connection with the learner's wealth of past experiences. And if we try to mitigate the problem by reducing the assimilative gap as much as possible, we may set ourselves up for designs that are trivial and offer little new function. Further, the "learning" involved here would again be extremely passive, requiring little cognitive effort on the part of the user, and might well lead to a less comprehensive understanding. Lastly, it seems attractive to contemplate designing for assimilation, attempting to incorporate concepts which have a natural link to prior knowledge, while stretching the mind by introducing inconsistencies at appropriate stages. However, the development of metaphors and other learning guidance for this is difficult, and only now beginning to have impact on user interface designs.

The paradoxes themselves are best thought of as indicative of fundamental orientations to learning, as properties of the learning. Of concern to us as design scientists, however, is the status of the solutions we have described. We have pointed to specific limitations in each case, and it is by no means clear that they can be resolved in any satisfactory way. It is important to note, though, that many of the limitations in these solutions stem from our analysis of the state of the art in interface design. We are not rejecting in principle the possibility that breakthroughs in design might speak to these problems in ways we cannot anticipate now. But frankly, we doubt it.

8 Learning and Design

We have argued that the issues associated with these paradoxes are complex enough, and the tradeoffs implied by the various solutions significant enough, that any single approach will not be sufficient. Rather designers will need to creatively sample from complementary or indeed even competing approaches. In this section, we briefly describe a method for undertaking such an eclectic process, illustrated by recent work on training manual design (for greater detail, see Carroll, 1984; Carroll and Rosson, 1985).

The first stage of design is analytic. It consists of the eclectic sampling of design elements implied by state-of-the-art empirical work, as well as by formal analyses of the design problem (e.g., as in Moran, 1981 and Reisner, 1984). An important constraint, though, is that the sampling be user-centered: it must be done in the context of specific considerations engendered by the particular design domain at hand: Who are the users? What are their special interests, needs, and difficulties? How does the particular system address these? None of this is front page design news. It makes perfect sense to have an understanding of what you are designing and who you are designing it for before you begin work—and to take advantage of whatever theoretical base is currently available (guidelines, user complexity metrics, etc.). Often, though, designers focus on a single approach to usability problems (e.g., the use of physical metaphors). We argue instead that they should be encouraged to incorporate complementary or even contradictory principles into their initial analysis.

The second stage of design involves detailed empirical testing of the subskills that will determine users' success with the system. This subskill analysis should also center on the activities of the intended users: What subskills are necessary for a typical user to perform a typical task on the system? Thus, a planning application intended for personnel managers to use in preparing salary and promotion plans, must be tested on a set of typical personnel managers who are asked to use it to prepare typical salary and promotion plans. Relevant

subskills might be an ability to describe the steps needed to accomplish a given task, a paraphrase understanding of menu selections and prompts, and an ability to recover from some of the more likely error states.

Because the goal of subskill testing is to rapidly gather detailed information about design elements, the testing should be qualitative rather than quantitative in nature, producing diagnostic rather than performance measures. Interpretation should focus on inadequacies in both the function provided by the system, and the interface for the user. For example, close observation of managers interacting with our hypothetical personnel planning application might reveal both that salary and promotion plans need to share data in particular ways (a function problem), and that typical personnel managers misunderstand certain specific prompts in the system dialog (an interface problem). This information must then be fed back into the design process so that the next iteration can remedy the problems. Subskill testing is inevitably a process of discovery and one in which the original design undergoes important changes.

While reiterative subskill testing guarantees a sort of local optimization of the design, it is not directed at providing an objective benchmark assessment of the final success of the design. Nonetheless it is useful in the end to know just how good a design really is, for example, relative to other contrasting designs or relative to particular usability goals. For example, can the final planning application for salary and promotion plans be used for routine tasks by typical personnel managers after one hour of training? Can it be learned faster than the current state-of-the-art alternative systems? Criterion testing is an important third stage of design, providing a means of empirically validating the results of the eclectic, iterative approach taken in the first two stages. We turn now to a description of a case study in which this three-stage approach was employed.

The development of the Minimal Manual (Carroll, 1984; Carroll, Smith-Kerker, Ford, and Mazur, 1985) exemplifies of the eclectic design process we have described. The initial design of the manual was a response to a number of observations about how naive users learn to use word processing systems, and many of these observations have already been discussed in illustrating the Production and Assimilation Paradoxes. But it is the design response to these paradoxes that is of particular interest here, because it reflects a sampling of approaches that might at first seem in conflict.

