MINIMALIST COMPUTER DOCUMENTATION

A Study on Constructive and Corrective Skills Development

A.W. Lazonder

CIP-DATA KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Lazonder, Adrianus Wijnandus

Minimalist computer documentation: A study on constructive and corrective skills development / Adrianus Wijnandus Lazonder. – [S.l. : s.n.]. -Ill.
Thesis Enschede. – With ref. – With summary in Dutch.
ISBN 90-9007479-1
Subject headings: tutorial documentation / human-computer interaction.

Press: CopyPrint 2000, Enschede

MINIMALIST COMPUTER DOCUMENTATION

A study on constructive and corrective skills development

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de Rector Magnificus,
prof. dr. Th.J.A. Popma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 16 september 1994 te 15.00 uur

door

Adrianus Wijnandus Lazonder
geboren op 30 mei 1967
te Vlissingen

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. S. Dijkstra

Assistent promotor:
dr. H. van der Meij

Acknowledgements

Including acknowledgements in a minimal manual is inconceivable. Excluding them from this thesis would be just as inconceivable, for it would wrong all those who have directly or indirectly contributed to its realization. I therefore would like to express my gratitude to everyone who has assisted and encouraged me during the past four years. In particular, I would like to mention the following people.

First of all, my supervisor Sanne Dijkstra, who has created a scientific environment in which I could carry out this project according to my own insights.

I am greatly indebted to Hans van der Meij, my assistant-supervisor. Without his intellectual stimulation, his constructive remarks (as well as the corrective ones) and his unremitting enthusiasm this thesis would not have come about.

I gratefully acknowledge the help of Jules Pieters and Jeroen van Merriënboer. They were always willing to discuss my work and were kind enough to read and comment upon the articles that underlie this thesis.

I wish to thank John Carroll, for his inspiring ideas on designing computer documentation and for his valuable comments on the first draft of this thesis.

I would also like to express my gratitude to those who have assisted me in designing and conducting the experiments. I particularly would like to thank Pieter van der Poort and Maria Driessen, who critically reviewed the first version of the manuals; Vincent Oude Griep for designing and adapting the registration program that was used in the experiments; Christine Apperloo, Tim Remmers, Yvonne de Thouars and Ingrid Zemmelink, who advised and assisted me during the experimental sessions; and, again, Tim Remmers for developing the ERR-system that was used in the third and fourth experiment.

In preparing the final version of the text, I gratefully acknowledged the linguistic corrections of Afineke de Vries. I would also like to thank Pauline Teppich for her suggestions on the styling of the manuscript – and for constantly reminding me to "work hard".

Throughout the project, I truly appreciated the moral support of my roommate Jelke van der Pal. I have benefited a great deal from our discussions and I will remember our loose talks with pleasure; they made my stay at the department of Instructional Technology a pleasant one.

I would also like to thank Charlotte Laret, who was always willing to listen to my problems and usually pointed me to an obvious, yet overlooked solution.

Finally, special thanks to Petra. I know it is conventional to express gratitude to one's partner for his or her patience and forbearance. I also know that my gratefulness comes from the heart.

Ard Lazonder

July 1994

Contents

CHAPTER 1
General introduction

1.1 Problem definition

The apparent ease with which people use devices such as compact-disc players, micro wave ovens, and fax machines tends to overshadow the difficulties they had in learning how to use them. These difficulties would not aris if every device were to embody its own use. But, in the era of technology, contrivances have become so specialized that a match between function and operation is no longer possible. As a consequence, people often encounter problems when first using a piece of technical equipement.

Some of these difficulties are caused by the apparatus. Most technical devices are operated by pressing (a series of) buttons. The function of these buttons is symbolized by icons; their effect often appears in the form of an encrypted message on a display. To first-time users this type of symbolic interaction turns out to be a major problem source: even with proper instructions they find it difficult to understand how the device's input and output relate to their own goals and actions.

Another major source of difficulties is the product's documentation that, quite ironically, was meant to help. Assembly instructions, installation guides, and user manuals often are, as Pirsig (1974) wrote "...full of errors, ambiguities, omissions and information so completely screwed up you [have] to read them six times to make any sense out of them..." (p. 24).

In this respect, the microcomputer is no exception. Its appearance (i.e., the interface) does not reflect its functions. Nor do the cryptically labeled commands. Moreover, meaningless combinations of keys have to be pressed and their effect often does not become clear from looking at the screen – if one knows what to look for anyway. And, when looking for help, the manual that came with the computer or the software often turns out to be confusing rather than clarifying.

People's struggle in getting to know a computer has often been the subject of humorous anecdotes. Less often it has been the basis for scientific inquiries. But, as more and more people start using computers, studying how computer knowledge and skills are acquired becomes increasingly important. The present research therefore deals with how people learn to use a computer program. The general research problem for this thesis is defined as:

> *What instructional methods should be applied in teaching people how to use a computer program?*

This chapter starts with a description of the central components of this problem definition (i.e., target audience, software package, and instructional material). Then the general problem statement is further specified into a research question and the instructional design theory used to address this question is introduced.

1.2 Target population

In training computer related skills, learners often are classified according to their level of prior computer experience. A common classification is that of novices, beginners, intermediates, and experts (e.g., Brockmann, 1990; Chin, 1986). In this thesis the primary focus will be on the adult computer novice[1]. The key features of this user group (i.e., level of computer experience, prior knowledge and skills, learning preferences) are described below.

Most commercially available software packages are designed for a broad population of users, differing considerably on variables like age, educational background, cognitive ability, and learning style. The novices among these users share one characteristic: they have little or no computer experience. Based on a review of the literature, both Allwood (1986) and Schriver (1986) concluded that this qualification has been given different meanings, causing the term 'novice user' to denote anything from users without any previous computer experience to users approaching the expert status. Consequently, a more explicit definition is called for. In this thesis, novices are defined as users who have worked with a computer for less than 50 hours, and, who have no experience with the software that is central in training.

Novice users thus lack nearly all computer-related knowledge and skills. They have little or no knowledge of how a computer works, and they are not familiar with its jargon. The same goes for the software. New users know little of its operation, and they lack most necessary skills to work with it (e.g., Allwood, 1986; Mack, Lewis & Carroll, 1987; Sein & Bostrom, 1989). Their interactions with the program can therefore be characterized as a problem solving situation (Allwood, 1986; Moran, 1981).

Notwithstanding their scanty experience in working with a computer, adult computer novices do not come to their learning task as blank slates. Computer

[1] the terms novice, first-time user and new user are used interchangeably.

application programs are always meant to be used in some task domain (e.g., accounting, clerical work, statistics), and adult novices often have considerable knowledge and skills in these domains. In that respect, they bring a variety of experience to the training scene. For instance, graphical designers who want to learn a desktop publishing package are familiar with the graphical terminology and are skilled at paging up texts, designing posters or drawing comprehensible graphs. But, what they do *not* know is how to use the computer for this purpose.

Their knowledge of the underlying task domain provides new users with clearly outlined ideas and expectations about the training. They are not learning for learning's sake. Rather, they are trying to use a tool, a tool they believe will facilitate their work and activities. As a result, they are highly motivated to achieve personal goals (Carroll, 1990; Knowles, 1984). Few (if any) of these goals refer to learning about the program. Thus, novice users show very little interest in getting to know and understand all the facts of a program. Rather, they prefer to act, to do things with the program. They only want to gain understanding of the components of the system and their relations if it helps them achieve their goals. In short, they want to 'read to learn to do' rather than to 'read to learn' (Redish, 1988).

The learning preferences of novice users contrast with their information needs. This dissimilarity has become known as the paradox of sensemaking: first-time users show a strong desire to act in order to learn, but, at the same time, their actions, errors, and misconceptions show that they need to learn in order to be able to act (Carroll & Rosson, 1987). This paradox implies that designing workable training materials for novice users requires a constant balancing between their desire to act and their need for knowledge. More specifically, novice users' training needs can be met by presenting them only the prerequisite knowledge to cope with the activity and allowing them to learn more from engaging in it.

1.3 Software package

To a large extent, research on novice computer users has concentrated on word processing (e.g., Allwood & Eliasson, 1987; Bovair, Kieras & Polson, 1990; Carroll & Carrithers, 1984; Charney & Reder, 1986; Czaja, Hammond, Blascovich & Swede, 1986; Douglas & Moran, 1983; Foss, Smith-Kerker & Rosson, 1987; Frese *et al.*, 1988; Gomez, Egan & Bowers, 1986; Singley & Anderson, 1987). The choice for this kind of software is obvious. Word processing is a very general use of computer systems (e.g., Kalén & Allwood, 1991; Penrose & Seiford, 1988). For many people, the use of a word processor is their primary interface with a computer. More importantly, in most instances, word processing is the novice user's introduction to using a computer.

Given the widespread use of word processors by novice computer users, it is interesting to study how the operation of these applications is learned. The key question here is which factors influence a person's success in learning to use these systems. The answer to this question may be used as input to the design of future word processing packages and to the design of training programs and materials.

Word processing is a procedural task that can be described as a set of skills that are performed in a relatively fixed order. Learning word processing comes down to learning these skills and the knowledge that is relevant to it. More specifically, the information to be learned can be classified as knowledge and skills that (a) specifically relate to the word processor at hand, (b) may be transferred (either positively or negatively) from the use of similar devices (e.g., a typewriter), and (c) relate to the underlying task domain (i.e., writing and styling text). This classification is illustrated in Table 1.1.

In learning word processing, the categories from Table 1.1 differ with regard to their relevance. Most of the knowledge and skills pertaining to the styling of text and to the writing process in itself are not conditional to learning to use a word processor. Hence, the instruction can, and should, start from the users' prior knowledge and skills of the task domain. Furthermore, the instruction can partly fit in with the knowledge and skills that are familiar from typewriting. Knowledge and skills that transfer positively from typewriting to word processing need not be addressed (e.g., typing, the use of the TAB key); additional explanation should only be given in case of negative transfer to prevent users from making errors (e.g., using the space bar to move across the screen instead of the cursor keys; see Allwood & Eliason (1987) and Douglas & Moran (1983) for an overview of typewriter analogy errors). As novice users have no command of the knowledge and skills required to operate the word processor, all of this information should be covered in the instruction.

Clearly, instruction for novice users should not cover every possible aspect of the word processor. Only basic tasks in operating the program (i.e., initial skills) should be addressed. In word processing, these fundamental activities are characterized by the following cycle: (1) starting the program, (2) typing text or retrieving an existing document, (3) formatting, saving, and printing the document, and (4) ending the program (cf. Boeke, 1990). A complete overview of the basic word processing tasks is presented in Appendix 1.

Table 1.1
Illustrative classification of knowledge and skills in word processing

	Knowledge	Skill
Word processor	• terms like 'cursor', 'printer', 'macro' • text looks differently on paper than on screen • hidden codes can be revealed	• moving or copying a block of text • using the thesaurus • saving a document • creating macros and styles
Typewriter	• line transport within paragraphs • the QWERTY keyboard • the function of the TAB key and the SHIFT key	• inserting a blank line • changing the line spacing • typing text (upper-case and lower-case letters) • underlining text
Task domain	• the layout of a letter • terms like 'footnote', 'typeface', 'margins' • the difference between 'subscript' and 'superscript'	• writing in a terse style • using grammar correctly • styling a text

Note. This table merely illustrates the classification of knowledge and skills in word processing. It is therefore not meant to be exhausting.

As their learning preferences indicate, novices want to engage in real, coherent tasks instead of contrived drill and practice exercises (Cuthbert, 1992; Wright, 1988). The instruction should comply with this desire by focussing on how the basic word processing tasks are to be performed. Moreover, the sequencing of these tasks in the instruction should be in accordance with the above-mentioned cycle. In a way, users should see themselves using the program while reading the training material.

1.4 Instructional material

In general, there are three approaches to teaching basic computer skills. Still the most current (and perhaps the most obvious) one is by paper documentation. The second way to train users is by presenting information on the screen. Examples of this increasingly used technique are help-screens, on-line documentation, and computer assisted instruction. A third possibility is by an instructor-based training program. Because of its clear benefits (e.g., low costs, high availability, easy accessibility), manufacturers often prefer paper documentation. Its widespread use is likely to expand even further, as written documentation is increasingly considered a necessity to satisfy legal constraints concerning product liability. Therefore, the focus will be on paper documentation (i.e., manuals).

There are many different types of manuals, each of which has a specific audience or covers a specific function. Two main categories are being distinguished: instruction and reference. Instruction manuals teach people how to operate a system or how to use a program. Reference documentation gives users key definitions, facts, commands, and codes they cannot be expected to memorize (Weiss, 1991). Instruction manuals are further classified into tutorials and user guides. Tutorials have an educational intent. They teach basic skills to users who have never used the product before. User guides are designed for the more experienced user. They contain exhaustive explanations of simple to very complex tasks associated with using a piece of hardware or software.

Matching the document type with the user group's information needs indicates that novice users are best served by a tutorial manual. This type of manual is defined as "instructional information which familiarizes users with a new piece of hardware or software and teaches them the rudiments of their use" (Bradford, 1984, p.167). Effective tutorials should thus present users with the basics they need to get started with a program.

Tutorial documentation does not only teach certain basic skills; it also stands alone in doing so. A tutorial is therefore a typical example of self-instruction material. It is an instructional tool that users should be able to use independently, without any supplementary assistance. This requires tutorials to be flexible enough to be self-explanatory to a wide variety of users. The absence of an instructor implies further that a tutorial should motivate users and maintain their motivation throughout training.

To meet these requirements, it is useful to follow a systematic approach in designing and writing tutorial documentation. In general, there are two approaches, differing with regard to their primary focus (Rowntree, 1986). In a subject-matter oriented approach the primary focus is on the content of the instruction. In a learner oriented approach the designer first looks at the features of the target audience. As tutorials are intended for learners with divergent backgrounds, information needs, and learning preferences, an instructional design theory that takes these features as a starting point for design is called for. Developing (a part of) such a design theory then becomes the central theme of this thesis.

1.5 Instructional design theory

Having defined the principal components of learning to use a computer program, the problem description is reformulated into a more detailed research question:

> *What instructional design principles should be applied in paper tutorial manuals for teaching novice computer users elementary word processing skills?*

To answer this question, a design theory known as *minimalism*[2] is used (Carroll, 1984*a,b*, 1990). Dating from the early 1980s, minimalism is a relatively new design theory, developed especially for designing self-instruction materials with which users can learn to use computer programs.

The minimalist approach is learner oriented. In this respect, Kerr and Payne (1994) stated that "the term 'minimalist' denotes a broad instructional philosophy, in which the design of instructional materials seeks to interfere with the learner's purposes and motivations as little as possible." (p. 4). In turn, Carroll (1990) outlined the rationale for the minimalist approach as follows: "The key idea in the minimalist approach is to present the smallest possible obstacle to learners' efforts, to accommodate, even exploit, the learning strategies that cause problems for learners using systematic instructional materials. The goal is to let users get more out of the training experience by providing a less overt training structure." (pp. 77-78).

Following from this rationale, minimalist instruction is action oriented in that it offers learners little to read and much to do. Minimalist training material invites users to get started immediately on real and meaningful tasks and frequently encourages them to explore the program. Throughout training, only information that is essential for working with the program is explained. This explanation is always motivated in the task. That is, it is presented immediately before or after the relevant action steps describing what something 'does' rather than what it 'is'. In addition, ample support is given for recovering errors that may occur during task execution. This information too is presented 'in context': it appears in the instructional text, directly after the actions it refers to. (A more detailed description of the minimalist approach is presented in chapter 2).

The minimalist approach has revealed some promising results. Research has shown that people who used a *minimal manual* (i.e., a manual designed according to the minimalist approach) learned to use a word processor in 40% less time with 58% better retention of skills than people who used the

[2]in the literature, this design theory is also referred to as "the minimalist approach".

commercially available control manual. Moreover, they made 20% fewer errors and were significantly more efficient in error-recovery (Carroll, Smith-Kerker, Ford & Mazur-Rimetz, 1987).