The Minimal Manual design work addressed the Production Paradox by simultaneously attempting to both *attack* and *support* the end-product bias. Thus, the manual included *On Your Own* sections that encouraged users to apply the procedures they had just worked through to new problems of their own choosing, leaving the instructions intentionally incomplete in an effort to promote intrinsic interest in the learning process through a performance challenge. This aspect of the design directly competed with another aspect, which was to *support* the end-product bias by streamlining prerequisites and focussing training activity on the production of real work output. In this case, two apparently conflicting strategies were consciously combined to yield a richer design solution.

The design approach adopted for problems stemming from the Assimilation Paradox was similar. A major feature of the Minimal Manual design was the removal of the conceptual material often found in training manuals, and a focus instead on concrete procedures. This constitutes an *attack* on assimilation through an emphasis on procedural rather than conceptual knowledge. However, this approach was combined throughout with instances of designing *for* assimilation through the careful use of metaphoric references. So, in describing procedures for removing unwanted line-end characters from the data stream, the manual specifically introduced the “blank line” metaphor as a way of identifying the problem. Importantly, though, it then went on to identify the difference between the metaphoric reference (a physical blank line) and the word processing problem (the presence of a line-end character). Ideally, pointing to such a divergence would serve not only to aid

learners in correcting this specific problem, but also to initiate processing leading to more general insights about the control of page layout via special formatting characters (Carroll and Mack, 1985).

After its initial design, the Minimal Manual underwent subskill analysis and testing. In some cases, this testing confirmed that the design principles had corrected problems observed with other training manuals. So, for example, the emphasis on procedural rather than conceptual material significantly reduced the problems learners encountered in achieving the important subskill of getting to the typing area. This activity requires traversing several menus, and can seem quite complex when described within a conceptual framework; the Minimal Manual reduced this to a few simple steps (in part by sheer deletion of conceptual material).

Importantly, though, the subskill testing also uncovered points at which the basic procedural approach was not optimal. One such point was the assignment of a document name, a fairly simple procedure, but one that appeared to confuse learners conceptually. This problem was treated by the addition of two brief conceptual sections. One developed a metaphor based on the practice of labelling physical office file folders to introduce the requirement that data objects like documents have names; the other developed a metaphor based on the practice of naming babies before they are born to introduce the requirement that data objects must be named before they can be used at all. This material, while it represented a departure from simple procedural descriptions, filled an important need. Without the early qualitative testing, the need may not have been discovered.

After the iterative design process, guided by subskill testing, the Minimal Manual underwent criterial testing (Carroll, Smith-Kerker, Ford, and Mazur, 1985). In one experiment, learners used one of five training methods (including two variations of the Minimal Manual) for up to seven full working days. The Minimal Manual proved to be substantially faster than the other manuals for the basic topic areas it covered—and to produce learning achievement at least as good as the other methods. The Minimal Manual only covered basic topics, where the commercial manuals covered advanced topics as well. In a later phase of the experiment, Minimal Manual learners were transferred to the advanced topics sections of a commercial manual. Notably, they still were substantially faster, but in this comparison their performance on learning achievement tests was better by a factor of eight. In sum, this experiment provided evidence that the final Minimal Manual design was an order of magnitude more effective than comparable state-of-the-art commercial manual designs, and as such represents a successful application of an eclectic design process.

The paradox of the active user seriously constrains designs of computing environments for people to use. We have presented an analysis of the paradox from cognitive and motivational standpoints, and we have described a variety of programmatic approaches to its resolution. Nevertheless, it is not our view that any cookbook engineering solution is likely to develop and "solve" this problem *tout court*.. Rather, we believe that this paradox inheres in human-computer interaction, that it derives from fundamental properties of human behavior and experience, and that addressing it through usability research and design will be an on-going project in the forseeable future. The social and technological urgency of this project entails an outstanding opportunity and challenge to cognitive science by placing it in a public spotlight in which the power of its theory and methodolology will be assessed by the absolute yardstick of practical efficacy.

References

Ausubel, D.P. 1960. The use of advance organizers in the learning and retention of meaningful verbal material. *Journal of Educational Psychology, 51*, 267-272.

Bartlett, F.C. 1932. *Remembering: An experimental and social study*. Cambridge: Cambridhe University Press.

Bayman, P. and Mayer, R.E. 1984. Instructional manipulation of users' mental models for electronic calculators. *International Journal of Man-Machine Studies, 20,* 189-199.

Berlyne, D. 1967. *Structure and direction in human thinking*. New York: John Wiley.

Card, S.K., Moran, T.P., and Newell, A. 1980. The Keystroke-Level Model for user performance time with interactive systems. *Communications of the Association for Computing Machinery, 23,* 396-410.

Card, S.K., Moran, T.P., and Newell, A. 1983. *The psychology of human-computer interaction.* Hillsdale, NJ: Erlbaum.

Carroll, J.M. 1982a. The adventure of getting to know a computer. *IEEE Computer, 15/11,* 49-58.

Carroll, J.M. 1982b. Learning, using and designing command paradigms. *Human Learning, 1,* 31-62.

Carroll, J.M. 1983. Presentation and form in user interface architecture. *Byte, 8/12,* 113-122.

Carroll, J.M. 1984. Minimalist training. *Datamation, 30/18,* 125-136.

Carroll, J.M. and Carrithers, C. 1984. Training wheels in a user interface. *Communications of the Association for Computing Machinery, 27,* 800-806.

Carroll, J.M. and Kay, D.S. 1985. Prompting, feedback and error correction in the design of a scenario machine. *CHI '85 Human Factors in Computing Systems Proceedings.* New York: ACM SIGCHI.

Carroll, J.M. and Mack, R.L. 1984. Learning to use a word processor: By doing, by thinking, and by knowing. In J.C. Thomas and M. Schneider (Eds.) *Human factors in computer systems*. Norwood: Ablex.

Carroll, J.M. and Mack, R.L. 1985. Metaphor, computing systems, and active learning. *International Journal of Man-Machine Studies, 22,* 39-57.

Carroll, J.M., Mack, R.L., Lewis, C.H., Grischkowsky, N.L., and Robertson, S.R. 1985. Exploring exploring a word processor. *Human Computer Interaction, 1,* 283-307.

Carroll, J.M. and Mazur, S.A. 1985. LisaLearning. *IBM Research Report, RC 11427*.

Carroll, J.M. and Rosson, M.B. 1985. Usability specifications as a tool in iterative development. In H.R. Hartson (Ed.), *Advances in human-computer interaction.* Norwood, NJ: Ablex.

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R. and Mazur, S.A. 1985. The Minimal Manual. *IBM Research Report,*

Carroll, J.M. and Thomas, J.C. 1982. Metaphor and the cognitive representation of computing systems. *IEEE Transactions on Systems, Man and Cybernetics, 12,* 107-116.

Coombs, M. and Alty, J. 1984. Expert systems: An alternative paradigm. *International Journal of Man-Machine Studies, 20,* 21-43.

Douglas, S.A. and Moran, T.P. 1983. Learning text editor semantics by analogy. *CHI'83 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI.

Draper, S.W. 1984. The nature of expertise in UNIX. In B. Schackel (Ed.), *INTERACT'84: Proceedings of the first IFIPS conference on human-computer interaction.* Amsterdam: North Holland.

du Boulay, B., O'Shea, T. and Monk, J. 1981. The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies, 14,* 237-249.

Foss, D.A., Rosson, M.B., and Smith, P.L. 1982. Reducing manual labor: Experimental analysis of learning aids for a text-editor. *Proceedings of Conference on Human Factors of Computer Systems*, Gaithersberg, MD: National Bureau of Standards.

Furnas, G.W., Landauer, T.K., Gomez, L.M. and Dumais, S.T. 1984. Statistical semantics: Analysis of the potential performance of keyword information systems. In J.C. Thomas and M. Schneider (Eds.) *Human factors in computer systems*. Norwood: Ablex.

Gomez, L.M. and Lochbaum, C.C. 1984. People can retrieve more objects with enriched key-word vocabularies. But is there a human performance cost? In B. Schackel (Ed.), *INTERACT'84: Proceedings of the first IFIPS conference on human-computer interaction*. Amsterdam: North Holland.

Good, M.D., Whiteside, J.A., Wixon, D.R., and Jones, S.J. 1985. Building a user-derived interface. *Communications of the ACM,27*, 1032-1043.

Gordon, R.F., Leeman, G.B., and Lewis, C.H. 1984. Concepts and implications of interactive recovery. *IBM Research Report*, RC 10562.

Greene, D. and Lepper, M.R. (Eds.) 1979. *The hidden costs of reward*. Hillsdale, NJ: Erlbaum.

Halasz, F. and Moran, T. 1982. Analogy considered harmful. *Proceedings of Human Factors in Computer Systems Conference*, National Bureau of Standards, Gaithersburg, MD.

Jagodzinski, A.P. 1983. A theoretical basis for the representation of on-line computer systems to naive users. *International Journal of Man-Machine Studies, 18*, 215-252.

Kidd, J.R. 1977. *How adults learn*. New York: Association Press.

Knowles, M.S. 1973. *The adult learner: A neglected species*. Houston: Gulf Publishing Company, American Society for Training and Development.

Mack, R.L. 1984. Understanding text-editing: Evidence from predictions and descriptions given by computer-naive people. *IBM Research Report*, RC 10333.

Mack, R.L., Lewis, C. and Carroll, J.M. 1983. Learning to use office systems: Problems and prospects. *ACM Transactions in Office Information Systems, 1*, 254-271.