The minimalist approach and its claims of effectiveness have raised some criticism. Perhaps one of the most important arguments here is that minimalist instruction is not clearly defined (yet). A gradually changing set of features has been imposed on minimalist training materials. A related problem is the absence of explicit guidelines for designing minimalist training materials, making it difficult to ascribe the above results to distinct features of the instruction. In addition, there is little sound empirical evidence on the functionality of the minimalist approach. Although the first experimental results were quite impressive, the replicability and construct validity of these studies call for some concern.

1.6 Overview of the thesis

The work that is reported in this thesis attempts to solve these problems. Chapter 2 contains an operational definition of the minimalist approach. The succeeding chapter reports on an investigation into the functionality of this approach. In this experiment, a minimal manual that was designed according to the principles presented in chapter 2 was compared with a state-of-the-art tutorial.

The outcomes of this experiment served as a starting point for the research presented in chapter 4 to 6. The study in chapter 4 is an early attempt to reveal the effect of a single minimalist principle (i.e., error-information) on learning outcomes. In chapter 5 the preconditions for error-information to affect the users' performance during practice are identified and examined in an exploratory fashion.

The effect of error-information on user behavior is also the subject of the experiment described in chapter 6. This experiment was designed according to the requirements identified in chapter 5. The study is a detailed investigation into the effect of error-information on users' learning activities and learning outcomes. Chapter 7 presents a general discussion of the work in this thesis.

References

Allwood, C.M. (1986). Novices on the computer: A review of the literature. *International Journal of Man-Machine Studies, 25*, 633 - 658.

Allwood, C.M., & Eliasson, M. (1987). Analogy and other sources of difficulty in novices' very first text-editing. *International Journal of Man-Machine Studies, 27*, 1 - 22.

Boeke, H. (1990). *Werken met WordPerfect* [Working with WordPerfect]. Amsterdam: GW Boeken.

Bovair, S., Kieras, D.E., & Polson, P.G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. *Human-Computer Interaction, 5*, 1 - 48.

Bradford, A.N. (1984). Conceptual differences between the display screen and the printed page. *Technical Communication*, *3*, 13 - 16.

Brockmann, R.J. (1990). *Writing better computer user documentation: From paper to hypertext* (2nd. edition). New York: Wiley.

Carroll, J.M. (1984*a*). Minimalist design for active users. In B. Shackle (Ed.), *Interact'84: First IFIP Conference on Human Computer Interaction* (pp. 621 - 626). Amsterdam: Elsevier.

Carroll, J.M. (1984*b*). Minimalist training. *Datamation, 30*, 125 - 136.

Carroll, J.M. (1990) *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill*. Cambridge: MIT Press.

Carroll, J.M., & Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors, 26*, 377 - 389.

Carroll, J.M., & Rosson, M.B. (1987). Paradox of the active user. In J.M. Carroll (Ed.), *Interfacing thought: Cognitive aspects of human-computer interaction* (pp. 80 - 111). Cambridge: MIT Press.

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., & Mazur-Rimetz, S.A. (1987). The minimal manual. *Human-Computer Interaction, 3*, 123 - 153.

Charney, D.H., Reder, L.M. (1986). Designing interactive tutorials for computer users. *Human-Computer Interaction*, *2*, 297-317.

Chin, D.N. (1986). User modeling in UC, the UNIX consultant. In M. Mantei & P. Orberton (Eds.), *Human factors in computing systems-III: Proceedings of the CHI'86 conference* (pp. 24 - 28). Amsterdam: Elsevier.

Cuthbert, P. (1992). Concepts not keystrokes - Can we improve our teaching of word processing? *Computer Education, 72*, 2 - 5

Czaja, S., Hammond, K., Blascovich, J.J., & Swede, H. (1986). Learning to use a word processor as a function of training strategy. *Behaviour and Information Technology, 5*, 203 - 216.

Douglas, S.A., & Moran, T.P. (1983). Learning text editor semantics by analogy. In A. Janda (Ed.), *Human factors in computing systems: Proceedings of the CHI'83 conference* (pp. 207 - 211). Amsterdam: Elsevier.

Foss, D.J., Smith-Kerker, P.L., & Rosson, M.B. (1987). On comprehending a computer manual: analysis of variables affecting performance. *International Journal of Man-Machine Studies*, *26*, 277 - 300.

Frese, M., Albrecht, K., Altmann, A., Lang, J. Von Papstein, P., Peyerl, R., Prümper, J., Schulte-Göcking, H., Wankmüller, I. and Wendel, R. (1988). The effect of an active development of the mental model in the training process: Experimental results in a word processing system. *Behaviour and Information Technology, 7*, 295 - 304.

Gomez, L.M., Egan, D.E., & Bowers, C. (1986). Learning to use a text editor: Some learner characteristics that predict success. *Human-Computer Interaction, 2*, 1 - 23.

Kalén, T., & Allwood, C.M. (1991). A survey of the training of computer users in Swedish companies. *Behaviour & Information Technology, 10*, 81 - 90.

Kerr, M.P., & Payne, S.J. (1994). Learning to use a speadsheet by doing and by watching. *Interacting with Computers, 6,* 3 - 22.

Knowles, M.S. (1984). *The adult learner: A neglected species* (3rd. edition). Houston: Gulf.

Mack, R.L., Lewis, C.H., & Carroll, J.M. (1987). Learning to use word processors: Problems and prospects. In R.M. Baeker & W.S. Buxton (Eds.), *Readings in human-computer interaction: A multidisciplinary approach* (pp. 269 - 277). Los Altos: Morgan Kaufmann.

Moran, T.P. (1981). An applied psychology of the user. *Computing Surveys, 13*, 1 - 11.

Penrose, J.M., & Seiford, L.M. (1988). Microcomputer users' preferences for software documentation: An analysis. *Journal of Technical Writing and Communication, 18*, 355 - 366.

Pirsig, R.M. (1974). *Zen and the art of motorcycle maintenance*. New York: William Morrow.

Redish, J.C. (1988). Reading to learn to do. *The Technical Writing Teacher, 15*, 223 - 233.

Rowntree, D. (1986). *Teaching through self-instruction.* London: Kogan Page.

Schriver, K.A. (1986). *Designing computer documentation: a review of the relevant literature* (Tech. Rep. No. 31). Pittsburgh: Carnegie-Mellon University, Communications Design Center.

Sein, M.K., & Bostrom, R.P. (1989). Individual differences and conceptual models in training novice users. *Human-Computer Interaction, 4,* 197 - 229.

Singley, M.K., & Anderson, J.R. (1987). A keystroke analysis of learning and transfer in text editing. *Human-Computer Interaction, 3,* 223 - 274.

Weiss, E.H. (1991). *How to write usable user documentation* (2nd. edition). Phoenix: Oryx Press.

Wright, P. (1988). Communicating with the user. In N. Heaton & M. Sinclair (Eds.), *Designing end-user interfaces* (Pergamon State of the Art Report No. 15:8 pp. 121 - 129).

CHAPTER 2

The minimalist approach to tutorial documentation[3]

In an effort to improve tutorial documentation for first-time computer users, Carroll and his colleagues at IBM designed a so-called minimal manual. Unfortunately, ever since its development it has been somewhat unclear what is meant by a minimal manual. In this chapter an attempt is made to provide an operational definition of the minimalist approach to tutorial documentation. For that purpose, the origin and characteristics of Carroll's minimal manual are reviewed. Directions for research on minimalism are identified.

2.1 Introduction

Until the 1970s, computers were used almost exclusively by engineers and programmers. This selected audience of experts was assumed to be capable of and willing to invest time and effort in really getting to know a program. Computer documentation was developed accordingly; document designers hardly needed to worry about instructional design issues such as presentation forms, principles of sequence, instructional objectives, and motivation (Carroll & Carrithers, 1984; Maynard, 1979).

With the advent of the microcomputer in the late 1970s, computer programs for office and home use became available, and a different audience emerged. This audience had no background in computer science, programming, or electronics, and it had little or no inclination to gain fundamental understanding of a program. Their learning preferences were relatively simple: they wanted to learn to use a program as fast as possible.

Documenters at first did not realize that this shift of training needs also changed the demands made on their work. It was only after their companies found out that a good manual was a selling point for software products (Sohr,

[3]based on: (a) Lazonder, A.W., & Van der Meij. H. (1992). *Towards an operational definition of the minimal manual* (Tech. Rep. IST-MEMO-92-02). Enschede: University of Twente, Department of Instructional Technology; and (b) Van der Meij, H., & Lazonder, A.W. (1993). Assessment of the minimalist approach to computer user documentation. *Interacting with Computers, 5*, 355-370.

1983) that they began to take this issue seriously. In the early 1980s, documentation thus became a specialized job, which, among others, directed technical writers toward producing manuals especially for novice users. This, in turn, led to a range of handbooks on how to develop effective computer documentation (e.g., Brockmann, 1986; Crandall, 1987; Foehr & Cross, 1986; Grimm, 1987; Price, 1984; Steehouder, 1989).

In 1984[4], Carroll's *minimal manual* signaled a distinct new approach to computer user documentation (Carroll, 1984*a*,*b*). Carroll introduced a typically short tutorial that differed from the then-existing manuals in that it complied with novice users' desire for quick and self-controlled hands-on experience. This manual was 'minimal' in the sense that it was a flexible guide in which concise instruction was alternated with ample opportunity to engage in meaningful interactions with the program.

Practical findings indicated that the minimal manual was highly successful. Apparently, it succeeded in meeting the learning preferences of first-time users: they frequently expressed their satisfaction with the manual for it allowed them to learn to use the program by actually working with it instead of reading about it (e.g., Arnold, 1988; Black, Bechtold, Mitrani & Carroll, 1989; Carroll, Smith-Kerker, Ford & Mazur, 1986). Other studies have shown that the use of a minimal manual also improved the users' skills in operating the program (e.g., Carroll, Smith-Kerker, Ford & Mazur-Rimetz, 1987; Gong & Elkerton, 1990; Vanderlinden, Cocklin & McKita, 1988).

Unfortunately, it is not entirely clear which instructional design principles contribute to the success of the minimal manual. Although its major features have been properly described (Carroll, 1990*b*), it is difficult to identify the specific rules on which the design of a minimal manual is based. In this chapter an attempt is made to designate these design rules. First, a brief historical sketch is given, illustrating both the reason for developing the minimal manual and its precursor. Then the characteristic features of the minimal manual are described. This description comes in the form of an operational definition in that it presents an overview of the design principles that underlie the minimalist approach. In the discussion directions for research on minimalism are presented.

[4]actually, Carroll started working on the minimal manual in 1982, but the work was not cleared for discussion outside IBM until 1984 (Carroll, pers. comm.)

2.2 Origin of the minimal manual

The minimal manual was first introduced by Carroll (1984*a,b*). Its conception was prompted by the considerable difficulties people encountered when learning to use a computer program. To understand these problems, and to find ways to overcome them, several observation studies were conducted (e.g., Carroll, 1982; Carroll & Mack, 1984; Rosson, 1984). These observations and the early attempts to solve the users' problems gave rise to the development of the minimal manual.

2.2.1 RAISON D'ETRE

Many people experience the process of becoming proficient with a new computer program as a difficult and sometimes even daunting endeavor. To a great extent, their training problems depend on the idiosyncratic details of the software. That is, first-time users often have difficulties understanding the program's prompts, its interface, and its commands. Considering these problems in a wider context, they can be classified as problems with software and problems with manuals.

Some software problems are caused by 'inconsistencies' in the program. In early word processors, for example, the function of the ESC key often depended on the program state. It served as an undo button in the command mode but as a shortcut to quitting the program in the text mode. Software problems may also result from a poor interface. In multi-layered programs like data-base or spread-sheet applications, new users often have trouble keeping track of the relevant contexts. In addition, many of the program's command names are potentially confusing and make little sense to novice users. Problems also arise when the program gives mysterious and ineffective error messages. For example, to first-time users prompts like "general EXEC failure" merely indicate that something has gone wrong. The expression EXEC does not explain what the user did wrong, and the message contains no information on how to correct the error.

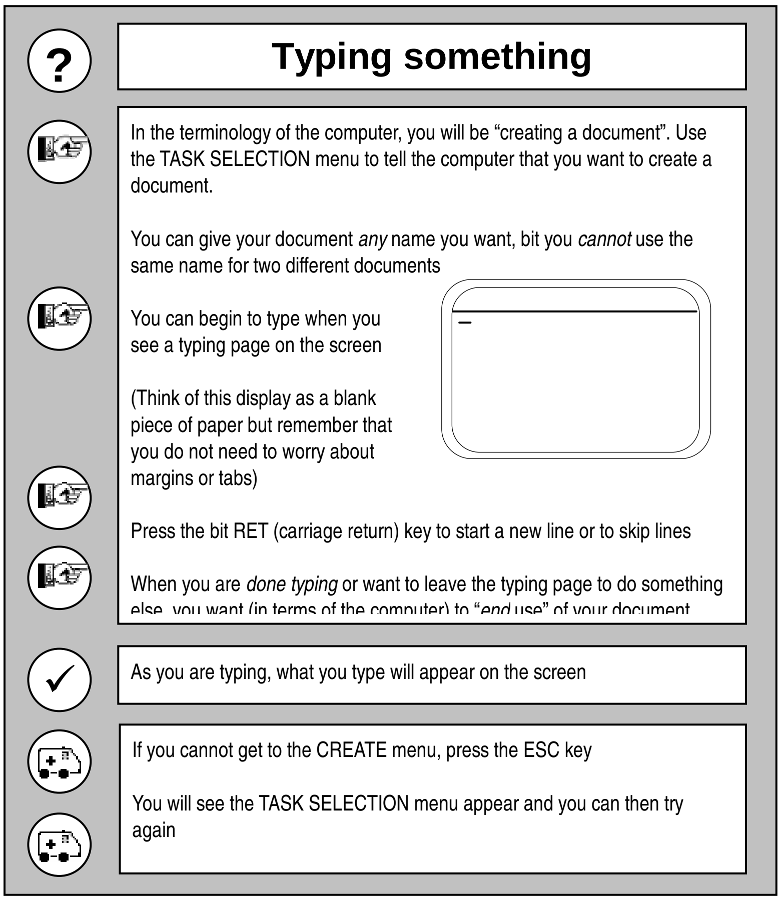
Most of these problems can, at least in theory, be overcome by good documentation. In fact, manuals that addressed these problems were state-of-the-art in the early 1980s. These manuals gave exhaustive descriptions of the different functions of keys before they were to be used. They explicitly defined the structure and operation of the program on the basis of various screendumps and detailed specifications of the actions to be performed. Some manuals even contained a separate trouble-shooting section that listed possible error-messages.

However, these manuals did not eliminate the new users' problems with the software. Rather, it turned out that new users often had problems with the manual itself. According to Carroll (Carroll, 1982, 1984*a,b*; Carroll & Mack, 1984; see also Mack, Lewis & Carroll, 1983), these problems mainly arose because the manual did not address the learning styles of new users. His observations showed that in their very first interaction with a computer program, new users' actions can be classified into three classes of learning strategies: learning by doing, by thinking and by knowing.

New users *learn by doing*. They tend to be active learners that want to do things with the program rather than read through endless pages of what they consider 'just information'. This desire to act comes from the fact that they often enter the training scene with a clear goal on which they want to start working right away. They want the manual to tell them what to do – what keys to press – not why to do it. However, the then-existing tutorials did not allow users to pursue their personal goals. These manuals expected users to study lengthy chapters containing program specifications, screendumps, and other descriptive information. The users' interaction with the program merely consisted of following programmed exercises, something new users considered to be passive rather than active.