Malone, T.W. 1981a. What makes computer games fun? *Byte*, *6/12*, 258-277.

Malone, T.W. 1981b. Toward a theory of intrinsically motivating instruction. *Cognitive Science, 4*, 333-369.

Mantei, M. and Haskell, N. 1983. Autobiography of a first-time discretionary microcomputer user. *CHI'83 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI.

Mayer, R.E. 1976. Some conditions of meaningful learning for computer programming: Advance organizers and subject control of frame order. *Journal of Educational Psychology*, *67*, 725-734.

Mayer, R.E. and Bayman, P. 1981. Psychology of calculator languages: A framework for describing differences in users' knowledge. *Communications of the Association for Computing Machinery, 24*, 511-520.

McKendree, J.E., Schorno, S.S., and Carroll, J.M. 1985. Personal Planner: The Scenario Machine as a Research Tool. Videotape demonstration, *CHI'85 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI (short version distributed by SIGGRAPH).

Moran, T.P. 1981. Command language grammar. *International Journal of Man-Machine Studies, 15*, 3-50.

Nielsen, J., Mack, R.L., Bergendorff, K., and Grischkowsky, N.L. 1986. Integrated software usage in the professional work environment: evidence from questionnaires

and interviews. *CHI'86 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI.

Norman, D.A. 1983a. Some observations on mental models. In D. Gentner and A.L. Stevens (Eds.), *Mental models*. Hillsdale, NJ: Erlbaum.

Norman, D.A. 1983b. Design principles for human-computer interfaces. *CHI'83 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI.

O'Malley, C., Smolensky, P., Bannon, L., Conway, E., Graham, J., Sokolov, J., and Monty, M. 1983. A proposal for user centered system documentation. *CHI'83 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI.

Piaget, J. 1954. *The construction of reality in the child*. New York: Basic Books.

Piattelli-Palmarini, M. 1980. *Language and learning*. Cambridge: Harvard University Press.

Pirolli, P.L., Anderson, J.R. and Farrell, R. 1984. Learning to program recursion. *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, CO.

Pope, B. 1985. A study of where users spend their time using VM/CMS. *IBM Research Report*, RC 10953.

Reisner, P. 1984. Formal grammar as a tool for analyzing ease of use: Some fundamental concepts. In J. Thomas and M. Schneider (Eds.) *Human factors in computing systems*. Norwood, NJ: Ablex.

Rosson, M.B. 1983. Patterns of experience in text editing. *CHI'83 Human Factors in Computing Systems Proceedings*. New York: ACM SIGCHI.

Rosson, M.B. 1984a. The role of experience in editing. In B. Schackel (Ed.), *INTERACT'84: Proceedings of the first IFIPS conference on human-computer interaction*. Amsterdam: North Holland.

Rosson, M.B. 1984b. Effects of experience on learning, using, and evaluating a text-editor. *Human Factors, 26*, 463-475.

Rosson, M.B., Gould, J.D., and Grischkowsky, N. 1983. *Field observations of IBM Displaywriter use*. Unpublished research, IBM Watson Research Center.

Scandura, A.M., Lowerre, G.F., Veneski, J., and Scandura, J.M. 1976. Using electronic calculators with elementary children. *Educational Technology, 16*, 14-18.

Scribner, S. (Ed.) 1984. Cognitive studies of work. *Quarterly Newsletter of the Laboratory of Comparative Human Cognition, 6*, Numbers 1 and 2.

Shneiderman, B. 1983. Direct manipulation: A step beyond programming languages. *IEEE Computer, 16*(8), 57-69.

Shrager, J. and Finin, T. 1982. An expert system that volunteers advices. *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh: Carnegie-Mellon University, 339-340.

Singley, M.K. and Anderson, J.R. 1985. The transfer of text-editing skill. *International Journal of Man-Machine Studies, 22*, 403-423.

Smith, D., Irby, C., Kimbal, R., Verplank, B., and Harslem, E. 1982. Designing the Star user interface. *Byte, 7*/4, 242-282.

Thomas, J.C. and Carroll, J.M. 1979. The psychological study of design. *Design Studies, 1*/1, 5-11.

Underwood, B.J. 1957. Interference and forgetting. *Psychological Review, 64*, 49-60.

Weiner, B. 1980. *Human motivation*. Chicago: Rand McNally.

Whiteside, J. and Wixon, D. 1985. Developmental theory as a framework for studying human-computer interaction. In H.R. Hartson (Ed.), *Advances in human-computer interaction.* Norwood, NJ: Ablex.

Young, R. 1981. The machine inside the machine: Users' models of pocket calculators. *International Journal of Man-Machine Studies, 15,* 51-85.