New users are also active in that they try to make sense of what happens while they interact with the program. That is, they *learn by thinking*. They tend to reflect upon why the program operates as it does and try to interpret what appears on the screen. Because new users lack basic knowledge about how the software operates, their ideas are often incomplete or incorrect. These misconceptions are a major source of errors which most then-existing tutorials were unfit to remove. New users tended to skip its long-winded explanations, and, when they did read them, often had trouble extracting the right information. In addition, these manuals lacked information to correct the errors that might result from these misconceptions.

New users also build on their prior knowledge in developing new understanding. They *learn by knowing*. Because adult novices usually have considerable knowledge of the underlying task domain, they have clear expectations about how certain tasks are performed on the computer. What they already know may, however, conflict with what they are trying to learn. For example, a word processor resembles a typewriter only in some respects. There are many similarities but also many differences (cf. Allwood & Eliasson, 1987). Typical errors such as trying to move the cursor by pressing the space bar may arise because of this typewriter metaphor. Again, most standard self-instruction manuals offered little support in removing these misconceptions and correcting the errors that result from them.



Typing something

In the terminology of the computer, you will be “creating a document”. Use the TASK SELECTION menu to tell the computer that you want to create a document.

You can give your document *any* name you want, bit you *cannot* use the same name for two different documents

You can begin to type when you see a typing page on the screen

(Think of this display as a blank piece of paper but remember that you do not need to worry about margins or tabs)

Press the bit RET (carriage return) key to start a new line or to skip lines

When you are *done typing* or want to leave the typing page to do something else, you want (in terms of the computer) to “*end* use” of your document

As you are typing, what you type will appear on the screen

If you cannot get to the CREATE menu, press the ESC key

You will see the TASK SELECTION menu appear and you can then try again

Figure 2.1
Example of a guided-exploration card (Adapted from "The Nürnberg Funnel: Designing minimalist instruction for practical computer skill" (p. 112) by J.M. Carroll, 1992, Cambridge: MIT Press).

2.2.2 HISTORICAL PREDECESSOR

The observations discussed above prompted Carroll and his colleagues to design instruction that would not only solve new users' software problems but also their problems with manuals. They developed guided-exploration (GE) cards for learning to use a word processor (Carroll, 1990*b*; Carroll, Mack, Lewis, Grischkowski & Robertson, 1985). This deck of cards summarized the software's basic functions and they were, of course, intended to comply with the learning strategies of new users. An example is shown in Figure 2.1.

To allow users to learn by doing, each card concentrated on a learning goal first-time users really want to achieve. For example, 'underlining something' or 'printing a document on paper'. Moreover, the information on the cards was intentionally left incomplete. Only essential explanations were given and there was no step-by-step specification of procedures. Instead, only the critical elements of a procedure were defined. This, in turn, impelled the users to infer the missing information by reasoning from their own knowledge and understanding (i.e., learning by thinking).

Learning by thinking was further supported by checkpoints and remedies. As users were to fill in procedural details by inferencing and by reasoning, they were assumed to err frequently. Therefore, information to detect and correct errors was included. Checkpoints were used to indicate whether the user was still on the right track, and each card explicitly specified procedures to correct salient errors. Both means were intended to give users a feeling of safety and to motivate them to further explore the program.

The GE-cards also supported learning by knowing in that they were task-oriented. Each card contained a goal statement that addressed a topic new users could easily understand from prior knowledge of the underlying task domain (i.e., routine office typing procedures). Moreover, typewriter knowledge was exploited and confusions stemming from this analogy were addressed (i.e., negative transfer, see chapter 1).

In addition, because new users are willing to achieve personal goals, the GE-cards used a modular approach. Each card covered a discrete procedure without reference to material covered on other cards. The GE-cards thus became a set of independent, unordered cards that could be used in any sequence. The presentation of the information on the cards was modular as well. Each card contained a goal statement, hints, checkpoints, and remedies. These information types were graphically delineated and icons were used to identify them (see Figure 2.1).

The functionality of the GE-cards was established by Carroll *et al.* (1985). They studied 12 novice computer users who learned to use a word processor. All subjects had experience in typing letters in the office environment but had no experience in word processing. Half of the group used the GE-cards, the other half received a commercially developed self-study manual. Both instructional methods covered the same topics, differing only with regard to the presentation form: the 25 GE-cards represented the content of 94 pages of the manual.

The results indicated that GE-users required 51% less practice time and that they were significantly faster and better at performing tasks after practice. GE-users also raised significantly fewer questions about the purpose of an activity and they were more likely to engage in exploring activities not described on the cards. Moreover, they detected more errors and more often corrected them independently.

When asked how the materials could be improved, the GE-users requested more explanatory material and better graphical separation of the different sections of the cards. They also voiced a desire for a more structured training tool; in particular, they asked for a manual (Carroll *et al.*, 1985, 1987). Based on these suggestions, Carroll decided to develop a self-study manual that capitalized on the strengths of the GE-materials and also fulfilled the desire of learners to have a structured manual. This led to the minimal manual.

2.3 Characteristics of a minimal manual: minimalist principles

Carroll's published work leaves it somewhat obscure what exactly must be understood by a minimal manual. Since its conception, a gradually changing set of features has been attributed to the minimal manual. However, in general, a minimal manual can be defined by four so-called minimalist principles: (1) task orientation, (2) text optimization (3) support of error-recovery, and (4) modularity (Carroll, 1990*a*,*b*; Carroll *et al.*, 1987).

Due to the general nature of these principles, practitioners have repeatedly called for more detailed guidelines and worked examples on their application (Hallgren, 1992; Horn, 1992; Nickerson, 1991; Tripp, 1990). To comply with this request, Carroll's original minimalist principles were specified into more detailed design principles (see Table 2.1). They are described in the following sections and illustrated with examples from both the original minimal manual[5] and the one used in the experiments described in this thesis (henceforth referred to as the WordPerfect manual; see Appendix 2).

2.3.1 TASK ORIENTATION

Minimalist instruction focusses on task execution, on functionality for the user. Its primary goal is to help the novice user accomplish basic tasks. In a way, a task-oriented manual resembles a cookbook: it provides recipes for all the things a user might want to do with the software, showing how to use each command in the context of the recipe. The five design principles that

[5]the original minimal manual was published as an appendix to Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., & Mazur, S.A. (1986). *The minimal manual* (IBM Research Report No. 11637). Yorktown Heights: IBM.

Table 2.1
Minimalist principles and their corresponding design principles

1.	**Task orientation**	
	1.1	Focus on the program's basic functions
	1.2	Treat general methods before specific ones
	1.3	Give the opportunity to act early on
	1.4	Encourage exploration and problem solving
	1.5	Focus on real and familiar tasks
2.	**Text optimization**	
	2.1	Do not spell out everything
	2.2	Replace unnecessary jargon by familiar terms
	2.3	Write in short, simple sentences
	2.4	Use an active tone of voice
3.	**Support of error-recovery**	
	3.1	Give linkage-information to teach monitoring skills
	3.2	Use a standard formula for error-information
	3.3	Give 'on the spot' error-information
	3.4	Treat general recovery methods before specific ones
4.	**Modularity**	
	4.1	Provide closure for chapters
	4.2	Make chapters short
	4.3	Present different information types differently

contribute to the task-oriented nature of the minimal manual are described below.

A minimal manual is primarily intended for first-time users. It therefore deals with the basic functions of a program, using a simple-to-complex sequence. The minimal manual for WordPerfect first addresses the elementary tasks in the word processing cycle: typing, saving, revising, and printing a text (see chapter 1). In the course of practice, these tasks (and their related skills) are used as a prerequisite for more advanced tasks such as centering text, changing the typeface, or adjusting margins.

In explaining a programs' elementary functions, a minimal manual treats general methods before specific ones. Many of today's application programs offer more than one method to execute a command. For example, in WordPerfect there are at least three ways to retrieve a document: (1) by selecting the RETRIEVE command from the FILE menu, (2) by pressing the F5 key twice, or (3) by typing the filename directly from the DOS-prompt. The

WordPerfect manual uses the menu approach because it closely resembles the methods that are used to execute other commands.

According to their learning preferences (see chapter 1), new users typically have a strong desire to act. The minimal manual supports this desire by giving users the opportunity to act early on. It emphasizes the procedural part of a program, leaving out all information that does not directly relate to 'doing things'. Minimal manuals therefore have no preface, long-winded introduction, or general description of how the program works. Rather, they give users the opportunity to act early on. In the WordPerfect manual users receive their first instructions to act on page 2. In contrast, a brief inventory of commercially available tutorials for WordPerfect shows that the first instruction normally appears around page 15 (Van der Meij & Carroll, in press).

The minimal manual also supports learning by thinking. It invites and even stimulates users to explore new aspects of the program. Following from the GE-training materials, this principle has frequently been interpreted as a plea for leaving out basic action information. This is a mistake, however. All the necessary action steps for performing a task are described in a minimal manual. Instead, exploration is encouraged in open-ended exercises and in so-called 'do it yourself' sections. In both instances, the minimal manual explicitly relates explorations to a larger framework of goals and methods. In a way, it guides the users exploratory behavior, thus providing an exploratory environment in which it is safe to experiment with the program and try things out on your own (see Appendix 2).

As the fifth design principle indicates, minimalist instruction is always anchored, or situated, in the underlying task domain. That is, minimalist training tasks are real and familiar to the target population. The users' interest in and understanding of these tasks is what incited them to learn to use the program in the first place. By designing instruction around these tasks, new users' learning preferences are met and, consequently, their motivation is sustained. In Carroll's minimal manual tasks addressed genuine clerical activities. Trainees were secretaries who, like in their daily work, were asked to type short memos, revise letters, and create press reports. In the WordPerfect manual users (i.e., students) had to perform tasks like typing a short invitation, revising the minutes of a member's meeting, and changing the lay-out of a complaint to a telephone company. In contrast, most commercially developed word processing tutorials expect users to type a full-page sample text on fictitious topics like wine-producing countries (Mincberg, 1988), an overview of bookkeeping (Mincberg, 1987), or an incorrect delivery of 10,000 video tapes (Boom, 1990).

2.3.2 TEXT OPTIMIZATION

The second minimalist principle is basically a mixture of design principles that accounts for the minimal size of the manual. This principle was originally referred to as 'slashing the verbiage', but, as Table 2.1 shows, there is more to text optimization than just weeding out the excess. Rather, the text is adapted to or even 'written around' the users' actions.

The first design principle is perhaps the only true instance of 'slashing the verbiage'. It is realized by eliminating or radically cutting down all material not related to 'doing things'. Thus the minimal manual lacks sections like the welcome word, introduction, trouble-shooting section, index, and glossary. Within each chapter repetitions, summaries, reviews, advance organizers, and the like are also almost entirely absent. Moreover, definitions are operational instead of conceptual. That is, they are presented in context, immediately before or after the relevant action steps and describe what something 'does' rather than what it 'is'. As a result, the chapters in a minimal manual have an average length of three pages.

The minimalist call for intentionally incomplete information also means that information that can easily be inferred is left out. Clearly, it is difficult to judge when such situations apply. As a rule of thumb, incomplete information is given when the program provides sufficient support to fill in the missing directions. A minimal manual exploits the program's prompts by directing the users' attention to the screen. For example, the minimal manual contains messages like "Look at the screen. WordPerfect tells you what to do to actually remove these lines". On the screen, the prompt **Delete Block? N**o (**Y**es) appears. Because the manual does not tell users what to do here, they are forced to do some thinking (inferencing) on their own.

The second design principle on text optimization refers to the control of terminology. Because new users have no computer knowledge, computer terms are incomprehensible to them. In a minimal manual all but the necessary jargon and technical expressions are therefore substituted by more common terms. These terms are drawn from the task domain and the users' prior knowledge. In word processing, for example, terms like blank line and document are used instead of carriage return and file. The keys are indicated by their full keyname rather than by some code (e.g., the ENTER key instead of enter, [enter] or [↵]). In addition, the headings clearly signal the goals users may want to achieve (e.g., "Changing the margins") instead of the actions the program can perform (e.g., "Strategies for indenting and aligning text").

There are two exceptions to this rule. Firstly, the wording of the manual is at all times in line with the terms that are used in the program's menu choices and in the system messages. This might lead to inconsistencies (e.g., in WordPerfect the terms document and file are used interchangeably) which, in turn, support the notion that a manual can sometimes also be used as symptom of poor interface design. Secondly, terms that belong to the basic computer jargon (e.g., printer, diskette, cursor) are not replaced because new users either know or should get to know these terms.

The wording of a manual may never stand in the way of the understanding of its content. That is, reading and understanding the text may never tax the users' processing capacities. For that reason, sentences in a minimal manual are short, about 14 words. The minimal manual thus aims at a reading level of 12-year-olds (Flesch, 1964). Sentences use a simple subject-predicate order and embedded sentences are minimized. On the word level, simple, easy-to-understand words are used. Long words and slang are avoided as much as possible.

A related design principle is the use of an active tone of voice. In a way, the style of writing should reflect the users' active orientation toward learning. By using an active tone of voice, the length and complexity of a sentence is almost automatically reduced. For example, action steps like " You must now press the ENTER key to save the text you have just typed" can be rewritten as "Press the ENTER key to save your text" without losing essential information.

2.3.3 SUPPORT OF ERROR-RECOVERY[6]

Carroll and his colleagues have been among the first to recognize the importance of errors in learning to use software. First-time users make many mistakes, and correcting these mistakes can be very time-consuming (Arnold & Roe, 1987; Graesser & Murray, 1990; Lazonder & Van der Meij, 1994). Minimalist instruction therefore not only teaches users how to do things, but also how to undo the things that have gone wrong. The minimalist design principles that allow for the development of these so-called corrective skills are summarized below.

New users often find it difficult to co-ordinate the processing of manual, screen, and keyboard (Van der Meij, 1994). When users ignore the screen, it is unlikely they will discover mistakes. Consequently, errors will pile up, and it will become increasingly difficult to correct them. To overcome this so-called 'nose-in-the-book syndrome' (Carroll, 1984*b*; Jansen & Steehouder, 1989) a minimal manual contains linkage-information. Linkage-information prompts users to look at the screen and locate information like system cues and program messages. In Carroll's minimal manual a typical example of linkage-

[6] this minimalist principle is central to chapter 4 to 6. For that reason, it is discussed briefly here.

information reads: "Can you find the letters Repl on the 'status line' at the very top of the screen?".

Apart from this regulation process, minimalist instruction also supports the error-recovery process itself. This process consists of three stages: detection, diagnosis and correction. In a minimal manual, error-information supports these stages in a fixed order. Detection comes first, then diagnosis, and then correction (Lazonder & Van der Meij, 1994). As users who have not made a mistake may also read the error-information, it is important to present the detection information as a proviso. In combination with what users might do next, this leads to the standard "If ... (detection) then ... (cause) then ... (correction)" formula. Note that the detection part of the error-information again directs the users' attention to the screen to check if that particular error has occurred.

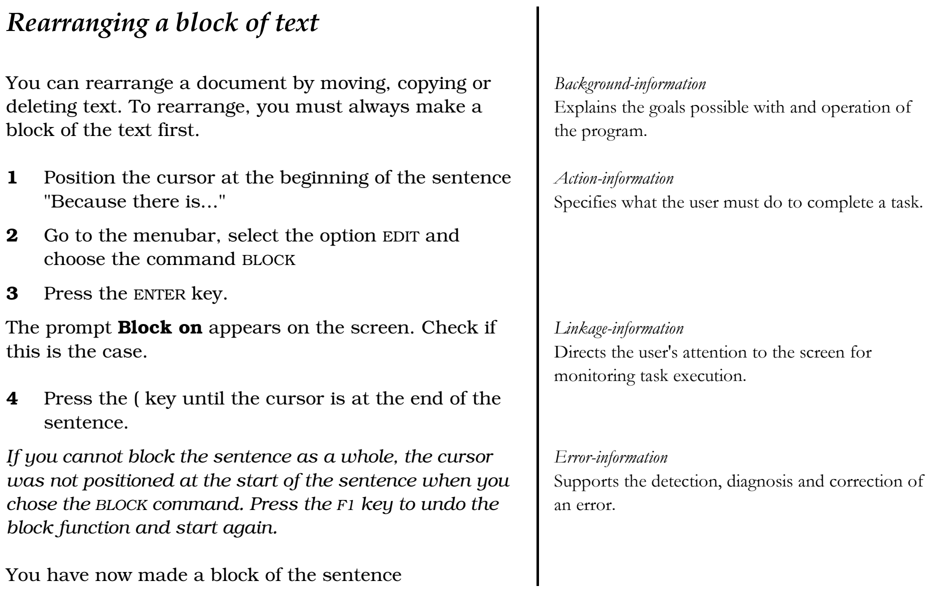
Early detection of an error is especially important for its correction. In a minimal manual error-information is therefore presented where users need it most. That is, not in a separate trouble-shooting section but directly after the actions it refers to. As a rule of thumb, error-information is given when a set of actions causes (or should cause) a distinct outcome. For example, it appears in the manual when actions lead to a program message, a print preview, or a menu on the screen.

Most programs nowadays have at least two ways to correct an error: a general and a specific method. A typical example of a general correction method in the original minimal manual is pressing the CODE + CANCEL key. Because general methods do not work for all errors, specific strategies like "Press the F7 key and type an N twice" are sometimes necessary. In line with design principle 1.1, a minimal manual always treats general correction methods before specific ones because general methods can be used over and over again.

2.3.4 MODULARITY

Probably because of its generic nature, modularity is open to many different interpretations. In minimalist instruction, modularity basically means that it must be possible to study a chapter without any knowledge of the other chapters. But, as Carroll's study on GE-cards showed, users are not particularly fond of a strictly modular approach (Carroll *et al.*, 1985). Therefore, in the minimal manual modularity is somewhat subsided into the following design principles.

One of the main reasons for using a modular approach is to accommodate browsing (Arnold, 1988). New users do not read their manual cover to cover



Rearranging a block of text	
You can rearrange a document by moving, copying or deleting text. To rearrange, you must always make a block of the text first.	*Background-information* Explains the goals possible with and operation of the program.
1 Position the cursor at the beginning of the sentence "Because there is..." **2** Go to the menubar, select the option EDIT and choose the command BLOCK **3** Press the ENTER key.	*Action-information* Specifies what the user must do to complete a task.
The prompt **Block on** appears on the screen. Check if this is the case.	*Linkage-information* Directs the user's attention to the screen for monitoring task execution.
4 Press the (key until the cursor is at the end of the sentence.	
If you cannot block the sentence as a whole, the cursor was not positioned at the start of the sentence when you chose the BLOCK command. Press the F1 key to undo the block function and start again.	*Error-information* Supports the detection, diagnosis and correction of an error.
You have now made a block of the sentence	

Figure 2.2
Illustration of the information-types in a minimal manual. The right-hand column describes the various information-types. The left-hand column illustrates their sequencing and presentation in the manual.

but skip and skim through it in order to attain personal goals (e.g., Penrose & Seiford, 1988; Rettig, 1991; Scharer, 1983). To meet with this strategy, the chapters in a minimal manual are self-contained. Each chapter starts afresh with retrieving or creating a document and ends with saving the revised document. Explicit cross-references between chapters are not included. In addition, all chapters start from the basic text of a document; possible changes to a document are never elaborated on in subsequent chapters. Users who wish to attend to parts of the manual at will can do so.

There is, however, no complete modularity in a minimal manual. The first chapter usually deals with starting the program and is therefore prerequisite to the rest. But, by and large, the chapters in a minimal manual can be studied independently. This is not to say that they must be studied randomly. Following the users' suggestions to improve the GE-cards, the minimal manual is flexible for study. Users are given the opportunity to study the program in random order. However, when they do follow the sequencing of the manual, their training progresses in a simple-to-complex manner (see design principle 1.1).

Chapters in a minimal manual are short, varying from 2 to 4 pages. However, even such relatively short chapters may be very demanding for first-time users. In designing minimal manuals, a hard and fast rule is that 95% of all users should be able to work through a chapter within 30 minutes (Van der Meij & Carroll, in press). This proved to be the right time for users to keep concentrated and motivated. After that, users can either start working on another chapter or stop training. In the latter case, the short chapters provide many convenient points to do so.

Within each chapter modularity means that different information types are presented differently. A minimal manual usually contains four types of information: background-information, action-information, linkage-information and error-information (see Figure 2.2; see also Appendix 2). Various techniques can be used to differentiate them. For example, in the WordPerfect manual the action-information is signalled by a number. Error-information is printed in italics. Background-information and linkage-information are put in roman. However, the prompts in the linkage information are always designed to resemble the programs' prompt as much as possible (e.g., "Check if the text **Save Document Y**es (**N**o) appears on the screen").

2.4 Conclusion

Carroll and his colleagues have laid a good foundation for the development of better tutorial documentation. Their minimalist approach blazed a new trail by taking the users' information needs and learning preferences instead of the program's functions as a starting point for design. This learner oriented approach is reflected in a set of minimalist principles whose practical application has already triggered interesting dialogues (e.g., Williams & Farkas, 1992) and significantly advanced the understanding of how minimalism can be operationalized.

Carroll's directions on manual design are not unique for minimalism; some of the minimalist (design) principles can also be found in other tutorials. Still, only a limited number of these principles have found their way into practice or handbooks on manual design. This may be due to the fact that there has been some controversy over the application of the minimalist principles. To settle this dispute, this chapter presented a brief, operational definition of the minimalist approach. Recently, Carroll himself has presented a set of heuristics that go more deeply into the way the minimalist principles should be put into practice (Van der Meij & Carroll, in press).

Empirical studies show that the minimal manual works well (e.g., Black, Carroll & McGuigan, 1987; Carroll *et al.*, 1987; Gong & Elkerton, 1990;

Vanderlinden, Cocklin & McKita, 1988). The use of a minimal manual reduced the learning time with approximately 40% in all these studies. It also cut down the time to complete performance tests and it significantly increased the retention of skills.

However, whether a minimal manual will yield similar results in other than 'standard' circumstances is yet unknown. Most studies have focused on American novices learning to use a word processor. Whether a minimal manual is equally effective for different audiences is not yet clear. For example, experienced users might be unsettled by routines they have adopted from other programs. Consequently, their information needs might best be met by a manual that impedes this negative transfer by comparing the new program with other software packages and computer systems. A related question is whether the minimalists' emphasis on learning by doing fits users from other cultures. In fact, it has been argued that users with a conceptual, function-oriented orientation toward learning are unlikely to benefit from the minimalist approach (Aizu & Amemiya, 1985; Warren, 1994).

In addition, little is known about the functionality of minimalism for different programs. Until the 1990s, most experiments were performed with line-editors which were so user-unfriendly that the tutorial, in a way, had to make up for the weaknesses of the program. Since the art of designing user-friendly software has improved during recent years, minimalism might not yield the same benefits in learning to use a modern full-screen word processor. In similar fashion, one might question its suitability in domains other than word processing. For example, is minimalism also applicable in highly complex or dangerous task domains like welding, radiotherapy, or cranedriving?

From a practical viewpoint, little evidence exists that the effect of a minimal manual can indeed be generalized to other audiences and software packages. When this research project started in september 1990, European research on minimalism was scant (cf. Carroll, 1994). Although some studies examined features of user documentation that can also be found in a minimal manual (e.g., Frese *et al.*, 1988; Oatley, Meldrum & Draper, 1991; Wendel & Frese, 1987), the effectiveness of the minimalist approach as a whole had not yet been established. This notion provides a challenge for research on minimalist documentation. It also served as a starting point for the present inquiries into the functionality of the minimalist approach.

References

Aizu, I., & Amemiya, H. (1985). The cultural implications of manual writing and design. *Proceedings of the 32nd. International Technical Communications Conference* (pp. WE 33 - 35). Washington: STC.

Alwood, C.M., & Eliasson, M. (1987). Analogy and other sources of difficulty in novices' very first text-editing. *International Journal of Man-Machine Studies, 27*, 1 - 22.

Arnold, B., & Roe, R. (1987). User-errors in human-computer interaction. In M. Frese, E. Ulich & W. Dzida (Eds.), *Psychological issues of human computer interaction in the workplace* (pp. 203 - 220). Amsterdam: Elsevier.

Arnold, W.A. (1988). Learning modules in minimalist documents. *Proceedings of the 35th. International Technical Communications Conference* (pp. WE 16 - 19). Washington: STC.

Black, J.B., Bechtold, J.S., Mitrani, M., & Carroll, J.M. (1989). On-line tutorials: What kind of inference leads to the most effective learning?. *CHI'89 Conference Proceedings* (pp. 81 - 83). Austin: ACM.

Boom, M. (1990). *Learning Word for windows now*. Redmond: Microsoft press.

Brockmann, R.J. (1986). *Writing better computer user documentation: From paper to hypertext* (1st. edition). New York: Wiley.

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

Carroll, J.M. (1984*a*). Minimalist design for active users. In B. Shackle (Ed.), *Interact'84: First IFIP Conference on Human Computer Interaction* (pp. 621 - 626). Amsterdam: Elsevier.

Carroll, J.M. (1984*b*). Minimalist training. *Datamation, 30*, 125 - 136.

Carroll, J.M. (1990*a*) An overview of minimalist instruction. *Proceedings of the twenty-third annual Hawaii international conference on systems sciences* (pp. 210 - 219). Washington: IEEE Computer Society.

Carroll, J.M. (1990*b*) *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill*. Cambridge: MIT Press.

Carroll, J.M. (1994). Techniques for minimalist documentation and user interface design. In M. Steehouder, C. Jansen, P. van der Poort & R. Verheijen (Eds.), *Quality of technical documentation* (pp. 67 - 76). Amsterdam: Rodopi.

Carroll, J.M., & Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors, 26*, 377 - 389.

Carroll, J.M., & Mack, R.L. (1984). Learning to use a word processor: By doing, by thinking and by knowing. In J.C. Thomas & M.L. Schneider (Eds.), *Human factors in computer systems* (pp. 13 - 51). Norwood: Ablex.

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

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., & Mazur, S.A. (1986). *The minimal manual* (IBM Research Report No. 11637). Yorktown Heights: IBM.

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., & Mazur-Rimetz, S.A. (1987). The minimal manual. *Human Computer Interaction, 3*, 123 - 153.

Crandall, J.A. (1987). *How to write tutorial documentation*. Englewood Cliffs: Prentice Hall.

Flesh, R. (1964). *The ABC of style: A guide to plain English*. New York: Harper & Row.

Foehr, T., & Cross, T.B. (1986). *The soft side of software: A management approach to computer documentation*. New York: Wiley.

Frese, M., Albrecht, K., Altmann, A., Lang, J. Von Papstein, P., Peyerl, R., Prümper, J., Schulte-Göcking, H., Wankmüller, I. and Wendel, R. (1988). The effect of an active development of the mental model in the training process: Experimental results in a word processing system. *Behaviour and Information Technology, 7*, 295 - 304.

Gong, R., & Elkerton, J. (1990). Designing minimal documentation using a GOMS model: A usability evaluation of an engineering approach. *CHI'90 Proceedings* (pp. 99 - 106). New York: ACM.

Graesser, A.C., & Murray, K. (1990). A question-answering methodology for exploring a user's acquisition and knowledge of a computer environment. In S.P. Robertson, W. Zachary & J.B. Black (Eds.), *Cognition, computing and cooperation* (pp. 237 - 267). Norwood: Ablex.

Grimm, S.J. (1987). *How to write computer documentation for users* (2nd. edition). New York: Van Norstrand Reinhold.

Hallgren, C. (1992). The Nürnberg Funnel: A minimal collection. *The Journal of Computer Documentation, 16*, 11-17.

Horn, R.E. (1992). Commentary on the Nürnberg Funnel. *The Journal of Computer Documentation, 16*, 3-11.

Jansen, C.J.M., & Steehouder, M.F. (1989). *Taalverkeersproblemen tussen overheid en burger* [Linguistic interaction problems between government and civilian]. PhD thesis, Utrecht University, Utrecht, The Netherlands.

Lazonder, A.W., & Van der Meij, H. (1994). Towards a theory of effective error control in computer documentation. In F.P.C.M. de Jong & B.H.A.M. Van Hout-Wolters (Eds.), *Process-oriented instruction and learning from text* (pp. 165 - 174). Amsterdam: VU University Press.

Mack, R.L., Lewis, C.H., & Carroll, J.M. (1983). Learning to use a word processor: Problems and prospects. *ACM Transactions on Office Information Systems, 1*, 254 - 271.

Maynard, J. (1979). A user-driven approach to better user manuals. *Computer, 1*, 72 - 75.

Mincberg, M. (1987). *WordPerfect made easy*. Berkeley: McGraw-Hill.

Mincberg, M. (1988). *WordPerfect: Secrets, solutions and shortcuts*. Berkeley: McGraw-Hill.

Nickerson, R.S. (1991). A minimalist approach to 'the paradox of sensemaking'. *Educational Researcher, 20*, 24 - 26.

Oatley, K., Meldrum, M.C., & Draper, S.W. (1991). *Evaluating self-instruction by minimal manual and by video for a feature of a word-processing system.* Unpublished manuscript, University of Glasgow.

Penrose, J.M., & Seiford, L.M. (1988). Microcomputer users' preferences for software documentation: An analysis. *Journal of Technical Writing and Communication, 18*, 355 - 366.

Price, J. (1984). *How to write a computer manual: A handbook of software documentation*. Menlo Park: Benjamin Cummings.

Rettig, M. (1991). Nobody reads documentation. *Communications of the ACM, 34*, 19 - 24.

Rosson, M.B. (1984). Patterns of experience in text editing. *Proceedings of the CHI'83 Conference on Human Factors in Computing Systems* (pp. 171 - 175). New York: ACM.

Scharer, L.L. (1983). User training: Less is more. *Datamation, 29,* 175 - 182.

Sohr, D. (1983). Better software manuals, your companies' survival may depend on them. *Byte, 8*, 186 - 294.

Steehouder, M.F. (1989). *Gids voor het schrijven van computerhandleidingen* [Guide for writing computer documentation]. Enschede: Universtiteit Twente, Vakgroep Toegepaste Taalkunde.

Tripp, S.D. (1990). Book review of: The Nürnberg Funnel. *Educational Technology: Research and Development, 38*, 87 - 90.

Van der Meij, H. (1994). Catching the user in the act. In M. Steehouder, C. Jansen, P. van der Poort & R. Verheijen (Eds.), *Quality of technical documentation* (pp. 201 - 210). Amsterdam: Rodopi.

Van der Meij, H., & Carroll, J.M. (in press). Principles and heuristics for designing minimalist instruction. *Technical Communication.*

Vanderlinden, G., Cocklin, T.G., & McKita, M. (1988). Testing and developing minimalist tutorials; A case history. *Proceedings of the 35th International Technical Communications Conference* (pp. RET 196 - 199). Washington: STC.

Warren, T.L. (1994). Issues in internationalization of documentation: Quality control. In M. Steehouder, C. Jansen, P. van der Poort & R. Verheijen (Eds.), *Quality of technical documentation* (pp. 171 - 184). Amsterdam: Rodopi.

Wendel, R., & Frese, M. (1987). Developing exploratory strategies in training: The general approach and a specific example for manual use. In H.J. Bullinger, B. Schackel & K. Kornwachs (Eds.), *Proceedings of the second IFIP conference on human-computer interaction* (pp. 943 - 948). Amsterdam: Elsevier.

Williams, T.R., & Farkas, D.K. (1992). Minimalism reconsidered: Should we design documentation for exploratory learning? *SIGCHI Bulletin, 24,* 41 - 50.

CHAPTER 3

The minimal manual: Is less really more? [7]

Carroll, Smith-Kerker, Ford and Mazur-Rimetz (1987) have introduced the minimal manual as an alternative to traditional self-study manuals. While their research indicates strong gains, only a few attempts have been made to validate their findings. This study attempts to replicate and extend the original study of Carroll et al. Sixty-four first-year Dutch university students were randomly assigned to a minimal manual or a standard self-study manual for introducing the use of a word processor. During training, all students read the manual and worked training tasks on the computer. Learning outcomes were assessed with a performance test and a motivation questionnaire. The results closely resembled those of the original study: minimalist users learned faster and better. The students' computer experience affected performance as well. Experienced subjects performed better on retention and transfer items than subjects with little or no computer experience. Manual type did not interact with prior computer experience. The minimal manual is therefore considered an effective and efficient means for teaching people with divergent computer experience the basics of word processing. Expansions of the minimalist approach are proposed.

3.1 Introduction

Computers have gradually become ubiquitous over the past ten years. Initially, computer users were highly trained engineers, mathematicians or programmers. Now the majority of users are interested lay people with little or no computer knowledge. This new audience has different documentation needs, which companies began to take seriously after finding out that a good manual could clinch a sale (e.g., Foss, Smith-Kerker & Rosson, 1987; Jensen & Osguthorpe, 1985; O'Malley *et al.*, 1983; Paxton & Turner, 1984). One of the ways in which companies have been able to adapt and upgrade their documentation is through extensive analyses of the behavior of the novice user.

[7] Lazonder, A.W., & Van der Meij, H. (1993). The minimal manual: Is less really more? *International Journal of Man-Machine Studies, 39*, 729 – 752 (with minor modifications).

These studies have indicated that first-time computer users have relatively simple training needs. They want to get to know how to operate a computer program, and they want to get to know it fast (Carroll, 1990*b*, Scharer, 1983). They are not interested in detailed information on how the computer or software works. Rather, they want to 'read to learn to do' (Redish, 1988).

First-time users also frequently depart from the prescribed paths in their manuals. That is, they often explore the functions of the program on their own. Mistakes abound during these explorations and users expect the manual to help them with these problems (Cuff, 1980).

Traditional self-study manuals do not subscribe to these needs. This may explain why some research has indicated that only 14% of the users actually read the manual (Penrose & Seiford, 1988). Prompted by this misfit, Carroll and his associates have put forth a radically new approach to documentation (Carroll, 1990*a,b*; Carroll, Smith-Kerker, Ford & Mazur-Rimetz, 1987). Their observations of the behavior of first-time users led to the development of a minimal manual (MM) for teaching novices how to use a computer program.

Research on minimal manuals has only recently started to emerge (e.g., Frese *et al.*, 1988; Gong & Elkerton, 1990; Olfman & Bostrom, 1988; Raban, 1988; Ramsay & Oatley, 1992; Vanderlinden, Cocklin & McKita, 1988; Wendel & Frese, 1987). All of these studies suggest that a MM is better than a traditional self-study manual (SS). It is difficult to draw a clear conclusion from these studies, however. They often do not give an explicit account of the principles used to design the MM and the experimental design of some of these studies calls for a little caution.

The main objective of this study is to find additional evidence for the functionality of the minimalist approach to first-time user documentation. Before presenting the study, it is important to detail the various ways in which it has examined the claims of minimalism and how it seeks to broaden the original experiment of Carroll *et al.* (1987).

Firstly, an important critique of minimalism is that Carroll and his colleagues have not given enough examples and explicit design rules for creating a MM (Hallgren, 1992; Horn, 1992; Nickerson, 1991; Tripp, 1990). We found this criticism to be only partially just. Whereas the major points of departure have been well described in various papers and in Carroll's book The Nürnberg Funnel (1990*b*), we also felt a need for more detailed, design-oriented guidelines. Therefore, we studied Carroll's original MM in order to discover the various design rules that are subsumed under the four major principles. In this and other papers we have outlined these design rules (Lazonder & Van der Meij, 1992; Van der Meij, 1992; Van der Meij

& Lazonder, 1992, see also chapter 2).

Secondly, another point of criticism against Carroll has been the lack of a fair comparison between manuals. One critic has raised the question of whether the control manual is a good example of its kind (Nickerson, 1991). In the present study, both the MM and the control manual were created especially for the experiment in order to have a well-controlled manipulation. The two manuals shared the same basic content and menu-oriented approach to the software, but they obviously differed in their design principles. In addition, both manuals were pilot tested and revised on the basis of the test findings and comments of experts. In short, both manuals are probably good representatives of their approaches to documentation.

Thirdly, the study examines whether computer experience contributes to the effects of the MM. Until now, almost no other study on minimalism has examined this contribution. As more and more people develop computer knowledge and skill, it becomes more important to discover whether minimalism works for people with divergent computer experience (cf. Oatley, Meldrum & Draper, 1989).

Fourthly, the initial development of a MM was stimulated by the bad press for conventional manuals. Carroll *et al.* (1987) wanted to create a manual that would adapt as much as possible to the users' preferences and processing of paper documentation. As a result, the MM is supposed to be better suited to what users want from a manual. It should, therefore, lead to a more positive motivation than a conventional manual. This effect on motivation has not yet been studied.

Fifthly, it is interesting to find out whether minimalism also works in a context that differs in many ways from Carroll's original study. This study does so. We used different materials, a different software system, different kinds of subjects from a different country with a different language. Perhaps the most vital question of these is whether the procedural approach of minimalism, as opposed to a conceptual one, works for other than American cultures. There is, for example, some indication that the Japanese are unlikely to benefit from a MM (Aizu & Amemiya, 1985; Mackin, 1989; Stevensen, 1992). Firstly, the manual's direct action statements might contrast with their notion of politeness. Secondly, the procedural approach of the MM contrasts with their conceptual orientation towards learning. Likewise, some people have expressed doubts whether it fits the Europeans (Brockmann, personal communication). In short, it is important to find out whether the minimalist approach works for European subjects (cf. Ramsay & Oatley, 1992; Wendel & Frese, 1987).

The present study thus attempts to validate and expand the work of

Carroll *et al.* (1987) in several ways. The study examined the effect of manual type and computer experience on two classes of dependent measures: procedural skill and motivation. Procedural skill was operationally defined by the same measures (e.g., shorter learning time, better test-performances) used in the original study of Carroll *et al.* (1987). For motivation, the study measured attention, relevance, confidence, and satisfaction (Keller, 1983, 1987).

In line with Carroll *et al.*'s findings, the MM was predicted to yield better procedural skill than the SS. In addition, the MM-subjects were expected to end up with higher motivation regarding word processing than SS-subjects because the MM is designed to meet the users' learning styles, learning preferences and informational needs better. Effects of computer experience are studied in an exploratory fashion.

3.2 The minimalist approach to tutorial documentation

Over time, Carroll and others have accorded slightly different characteristics to the MM. But, in general, a MM is based on the following minimalist principles: (a) task orientation, (b) text optimization, (c) support of error-recovery, and (d) modularity (Carroll, 1990*b*; Hallgren, 1992; Horn, 1992; Lazonder & Van der Meij, 1992; Van der Meij, 1992).

The *task-oriented* nature of the MM means that the manual focuses on the basic functions of the program. Thus, the MM allows users to get started fast and hardly supports any secondary actions (e.g., installation, advanced tasks) or details concepts (e.g., what menus do). In our MM for a word processor, the chapters therefore deal with tasks that users are familiar with, such as typing an invitation, changing the lay-out of a letter to a telephone company and revising the minutes of a member's meeting. The task-oriented nature of the MM also transpires in the headings. Chapter headings denote overall goals users may want to attain (e.g., "Rearranging text", "Changing characters, words and lines") and section headings refer to subgoals (e.g., "Copying text", "Changing the fontsize"). As in the original MM, 'Do it yourself' sections were included to stimulate users to discover new goals that the program could satisfy.

There are two basic rules behind the principle of *text optimization*. Firstly, there should be as little text as possible. Like the original, our MM lacks a preface, advance organizers, an index, and summaries at the end of each chapter. There is also little conceptual information in the manual, nearly all information refers to 'doing things'. Not even all procedures are fully specified. For example, some information that users can find on the screen, or that they can easily infer, was left out intentionally (e.g., "Look at the screen. WordPerfect explains how you actually remove these lines."). This was done to force users into discovering parts of the program by studying the screen, inferencing and reflecting.

Secondly, the text should be simple and without jargon where possible. For this reason, the text was presented in short sentences of about 14 words, in a subject-predicate order. Embedded sentences, like this one, were not used. In addition, most of the jargon and technical expressions were substituted by more common terms. Some potentially confusing words were, however, not removed because they belong to the word processor's menu choices or system messages (e.g., intermittent use of the words 'document' and 'file'), or because they belong to the basic computer lingo (in English) that any user should get to know (e.g., 'printer', 'diskette' or 'cursor').

Learning how to use a complex program such as a word processor inevitably causes users to make mistakes. Our MM therefore contains ample information to *recover errors*. General exits out of the program and general recovery strategies in the program are introduced early. Namely, in the first chapter and they reappear later on where appropriate. In subsequent chapters specific recovery information is presented (e.g., "If you have made the wrong choice, press the F1 key again to return to the menu."). To prevent users from making mistakes in the first place, the manual frequently directs their attention to the screen to check whether they are still on the right track. Illustrations were used to clarify operations on the hardware (e.g., turning the power on, inserting a diskette) and to help users identify special keys (e.g., F1, BACKSPACE) the first time they were to be used.

There is not a complete *modularity* of all of the chapters in our MM. The first chapter deals with starting and ending the program and is therefore basic to the rest. The remaining chapters can be worked through independently from one another, however, and there is no cross-referencing. Each chapter is thus self-contained, which is exemplified in the numbering of the pages. Each chapter starts afresh with page 1, and the page number is preceded by the chapter number (e.g., instead of page 18, users see page 3.2, meaning the second page of Chapter three).

The MM was developed for WordPerfect 5.1 and it copied Carroll's manual as much as possible (for a detailed description of this construction, see Van der Meij & Lazonder, 1992). The manual is not just a replica because of variations in hardware (IBM vs. Sirex), software (Display Writer vs. WordPerfect 5.1) and language (English vs. Dutch).

Every effort was made to make the MM different from the SS *only* with regard to the minimalist principles. Thus, both manuals covered exactly the same basic tasks (see Appendix 1), and the same command names and approach to the program (menu-oriented rather than by using the function keys of WordPerfect). Moreover, the lay-out was identical. The SS was adapted from a sample of currently used WordPerfect manuals. These manuals start from the belief that the presence of declarative information is a necessary condition for the development of a skill (e.g., Anderson, 1985). In addition to the procedural information that it shared with the MM, the SS therefore gave ample conceptual information. Thus, the SS contained regular sections such as a welcome word, an introduction, an index, and summaries. Moreover, explanations accompanied most of the procedures (i.e., the manual explains what happens 'inside the computer' when a command is executed). All procedures to attain a certain goal were explicated as opposed to the occasional inferencing in the MM. The SS also explained jargon and technical terms in detail, many of which were introduced before the users had turned the computer on. Like most traditional self-study manuals, the SS gave little error-information and there were no 'Do it yourself' sections that invited users to explore additional options of the program. As a result, the SS was almost twice as thick as the MM and it had three times as many words as the MM. Illustrative pages of both manuals are shown in Appendix 2 and 3.

3.3 Method

3.3.1 SUBJECTS

Sixty-four first-year Dutch university students participated in this study. There were 15 males and 49 females with a mean age of 19.1 (*SD*=2.2). The subjects received course credits for participation. They were classified as *novice*, *beginner* or *intermediate* user (Brockmann, 1990; Chin, 1986) and randomly assigned to one of the two experimental conditions (MM or SS). The allocation of subjects to conditions is shown in Table 3.1.

Subjects were considered novices when they had less than fifty hours experience with computers and no background with word processors. Beginners had either less than fifty hours experience with computers and experience with word processors, or had more than fifty hours computer experience, but no experience in working with WordPerfect. Intermediate users had some experience with WordPerfect.

The drop-out rate was low. Only one (SS) subject did not attend the second session. Due to a computer break-down, there were incomplete data for seven subjects during the learning phase (2 MM; 5 SS) and for five subjects during the test (2 MM; 3 SS). The data for these subjects were

Table 3.1
Number of subjects per condition

	Condition		
	MM[a]	SS[b]	Row total
User			
Novice	13	18	31
Beginner	7	7	14
Intermediate	10	9	19
Column total	30	34	64

[a] Minimal manual [b] Self-study manual

excluded on an analysis-by-analysis basis, causing variable group sizes in some of the analyses.

Experimental checks on the random allocation to conditions revealed no significant differences between the groups for intelligence, educational background, sex, initial motivation or typing skill. As Table 3.1 shows, computer experience was evenly distributed between experimental groups.

3.3.2 MATERIALS

Experimental setting and word processor

All sessions took place in a computer class equipped with a network of 19 Sirex 386-SX personal computers and a laser printer. The word processor (WordPerfect 5.1) was downloaded from a network, thereby assuring an identical setup for all subjects. Given the fact of relatively inexperienced users, WordPerfect's menu (instead of its infamous function keys) was used to execute commands (see Cuff, 1980).

During the hands-on part of the training sessions and during the test phase, a registration program stored the subjects' actions in a logfile. Every time a subject struck a key, time and keypress were recorded.

Instructional materials

During training, all subjects received a manual (MM or SS) and a diskette containing all documents to be used in training. Both manuals were developed iteratively. A concept version of each manual was reviewed by three experts (an instructional technologist, a technical writer with detailed knowledge of the software and a graphical designer). In addition, pilot tests were run with a number of novice students. On the basis of these findings, both manuals were revised.

Questionnaires and test

The participants filled in a background questionnaire, containing questions about sex, age, previous schooling, native language, typing skill and prior experience with computers. Intelligence was assessed with a standardized Dutch verbal analogies test (DAT'83; Evers & Lucassen, 1983). Motivation was assessed by two questionnaires. One questionnaire determined the subjects' initial motivation; another measured their motivation after training. Both questionnaires consisted of 45 behavioral descriptions (fillers included) that were drawn from existing instruments (Lemos, 1978; Popovich, Hyde, Zakrajsek & Blumer, 1987; Reece & Gable, 1982; Temple & Lips, 1989). The subjects judged each description (positively and negatively stated) on a 5 point agree-disagree scale. (See Appendix 4 for the relevant items and their reliability coefficients (Cronbach's alpha)).

A performance test was administered to assess learning outcomes. This test included basic-managerial, retention and transfer items. The six basic-managerial items addressed tasks like document retrieval and file saving. Since these tasks were practiced more frequently than the other tasks in the MM (or in the SS), they were analyzed as a distinct category. The nine retention items dealt with simple word processing tasks rehearsed during practice (e.g., copying and moving text, restyling words, paragraphs and pages). Four transfer items addressed topics that went beyond the scope of the manuals (e.g., changing the position of the page number, altering a footnote). Task documents were offered on a separate diskette.

3.3.3 PROCEDURE

The experiment was run in five groups of 7 to 16 subjects. All subjects attended two sessions of four hours each. The time between sessions was one week exactly for each group. In all, the experiment took two weeks. All procedures were identical for the various groups and the same two experimenters conducted all sessions. Subjects in the same session were given the same manual.

At the start of the first session, subjects received the paper and pencil tests, a manual and a diskette and were seated at their computer. After a brief introduction, they filled in the background and initial motivation questionnaire. Subsequently, they were given exactly twenty minutes to complete the verbal analogies test.

After the tests, more detailed instructions for working with WordPerfect were provided. As in the original experiment, subjects were instructed to work individually, in their own way and at their own pace. They were to consult the experimenter only when a mechanical error had occurred, or when they were stuck for more than 15 minutes. The students were asked not to use WordPerfect between sessions.

The second session started with another 2.5 hours of hands-on experience during which all participants managed to complete their training. After a short breaks, all subjects filled in the final motivation questionnaire. Next, they were given sixty minutes for the performance test. During the test the subjects were allowed to consult their manual.

Coding and scoring

The coding resembled that of the original study as much as possible. Dependent variables were: time to complete training, time to complete the performance test, number of errors in the test, recoveries from errors in the test, quality of test performance, and motivation. These measures were calculated for completed tasks only.

As in Carroll *et al.*'s (1987) original study, these data also led to the following performance measures: (a) performance success; (b) performance efficiency; (c) recovery effectiveness; and (d) recovery efficiency. Performance success was defined as the number of successfully completed items, which was assessed by examining the task documents stored on diskette and the logfiles of each subject. Performance efficiency was the ratio of the relative number of successfully completed items to the time to complete these items. Error corrections and recovery time were combined into a measure of recovery efficiency: the number of successful recoveries divided by the total recovery time for a given item. Effectiveness of recovery was defined as the number of correct revisions divided by the total number of revisions for a given item (for further details, see Carroll *et al.* 1987).

The initial motivation questionnaire measured five constructs: (a) curiosity; (b) relevance; (c) confidence; (d) reference group; and (e) persistence. The final motivation questionnaire measured (a) attention, (b) relevance, (c) confidence and (d) satisfaction (Keller, 1987). Both questionnaires employed a 5-point Likert-type scale with options bearing simple weights of 5, 4, 3, 2, or 1 for positive items, and the reverse for negative items. Scores on all items were added for each subject and compared between groups. High scores represent a high motivation; low scores indicate a low motivation.

Data analyses

The study was set up as a quasi-experimental design with manual type and computer experience as the two main factors. Manual has two levels (MM and SS). Experience has three levels (novice, beginner, intermediate), leading to a 2 x 3 design.

Table 3.2
Mean time to reach getting-started benchmarks

	Complete training	Start system	Start WP	Print
Manual				
MM[a]	144.5 (38.7)	2.7 (2.0)	1.3 (0.9)	29.9 (10.1)
SS[b]	195.0 (45.6)	9.2 (4.5)	2.1 (1.9)	82.7 (70.9)

Note. Time in minutes, Standard deviations in parentheses.
[a] Minimal manual [b] Self-study manual

All data were analysed by means of analyses of variance. Where appropriate, multivariate MANOVA analyses preceded univariate ANOVA and post-hoc Scheffé analyses (alpha was set at .05). All outcomes were corrected for initial motivation by inserting the five initial motivation scores into the analyses as covariates. As no interactions were found between manual type and computer experience, these data will not be reported.

3.4 Results

3.4.1 TIME

Table 3.2 presents the mean time (in minutes) subjects required to complete training. Overall, MM-subjects needed over 25% less time to learn to use the word processor. This difference was statistically significant ($F(1,46)=16.46$, $p<.01$).

Computer experience had no effect on overall learning time ($F(3,46)=1.85$). Novices, beginners and intermediate users all needed about the same time to complete training.

One of the design objectives of the MM is to allow users to get started fast. Table 3.2 lists the mean time it took subjects to reach some getting-started benchmark tasks. MM-users did indeed get to these tasks sooner than SS-users ($F(3,43)=20.96$, $p<.01$). As the table shows, they started the system about 6 minutes earlier ($F(1,45)=45.19$, $p<.01$) and were faster with printing their first document ($F(1,45)=12.45$, $p<.01$). There was no effect of computer experience on these benchmark tasks.

The mean time subjects required to complete the various test items is shown in Table 3.3. Manual again produced an effect. MM-subjects required significantly less time to complete basic-managerial items ($F(1,48)=8.46$, $p<.01$) and retention items ($F(1,48)=5.37$, $p<.05$). No effect of manual was found for transfer items ($F(1,38)=.43$).

Table 3.3
Mean solution time on test items

	Item type		
	Basic	Retention	Transfer
Manual			
MM[a]	0.5 (0.3)	3.0 (1.0)	7.7 (4.2)
SS[b]	1.1 (0.9)	4.0 (2.1)	6.5 (3.4)
Computer experience			
Novice	0.9 (0.8)	4.2 (2.0)	7.7 (4.8)
Beginner	0.6 (0.5)	2.8 (1.1)	6.3 (3.4)
Intermediate	0.7 (0.5)	2.7 (0.7)	7.1 (2.9)

Note. Time in minutes, Standard deviations in parentheses.
Since not all subjects completed the performance test (10 subjects did not get to the transfer items), the ratio of time to the number of complete items was compared between experimental groups. Given the fact of speed-test, 'faster' automatically implies 'having completed more items'.
[a] Minimal manual [b] Self-study manual

Computer experience affected the time to complete retention items (*F*(2,48)=4.78, *p*<.05). Post-hoc Scheffé analyses indicated that novice users needed significantly more time to complete retention items than beginners or intermediates.

3.4.2 ERRORS AND RECOVERIES

With errors, the ratio of errors to the number of items completed was compared for basic-managerial, retention, and transfer tasks. Descriptive statistics are presented in Table 3.4.

There was a significant multivariate effect of manual (*F*(3,39)=5.19, *p*<.01). Overall, MM-users made fewer errors, but a significant univariate effect was found only for basic-managerial items (*F*(1,41)=10.56, *p*<.01). Computer experience did not affect the number of errors (*F*(6,76)=1.77).

Because the MM aims to support the detection and correction of errors, MM-users were expected to correct more errors. Table 3.5 shows the percentage of corrected errors per item type, as a function of manual type and computer experience. As the mean scores indicate, there was no overall effect of manual. Whereas MM-subjects did make more successful recoveries on basic-managerial items, this difference was not significant (*F*(1,31)=2.97, *p*<.10) due to the extreme high variability of scores.

Table 3.4
Mean number of errors on test items

	Item type		
	Basic	Retention	Transfer
Manual			
MM[a]	0.13 (0.24)	0.55 (0.24)	5.82 (6.37)
SS[b]	0.32 (0.17)	0.55 (0.24)	6.65 (6.07)

Note. Standard deviations in parentheses.
[a] Minimal manual [b] Self-study manual

There was a significant effect of computer experience on recoveries on transfer items ($F(2,39)=4.65$, $p<.05$). Scheffé analyses revealed that more experienced users were significantly more successful in recovering from errors on transfer items than novices.

3.4.3 QUALITY OF PERFORMANCE

As in the original experiment, this study assessed the effect of the main factors on performance success, performance efficiency, recovery effectiveness and recovery efficiency.

Table 3.6 shows the mean performance success scores. Overall, MM-subjects successfully completed a significant 9% more items than did SS-

Table 3.5
Percentage of successful recoveries on test items

	Item type		
	Basic	Retention	Transfer
Manual			
MM[a]	80.5 (33.2)	45.3 (38.8)	18.8 (21.9)
SS[b]	50.5 (42.6)	37.2 (36.4)	15.7 (25.8)
Computer experience			
Novice	54.2 (39.5)	41.1 (46.5)	7.7 (8.4)
Beginner	75.0 (41.8)	42.6 (24.3)	29.3 (32.9)
Intermediate	59.1 (49.1)	39.4 (26.3)	21.5 (24.6)

Note. Standard deviations in parentheses.
[a] Minimal manual [b] Self-study manual

Table 3.6
Mean performance success scores

	Item type		
	Basic	Retention	Transfer
Manual			
MM[a]	5.7 (0.5)	5.3 (1.4)	1.3 (0.8)
SS[b]	4.6 (1.4)	5.1 (2.0)	1.0 (1.2)
Computer experience			
Novice	4.7 (1.4)	4.4 (1.6)	0.7 (0.7)
Beginner	5.6 (0.7)	5.9 (1.7)	2.0 (1.1)
Intermediate	5.3 (1.2)	5.9 (1.5)	1.4 (1.0)

Note. Standard deviations in parentheses.
Performance success = number of items successfully completed.
[a] Minimal manual [b] Self-study manual

subjects (*F*(3,48)=4.23, *p*<.05). Manual type had a significant univariate effect only on performance success on basic-managerial items (*F*(1,50)=11.55, *p*<.01). For retention and transfer items the difference between MM-subjects and SS-subjects was not significant.

Computer experience also significantly affected these scores (*F*(6,94)=3.94, *p*<.01). There were univariate effects on performance success on retention and transfer items (*F*(2,50)=5.58, *p*<.01 and *F*(2,50)=10.14, *p*<.01 respectively). Scheffé analyses showed that novice users were less successful on these items than beginners or intermediates (see Table 3.6).

Table 3.7 shows the performance efficiency data. Manual had a significant multivariate effect (*F*(3,36)=5.57, *p*<.01). In general, the performance of MM-users was more efficient than that of SS-users. Manual type had a significant univariate effect on performance efficiency only on basic-managerial items (*F*(1,38)=11.71, *p*<.01). Contrary to expectations, SS-subjects showed a higher performance efficiency score on retention and transfer items. This difference was not significant, however.

Computer experience also significantly affected performance efficiency (*F*(6,74)=3.28, *p*<.01). A univariate effect on retention items was found (*F*(2,38)=7.32, *p*<.01). Scheffé analyses again showed that novice users were less efficient than beginners or intermediates.

Table 3.8 presents the main findings for the effectiveness and efficiency of recovery. Recovery-effectiveness differed in favor of the MM-group on all three item-types, but a statistically significant outcome was found only

Table 3.7
Mean performance efficiency scores

	Item type		
	Basic	Retention	Transfer
Manual			
MM[a]	40.0 (16.6)	2.6 (0.9)	2.7 (1.9)
SS[b]	25.2 (16.0)	3.0 (1.5)	3.8 (4.6)
Computer experience			
Novice	30.6 (13.3)	2.1 (0.7)	2.4 (2.0)
Beginner	39.9 (22.8)	3.2 (1.6)	4.1 (4.2)
Intermediate	31.5 (18.3)	3.3 (0.9)	3.5 (4.2)

Note. Standard deviations in parentheses.
Performance efficiency = % of items successfully completed per time (min.) □ 100.
[a] Minimal manual [b] Self-study manual

for basic-managerial items ($t(35)=2.62$, $p<.05$). A similar finding was obtained for recovery-efficiency for these items ($F(1,30)=4.32$, $p<.05$).

Experience with computers had no effect on these measures.

Table 3.8
Efficiency and effectiveness of error-recovery

	Item type		
	Basic	Retention	Transfer
Efficiency			
Manual			
MM[a]	334.9 (380.9)	48.5 (50.7)	31.4 (47.6)
SS[b]	126.2 (169.3)	41.1 (44.4)	72.4 (164.7)
Effectiveness			
Manual			
MM[a]	100.0 (0.0)	96.2 (10.4)	93.7 (22.7)
SS[b]	80.5 (38.2)	85.0 (25.9)	85.0 (35.1)

Note. Standard deviations in parentheses.
Recovery efficiency = number of recoveries per time (min.) □ 100; recovery effectiveness = number of successful recoveries to the total number of attempted recoveries □ 100.
[a] Minimal manual [b] Self-study manual

3.4.4 MOTIVATION

Two of the subjects initial motivation scores affected the outcomes. Curiosity had a significant effect on the time to print a document ($t(45)=-2.29$, $p<.05$). As one might expect, time and curiosity were negatively related. Curious subjects printed their document earlier. In addition, persistence significantly affected the time subjects needed to complete basic-managerial items ($F(1,48)=4.43$, $p<.05$). More persistent subjects were faster in completing these items. Persistence also significantly affected the outcomes ($F(1,39)=4.28$, $p<.05$). High persistent subjects made more successful recoveries from errors on transfer items.

The MM is designed to meet users' learning preferences as much as possible. Therefore, it was expected that the MM would increase the subjects' motivation more than the SS would.

Comparison between the subjects' final motivation scores (i.e., a between-subjects effect) were made with Mann-Whitney U-tests. These tests showed no effect whatsoever of manual on any of the four motivational constructs. Apparently, MM-users came out of their training as motivated as did SS-users.

Computer experience did, however, affect the users' confidence and relevance scores after training. Surprisingly, the novices ended with a higher self-confidence than beginners ($U(42)=101.0$, $p<.05$) or intermediates ($U(46)=127.0$, $p<.01$). Novices and beginners ended with lower scores for relevance than intermediates ($U(48)=95.5$, $p<.01$ and $U(30)=46.5$, $p<.01$ respectively).

3.5 Conclusions

The main objective of this study was to find additional evidence in favor of the minimalist approach to computer documentation. As in Carroll's experiment, the MM was hypothesized to lead to superior procedural skill than the SS. MM-users were further expected to increase their motivation (e.g., confidence, satisfaction) more than control subjects. Effects of computer experience were studied in an exploratory fashion.

The first hypothesis is clearly supported by the results. The MM-subjects were superior to the SS-subjects during practice and on the performance test. The MM helped users to get started faster and MM-users needed less time to complete the training. MM-users also had higher performance scores: they completed more test items successfully and required less time to do so than SS-users. Moreover, MM-users made fewer errors and successfully recovered errors more frequently. No conclusions can be drawn with regard to the performance on the transfer items since ten SS-subjects did not process these items (as in the original experiment of Carroll *et al*., 1987).

Our findings thus confirm *all* of the results of Carroll *et al*. (1987), supporting the strengths of the minimalist approach. In addition, they also justify the conclusion that our 'translation' of Carroll's minimalist principles into more detailed design oriented guidelines (see also Van der Meij, 1992; Van der Meij & Lazonder, 1992) lead to the construction of a 'true' MM.

Surprisingly, the results do not support the second hypothesis. The MM did not increase the subjects' motivation more than the SS did. This may have to do with the overwhelming nature of the subjects' first experience with a word processor. To novice users, WordPerfect may seem a wonderful tool for creating, editing and formatting text. Casual observations during the experiment support this stance: subjects were thrilled by the (graphical) options of WordPerfect and by their laser-printed texts. Another explanation is that MM-users liked some, but not all, minimalist design characteristics. For example, they may think positively of the error-information, but, at the same time, dislike the 'learning by doing' approach prompted in the 'Do it yourself' sections.

The most important finding with regard to computer experience is that it did *not* interact with manual type. Apparently, the MM has a similar positive effect on novices, beginners and intermediates. In view of the general increase of computer experience, this is an important additional sign of the strength of the minimalist approach.

Computer experience did affect the speed and quality with which the subjects learned to use the word processor. For example, novices needed significantly more time, and they were less successful and less efficient for retention items than beginners or intermediates. In addition, they were less capable of recovering from errors and had lower efficiency scores for transfer items.

Somewhat surprisingly, the novices showed the highest gains in self-confidence. This speaks favorably of the quality of the program and the two manuals, but it is unclear why this is so. Do the novices not yet see the more complex problems that lie ahead? Likewise, it is unclear why intermediates had significantly higher relevance gains than novices or beginners.

3.6 Discussion

Whereas this study confirms the functionality of the minimalist approach, little is yet known about how the minimalist principles work. Future research should therefore aim to study minimalism in depth and in breadth.

3.6.1 IN-DEPTH EXPANSIONS

With two exceptions, research has concentrated on how *all* minimalist principles affect performance. Gong and Elkerton (1990) studied the effect of including error-information in two types of manuals (a MM and a SS). They found that the error-information helped to prevent errors. In addition, it speeded up the time subjects needed to complete the transfer tasks. Black, Carroll and McGuigan (1987) compared four manuals to examine, among others, the effect of adequate verbiage. They found that the amount of written material correlated positively to learning time and test time. Subjects who had less to read completed their training and test faster. Both studies thus showed facilitative effects of a single minimalist principle. In future, such work should be continued in order to increase our understanding of the operation of distinct minimalist principles, and to give insights into *how* these principles help people learn from (minimalist) documentation. Knowing how people use a manual (and how they learn from that) is fundamental to knowing how manuals may meet users' learning styles and preferences.

In-depth extensions might also start from a rational perspective. The minimalist philosophy was originally derived from observations of first-time computer users. Due to this empirical approach, some relevant user-characteristics may have been overlooked. By contrasting the minimalist approach with theories of learning and instruction, principles possibly omitted by the original observations can be uncovered. For example, behavioristic learning theories, and the literature on guided discovery learning and human-computer interaction give insight into how users deal with errors. This knowledge might point to new directions for designing error-information.

3.6.2 IN-BREADTH EXPANSIONS

It is important to expand the suitability of the minimalist approach to different levels of expertise and user groups. Research has traditionally focused on initial skill learning, the area for which the MM was originally designed. It is not self-evident that MM-subjects are better equipped to learn the more advanced word processing procedures than subjects trained with a conventional manual. Some authors have even argued that such a transfer is hampered when too much emphasis is put (too early) on the development of procedural skills (e.g., Jelsma, Van Merriënboer, & Bijlstra, 1990). Would this be also the case for MM-subjects? Research has yet to address this important issue.

The present study shows that computer experience has a significant effect on learning and test performance. What it does *not* tell is how this experience affects the processing of a MM. More experienced users are likely to activate other (computer) prior knowledge and thus have different learning needs (Schriver, 1986). Because the MM capitalizes on exploiting the subjects' prior knowledge and needs, the effects on novice and more experienced users are bound to differ.

The MM is not one of the most attractive manuals that we have come across. Although Carroll and his co-workers have addressed some issues of lay-out and typography, their attention to it has been minimal. This is unfortunate because it tends to lead to a separation of content and presentation. By linking the two, other ways to operationalize the minimalist principles come into view. For example, attempts are being made to construct a manual that 'slashes the verbiage' by substituting nearly all text by illustrations. In this, and other ways, the operationalizations of good design principles are a continuous challenge for research on documentation in the nineties.

References

Aizu, I., & Amemiya, H. (1985). The cultural implications of manual writing and design. *Proceedings of the 32nd International Technical Communication Conference (ITCC)* (pp. WE 33 - 35). Washington: Society for Technical Communication.

Anderson, J.R. (1985). *Cognitive psychology and its implications*. San Francisco: Freeman.

Black, J.B., Carroll, J.M., & McGuigan, S.M. (1987). What kind of minimal manual is most effective? *CHI + GI Proceedings* (pp. 159 - 162). New York: AMC.

Brockmann, R.J. (1990). *Writing better computer user documentation: From paper to hypertext* (2nd. edition). New York: Wiley.

Carroll, J.M. (1990*a*). An overview of minimalist instruction. *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Science* (pp. 210 - 219). Washington: IEEE.

Carroll, J.M. (1990*b*). *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill*. Cambridge: MIT.

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., & Mazur-Rimetz, S.A. (1987). The minimal manual. *Human-Computer Interaction, 3*, 123 - 153.

Chin, D.N. (1986). User modeling in UC: The Unix consultant. In M. Mantei & P. Orbeton (Eds.), *Human factors in computing systems-III: Proceedings of the CHI'86 conference* (pp. 24 - 28). Amsterdam: Elsevier.

Cuff, R.N. (1980). On casual users. *International Journal of Man-Machine Studies, 12,* 163 - 187.

Evers, A., & Lucassen, W. (1983). *Differentiële aanleg testserie (DAT'83): Analogieën* [Differential aptitude tests: analogies]. Lisse: Swets & Zeitlinger.

Foss, D.J., Smith-Kerker, P.L., & Rosson, M.B. (1987). On comprehending a computer manual: Analysis of variables affecting performance. *International Journal of Man-Machine Studies, 26*, 277 - 300.

Frese, M., Albrecht, K., Altmann, A., Lang, J., Von Papstein, P., Peyerl, R., Prümper, J., Schulte-Göcking, H., Wankmüller, I., & Wendel, R. (1988). The effect of an active development of the mental model in the training process: Experimental results in a word processing system. *Behavior and Information Technology, 7*, 295 - 304.

Gong, R., & Elkerton, J. (1990). Designing minimal documentation using a GOMS model: A usability evaluation of an engineering approach. In J. Carrasco Chew & J. Whiteside (Eds.), *Empowering People: CHI'90 Conference Proceedings* (pp. 99 - 106). New York: ACM.

Hallgren, C. (1992). The Nürnberg Funnel: A minimal collection. *The Journal of Computer Documentation, 16*, 11 - 17.

Horn, R.E. (1992). Commentary on the Nürnberg Funnel. *The Journal of Computer Documentation, 16*, 3 - 11.

Jelsma, O., Van Merriënboer, J.J.G., & Bijlstra, J.P. (1990). The ADAPT design model: Towards instructional control of transfer. *Instructional Science, 19*, 89 - 120.

Jensen, R.P., & Osguthorpe, R.T. (1985). Better microcomputer manuals: A research-based approach. *Educational Technology, 25*(9), 42 - 47.

Keller, J.M. (1983). Motivational design of instruction. In C.M. Reigeluth (Ed.), *Instructional-design theories and models: An overview of their current status* (pp. 383 - 434). Hillsdale: Erlbaum.

Keller, J.M. (1987). Development and use of the ARCS model of instructional design. *Journal of Instructional Development, 10* (3), 2 - 10.

Lazonder, A. W., & Van der Meij, H. (1992). *Towards an operational definition of the minimal manual* (Tech. Rep. No. IST-MEMO-92-02). Enschede, University of Twente, Dept. of Instructional Technology.

Lemos, R.S. (1978). Students' attitude towards programming: The effect of structured walk-throughs. *Computers & Education, 2*, 301 - 306.

Mackin, J. (1989). Surmounting the barrier between Japanese and English technical documents. *Technical Communication, 36*, 346 - 351.

O'Malley, C., Smolensky, P., Bannon, L., Conway, E., Graham, J., Sokolov, J., & Monty, M.L. (1983). A proposal for user centered system documentation. In A. Janda (Ed.), *Human Factors in Computing Systems: Proceedings of the CHI'83 Conference* (pp. 282 - 285). Amsterdam: Elsevier.

Nickerson, R.S. (1991). A minimalist approach to the "paradox of sense making". *Educational Researcher, 20*(9), 24 - 26.

Oatley, K., Meldrum, M.C.R., & Draper, S.W. (1989). *Evaluating self-instruction by minimal manual and by video for a feature of a word-processing system*. Unpublished manuscript, University of Glasgow.

Olfman, L., & Bostrom, R.P. (1988). The influence of training on use of end-user software. *Proceedings of the Conference on Office Information Systems* (pp. 110 - 118). New York: ACM.

Paxton, A.L., & Turner, E.J. (1984). The application of human factors to the needs of the novice computer user. *International Journal of Man-Machine Studies, 20*, 137 - 156.

Penrose, J.M., & Seiford, L.M. (1988). Microcomputer users' preferences for software documentation: An analysis. *Journal of Technical Writing and Communication, 18*, 355 - 366.

Popovich, P.M., Hyde, K.R., Zakrajsek, T., & Blumer, C. (1987). The development of the attitude toward computer usage scale. *Educational and Psychological Measurement, 47*, 261 - 269.

Raban, A. (1988). Word processing learning techniques and user learning preferences. *SIGCHI Bulletin, 20*, 83 - 87.

Ramsay, J.E., & Oatley, K. (1992). Designing minimalist tutorials from scratch. *Instructional Science, 21*, 85 - 99.

Redish, J.C. (1988). Reading to learn to do. *The Technical Writing Teacher, 15*, 223 - 233.

Reece, M.J., & Gable, R.K. (1982). The development and validation of a measure of general attitudes toward computers. *Educational and Psychological Measurement, 42*, 913 - 916.

Scharer, L.L. (1983). User training: Less is more. *Datamation, 29*, 175 - 182.

Schriver, K.A. (1986). *Designing computer documentation: a review of the relevant literature* (Tech. Rep. No. 31). Pittsburg: Carnegie-Mellon University, Communications Design Center.

Stevensen, D. (1992). After dinner address at SIGDOC 1991. *The Journal of Computer Documentation, 16*(2), 18 - 28.

Temple, L., & Lips, H.M. (1989). Gender differences and similarities in attitudes towards computers. *Computers in Human Behavior, 5*, 215 - 226.

Tripp, S.D. (1990). Book review of: The Nürnberg Funnel. *Educational Technology Research and Development, 38*(3), 87 - 90.

Vanderlinden, G., Cocklin, T.G., & McKita, M. (1988). Testing and developing minimalist tutorials: A case history. *Proceedings of the 35th International Technical Communications Conference*, 196 - 199.

Van der Meij, H. (1992). A critical assessment of the minimalist approach to documentation. *SIGDOC'92 Conference Proceedings*, 7 - 17.

Van der Meij, H., & Lazonder, A.W. (1992). *A constructivistic approach to computer documentation.* Paper presented at the second EARLI-SIG workconference on comprehension of verbal and pictorial information, Nijmegen, The Netherlands, November 2 - 3.

Wendel, R., & Frese, M. (1987). Developing exploratory strategies in training: The general approach and a specific example for manual use. In H.J. Bullinger, B. Schackel & K. Kornwachs, Eds. *Proceedings of the Second IFIP Conference on Human-Computer Interaction* (pp. 943 - 948). Amsterdam: Elsevier.

CHAPTER 4

Toward effective error control in minimalist documentation[8]

In learning to use software, people spend at least thirty percent of their time on dealing with errors. It could therefore be desirable to exploit users' errors rather than to avoid them. That is, to include error-information in a manual to support users in dealing with errors. An experiment was performed to examine the functionality of such error-information in a manual for a word processor. Two minimal manuals were compared, one containing error-information and one from which nearly all the error-information had been removed. Forty-two subjects were randomly assigned to one of the two conditions. Subjects who used the manual with error-information were expected to become more proficient at using the word processor (i.e., better constructive and corrective skills) and to develop more self-confidence. The results were equivocal. On some aspects of skill the error-information led to better performance (i.e., correcting syntactic errors). On others it had an adverse effect (i.e., detection of semantic errors and overall error correction time). Explanations are advanced for these findings and topics for further research are identified.

4.1 Introduction

One of the most striking features of first-time computer users is that they are active learners. They want to 'do' things in order to reach personal goals, rather than read through endless pages of what they consider to be 'just information' (Carroll, 1990*b*; Redish, 1988; Scharer, 1983; Wendel & Frese, 1987). Unfortunately, the instructional strategy of most computer manuals does not suit this spontaneous learning strategy. Most manuals require users to proceed step-by-step through endless series of contrived drill and practice exercises, giving them (too) little freedom for active (i.e., self-initiated) learning.

[8]Lazonder, A.W., & Van der Meij, H. (1994). Effect of error-information in tutorial documentation. *Interacting with Computers, 6*, 23 - 40. (with minor modifications).

The main reason for this is that trainees get themselves in trouble when they explore (Njoo & De Jong, 1993). Although these explorations can be advantageous in learning computer-related tasks (e.g., Kamouri, Kamouri & Smith, 1986; Van Joolingen, 1993), this view is not generally accepted. Consequently, detailed step-by-step instruction is supposed to prevent users from making mistakes. This assumption is unrealistic, however. Research (e.g., Carroll & Mack, 1984; Mack, Lewis & Carroll, 1987; Redish, 1988) has consistently shown that new users have problems following the descriptions that manuals provide. Frequently, they consider these directions paradoxical or irrelevant to their goals. As a consequence, novice users tend to explore the system on their own. As many as 65 percent of the users may skip information they consider irrelevant and use the manual only when they need help (Penrose & Seiford, 1988).

When novice users 'jump the gun', problems can and will arise. Just as in step-by-step instruction, mistakes occur during exploratory behavior. That is, the user may be blocked from any further exploratory actions. Research has consistently indicated that new users spend 30 to 50 percent of their time on detection and correction of errors (Bailey, 1983; Card, Moran & Newell, 1983; Graesser & Murray, 1990).

4.2 Errors and learning

In conventional manuals, errors are seen as 'blocks' to learning that can be avoided by detailed step-by-step instruction. This view is in line with classical theories (e.g., behaviorism) that aim for a minimum number of errors. These learning theories advocate error minimization because: (a) errors hinder the control over learning, (b) errors cause frustration, which, in the end, may cause a learner to stop learning, and (c) errorless learning is richer in association, because it prompts and explicitly relates new knowledge to existing knowledge (Glaser, 1965; see also Glaser & Bassok, 1989).

A contrasting, and more fruitful approach in view of the above, is to perceive errors as a wonderful opportunity for learning (e.g., Brown, Burton & deKleer, 1982; Carroll, 1990*a*; Singer, 1978). There are two reasons why for this. Firstly, the nature of the learning experiences should reflect the intended training outcomes as much as possible. Learning to operate a computer program means developing constructive and corrective skills. Users must learn not only how to do things, but also how to undo things that go wrong. That is, they should also learn how to deal with errors. Training should therefore focus on the development of both these procedural skills (Wendel & Frese, 1987). Secondly, errors signal misconceptions in the users' conceptual model (Brown *et al.*, 1982;

Pickthorne, 1983; Stevens, Collins & Goldin, 1982). Errors may thus help users to reveal and remove these misconceptions, and, consequently, develop a better conceptual model.

Errors will only have a positive effect if they are controlled in the learning process. That is, when the instruction supports the user's corrective skills development. This chapter investigates how such error control can be brought about. It first outlines the stages involved in dealing with an error. From this model, demands for effective error control are identified. The second part of this paper reports an experiment that tests whether a manual that meets these demands assists first-time users in developing word processing skills.

4.3 A general model of error-recovery

The main goal of users who have made an error is to return to a normal, or at least acceptable system state (Johannsen, 1988). In achieving this goal, users tend to go through three stages: (a) detection, (b) diagnosis, and (c) correction (Brown, 1983; Curry, 1981; Jelsma & Bijlstra, 1990; Wærn, 1991). The main assumption behind these stages is that all user-activity is goal-directed (cf. Ashcraft, 1989; Card *et al.*, 1983; Norman, 1986; Stillings *et al.*, 1987). A detailed outline of these stages is presented in the model below.

4.3.1 DETECTION

Error detection is the first step in recovery. It is conditional to the other stages: an error that is not detected can never be corrected.

An error is detected when a user considers an outcome to contrast with his or her original goal. More specifically, there are two ways in which error detection may be triggered (Allwood, 1984). Firstly, triggering may come in response to some external cue. For example, a user perceives a discrepancy between an outcome and some definitive yardstick of correctness (Guthrie, Bennett & Weber, 1991; Lewis, 1981). Secondly, detection can be prompted internally. That is, it can be initiated by the user on his or her own accord (Lewis, 1981). The user may, for example, feel insecure with the selected method, the command(s), or its execution.

Triggering is not a sufficient condition for detection. The user may, for example, abandon the pursuit of an error that is not important and does not interfere with task execution. Moreover, triggering does not always occur at the right moment. Misconceptions about the expected outcome, or the appropriateness of a solution method may lead to undetected errors or to a delay in the detection of an error. On the other hand, triggering can also occur if no error has been made. In that case, correct performance is judged as erroneous.

So, in addition to triggering, the user has to spot the error on the screen to actually detect it. Locating an error occurs by evaluating, or reviewing the current system state and the actions that were performed.

4.3.2 DIAGNOSIS

After detection, the nature of the error is still only vaguely known. The user merely knows that something has gone wrong. In diagnosis, the two main activities are finding out the exact nature of the error and its possible cause.

First, the error must be identified to understand its exact nature. That is, the system's error-state must be compared with the user's original goal. By comparison the discrepancy between the observed and the desired output becomes clear. Second, the user is likely to reason about what may have caused the error (McCoy Carver & Klahr, 1986). Especially in the case of a more fundamental mistake, the user will wonder about the solution method that was applied.

Whereas the diagnosis of the *nature* of an error is conditional to correction, the diagnosis of its *cause* is not always needed for correction (Rasmussen, 1986). However, it does help users to construct a better conceptual model.

4.3.3 CORRECTION

Correction contains four different kinds of user activity. The user must first *select a (repair) goal*. As the difference between where the user is now and where he or she wants to be is known, the goal is obvious. The gap between the actual and the desired output (i.e., the user's original goal) must be bridged. This is often done by sub-goal decomposition (e.g., Anderson, 1985; Frederiksen, 1984; Newell & Simon, 1972). For example, the user may divide the overall goal 'correct a typo' into three subgoals: move the cursor, delete the incorrect text, and type the correct text.

Next, the user must *plan the method*, for there may be more than one method of achieving the repair goal. To select the most appropriate method, the users decides which selection rules apply (Card *et al.*, 1983). Each of these rules has the form of an if-then statement. In the above example, one of the selection rules for cursor movements might be: 'if the document contains 1 page, then use the arrow-keys to move the cursor; if the document contains 2 pages or more, then use the search-command'.

The method is then translated into a physical action-sequence. The user *selects the commands* that will be used and determines the order in which they will be executed. The last action in the model is the *execution of the commands*. Execution is the first physical action in this model.

Errors may be given different statuses. Some errors will be easier to detect and/or correct than others. For that reason, errors are classified into one of the following categories: (a) semantic; (b) syntactic; and (c) slip (cf. Douglas & Moran, 1984; Lewis & Norman, 1986). A semantic error occurs when an inadequate command is chosen to achieve a given goal. For example, the user may select 'Base Font' to try to set a word in italics. Carrying out a correct command improperly is termed a syntactic error (e.g., changing the line spacing into 1½ instead of 1.5). Slips are small mistakes at the keystroke level (e.g., typing errors). In general, there is no research on how to deal with these three types of error in computer documentation. Therefore, this study will only explore any differential effects.

4.4 Toward effective error control

To bring about effective error control, a manual should support users in dealing with errors. Such control is possible by including error-information in the manual. In keeping with the staged error-recovery model, good error-information should consist of (a) a characterization of the system-state for detecting and identifying the error, (b) conceptual information about the likely cause of the error, and (c) action statements for correcting the error (Lang, Lang & Auld, 1981; Mizokawa & Levin, 1988; Roush, 1992). A typical example of error-information might thus read:

> *If the code [Hrt] appears, you pressed the* RETURN *key instead of the* F2 *key. Remove the [Hrt] code by pressing the* BACKSPACE *key. Press the* F2 *key to start the search as yet.*

Special attention should also be given to the appropriate timing of the error-information in order to reduce the number of delayed detections. Error-information should be presented frequently, often directly after the commands, rather than in separate 'trouble shooting' sections (Bailey, 1983;

Retrieving a document	
Before you can retrieve a document from disk, you must always clear the screen first.	
1. Go to the menubar and choose the command EXIT **2.** Press the ENTER key **3.** Answer both questions by typing an **N**	
You have cleared the screen.	
*If there is still text on the screen, you may have pressed the wrong key. Press the F7 key and type an **N** twice to clear the screen as yet.*	Retrieving a new document into the current document has severe consequences for further task execution
4. Go to the menubar and choose the command RETRIEVE **5.** Press the ENTER key	
*If the text **Document to be retrieved:** does not appear, you have selected the wrong command. Press the F1 key to rectify your choice.*	Error-information is presented directly after the commands it refers to.
6. Type MANUAL.TXT **7.** Press the ENTER key	
The document MANUAL.TXT appears on the screen	
If the screen remains empty, you have probably made a typing mistake. Retype the name of the document and press the ENTER key.	Typing a filename is error-prone to new users. Note that even with this simple error the detection-diagnosis-correction format can be applied.

Figure 4.1
Error-information in a manual. The left column shows an example page of the manual ***with*** *error-information that was used in the experiment (MM+). The corresponding principles for effective error control are presented in the right column.*

Carroll, 1990*b*; Horton, 1990; Lewis & Norman, 1986). As a rule of thumb, it is to be presented when errors have severe consequences for further task execution, or when commands are prone to error. An example of how error-information should be incorporated in a manual is shown in Figure 4.1.

To examine the efficacy of error-information, an experiment was conducted using a manual with error-information (MM+) and a control manual containing almost no error-information (MM-). It was expected that subjects who used a manual with error-information would develop better procedural skill than subjects who used a manual without error-information. More specifically, the MM+ subjects were expected to perform better on test items measuring constructive and corrective skills. Constructive skills are needed to achieve the user's original goals, whereas corrective skills are necessary to recover errors (i.e., meet the repair goal).

Error-information provides users with a safety net (Brown, 1983; Carroll & Mack, 1984; Cuff, 1980). It assures them that, no matter how odd the system's response may seem, nothing is wrong as long as the described error-state does not occur and that possible errors can be corrected at all times. MM+ subjects were therefore expected to become more confident as well.

4.5 Method

4.5.1 SUBJECTS

The experiment was part of an introductory computer course for first-year students in Instructional Technology. Forty-two students took part in the experiment, receiving course credits for participation. There were 10 males and 32 females with a mean age of 19.0 (*SD*=1.34). Subjects were randomly assigned to one of the two experimental conditions. There were 21 subjects in the MM+ group and 21 in the MM- group. All subjects had some experience with computers (games and/or applications), but very little or no experience with the software used in the experiment. Preliminary checks on the random allocation to conditions revealed no significant difference between the two groups with regard to age, sex, educational background and initial self-confidence. The mean prior experience with computers was equal for both groups as well.

4.5.2 MATERIALS

Experimental setting and word processor
All sessions took place in a computer class provided with a network of 19 Sirex 386-SX personal computers. The goal of the course was to teach ele-mentary word processing skills with the menu-driven version of WordPerfect 5.1. WordPerfect was downloaded from the network, thereby assuring an identical setup of the word processor for all subjects.

A registration program was installed on each computer. It stored the subjects' actions in a logfile. Every time a key was struck, time and keypress were recorded.

Instructional materials
Subjects from both groups received a minimal manual (MM+ or MM-) and a diskette containing all documents to be used in practice. Both manuals were designed especially for the experiment, varying only with regard to error-information. In the MM+, all the error-information was designed according to the criteria for effective error control (see Figure 4.1). The MM- contained no error-

information at all. It introduced the two main function keys for error-recovery in the first chapter, however. A more detailed description of the MM+ can be found in Carroll (1990*b*), Lazonder and Van der Meij (1992, 1993 [chapter 3]) and Van der Meij (1992).

Questionnaires and tests
A background questionnaire was used to gather some personal data such as age, sex, educational background, and computer experience.

Subjects' confidence was assessed by three questionnaires (see Appendix 5). The first questionnaire determined subjects' initial confidence; the second and third assessed their confidence after practice and after the tests, respectively. Each questionnaire contained 20 behavioral descriptions, nine of which were fillers. The subjects judged each description (e.g., "Working with computers scares me") on a 5-point agree-disagree scale. Pilot studies revealed satisfactory reliability scores for the questionnaires (Cronbach's alpha ≥ .90).

Three tests were administered to assess learning outcomes. One test measured the subjects' constructive skill. It contained 5 retention tasks (i.e., elementary word processing skills rehearsed during practice, such as removing text or changing the line spacing) and 5 transfer tasks (i.e., tasks not covered by the manuals, such as changing the position of the page number or adjusting the margins).

Two tests assessed the subjects' capacities for error-recovery: a knowledge and a skill test. The corrective *knowledge* test was a paper-and-pencil test. It contained three semantic errors, five syntactic errors and one slip. Each item presented a goal and a screendump, displaying the result of a set of actions to achieve that goal. For each item, the subjects had to mark all errors. For each detected error, its diagnosis and correction had to be specified as well. The corrective *skill* test was performed on the computer. The subjects had to detect and correct six errors (4 semantic, 2 syntactic) in a task document. Items of both tests were further classified into retention (i.e., included in the error-information) and transfer (i.e., not covered by the error-information). As manual type had no effect on these retention and transfer scores, these measures will not be reported.

4.5.3 PROCEDURE

The experiment was conducted in 4 groups of 7 to 13 subjects. In each group, half of the subjects were given a MM+ manual; the other half received a MM- manual. Separate seatings prevented interactions between MM+ and MM- subjects in a session. Within two weeks, all subjects attended two sessions of four hours each. In all, up to four and a half hour (maximally) were available for practice. The remaining time was used to complete the tests. The maximum time between sessions was 3 days. All procedures were identical for the various groups and the same two experimenters conducted all sessions.

At the beginning of the first session, the subjects filled in the background questionnaire and the initial confidence questionnaire. Next, they received instructions. The subjects were told to work individually and to consult the experimenter only when a system error had occurred or when they were stuck for more than 15 minutes. They were told to work in their own way and at their own pace. The subjects were asked not to work with WordPerfect between sessions. Checks indicated that they complied with this request. After the instruction, the subjects started practise.

The second session started with another hour of hands-on experience, enabling all subjects to complete practise. Directly after practise, the subjects filled in the second confidence questionnaire. After a short break, they were given the constructive skill test and the corrective skill test using a counter-balanced administration to control for order effects. After these tests, the subjects completed the corrective knowledge test. The subjects worked individually on all tests. They were not allowed to consult their manual or the experimenter. Enough time was given for all subjects to complete each test. After the tests, subjects filled in the final confidence questionnaire.

Coding and scoring of the dependent variables

The dependent variables were constructive skill, corrective skill and confidence. Constructive skill was defined by three measures: test time, success rate and number of errors. Test time was defined as the time required to complete the constructive skill test. A difference was made between retention and transfer. Success was indicated by the number of successfully completed items on the constructive skill test. This was assessed by examining the task documents stored on diskette and the log-files produced by each subject. The number of errors was registered for each item of the constructive skill test.

There were three measures of error-recovery skill: (a) detection; (b) diagnosis; and (c) correction, which were scored as follows. Detection was scored on a 2-point right-wrong scale. The inter-rater reliability for detection was high (Cohen's Kappa = .94). Diagnosis was scored on the following 4-point ordinal scale: (a) both cause and effect are incorrect; (b) wrong cause, right effect; (c) right cause, wrong effect; and (d) both cause and effect are correct. In a similar fashion, the correction method was scored as one that: (a) obviously does not try to correct the error; (b) attempts to correct the

Table 4.1
Mean test time scores

	Condition	
	MM+	MM-
Retention	10.80 (8.51)	9.74 (6.14)
Transfer	32.31 (21.17)	36.29 (19.39)
Total[a]	50.26 (21.83)	54.54 (18.83)

Note. There were 5 retention and 5 transfer items. Time in minutes, Standard deviations in parentheses.
[a] As the time between tasks could not be taken into account for these measures, the overall test time is higher than the time for the distinct item types.

error, but is both semantically and syntactically incorrect or incomplete; (c) is semantically correct, but contains one or more syntactic errors; and (d) is both semantically and syntactically correct. Inter-rater reliability scores for diagnosis and correction were .77 and .93, respectively. For each subject, the time to complete the corrective skill test was recorded as well.

The three confidence questionnaires used a 5-point Likert-type scale. Scores on all items were added for each subject with high scores representing high confidence. Confidence changes were examined within subjects.

Data analyses

The majority of the data were analyzed by means of (M)ANOVAs using manual type (MM+ or MM-) as independent variable. Mann-Whitney U tests were applied to analyze the ordinal data. The (within-subject) confidence changes were analyzed by Wilcoxon Matched-Pairs Signed Rank tests.

All outcomes were corrected for subjects' prior experience with computers by inserting this measure into the analyses as a covariate. Given the relative small sample size, effects of manual type on users with identical computer experience were not computed.

4.6 Results

4.6.1 CONSTRUCTIVE SKILL

Time

Table 4.1 shows the mean time (in minutes) subjects required to complete the constructive skill test. Manual type produced no significant effect on this measure ($F(1,38)=.41$). MM+ required as much time for completing the

Table 4.2
Mean performance success and performance effciency scores

	Condition	
	MM+	MM-
Performance success[a]		
Retention	3.71 (8.51)	4.00 (0.76)
Transfer	1.86 (1.35)	1.62 (1.02)
Performance efficiency[b]		
Retention	55.01 (39.15)	51.78 (26.10)
Transfer	13.22 (14.05)	11.99 (11.35)

Note. There were 5 retention and 5 transfer items. Standard deviations in parentheses.
[a] Number of items successfully completed. [b] Number of items successfully completed per time (min.) $\times$ 100.

constructive skill test as MM- users. Retention and transfer items were also analyzed separately. There was no effect of manual type on the time to complete retention items (F(1,38)=.64) and transfer items (F(1,38)=.39).

Quality of performance
Table 4.2 reports the performance success scores. There was no significant effect of manual type on performance success (F(1,39)=.09). Overall, MM+ users produced as many correct solutions as their MM- counterparts. There was also no difference in performance success on retention items (F(1,39)=1.00) and transfer items (F(1,39)=.53).

Time and the number of successfully completed test items were combined into a measure of performance efficiency. The mean efficiency scores are presented in Table 4.2. As can be seen from this Table, the mean scores show no significant difference between the two groups (F(1,36)=.47). Clearly, users from both experimental groups performed equally efficient. Efficiency scores on retention and transfer items were slightly higher for MM+ users. However, none of these differences were significant at the .05 level (F(1,37)=.05, F(1,37)=.06).

Errors
The error-rates of both groups were examined by comparing the mean number of errors to the number of successfully completed items. Again, a difference between retention and transfer items was made. The mean error-rates are shown in Table 4.3.

Table 4.3
Mean number of errors on correctly solved test items

	Condition	
	MM+	MM-
Retention	0.48 (0.51)	0.62 (0.57)
Transfer	2.71 (2.86)	2.62 (3.06)

Note. Scores are the number of errors by the number of successfully completed test items. Standard deviations in parentheses.

Manual type had no significant effect on the total number of errors (*F*(1,38)=.01), indicating that, overall, subjects in the MM+ group made as many errors as subjects in the MM- group. As the mean error-rates show, MM+ users committed as many errors as MM- users on retention and transfer items. Again, manual type had no effect on the number of errors on retention items (*F*(1,38)=.65) and transfer items (*F*(1,38)=.01).

4.6.2 CORRECTIVE SKILL

There were three measures to assess error-recovery: (a) detection; (b) diagnosis; and (c) correction.

Detection
The number of detected errors were recorded on the corrective knowledge test and the corrective skill test. The mean number of detected errors are presented in Table 4.4.

As the mean detection scores show, the MM+ users detected more errors on the corrective knowledge test than MM- users. However, this difference was not statistically significant: a MANOVA on manual type by the number of detected semantic errors, syntactic errors and slips showed no multivariate effect (*F*(3,37)=.92).

On the corrective skill test, a t-test on the total number of detected errors by manual type produced no significant effect (*t*(40)=-.99). Again, the MM- group detected as many errors as the MM+ group. Manual type did affect the detection of semantic errors (*t*(40)=-2.32, *p*<.05). But, contrary to expectations, the MM- users detected *more* errors than the MM+ users. No effect of manual type on the number of detected syntactic errors was found (*F*(1,39)=.06).

Table 4.4
Mean number of detected errors

Error-type	Condition	
	MM+	MM-
Corrective Knowledge Test[a]		
Semantic	1.86 (0.79)	1.48 (1.03)
Syntactic	2.48 (0.98)	2.14 (0.91)
Slip	0.52 (0.51)	0.43 (0.51)
Corrective Skill Test[b]		
Semantic	3.67 (0.66)	4.00 (0.00)*
Syntactic	1.67 (0.48)	1.57 (0.51)

Note. Standard deviations in parentheses.
[a] Maximum score = 9 [b] Maximum score = 6
* p<.05

Diagnosis

The mean scores of the quality of the diagnoses are shown in Table 4.5. Overall, there was no difference in diagnosis scores between the two groups (*U*(42)=172.5). Apparently, the quality of the diagnoses of the MM- users was equal to that of the MM+ group. As the mean ranks in Table 4.5 indicate, the two groups hardly differed with respect to their diagnoses on the distinct error-types as well. Manual type had no effect on the diagnoses of semantic errors (*U*(42)=199.0), syntactic errors (*U*(42)=216.5) or slips (*U*(42)=199.5).

Table 4.5
Mean rank scores of the quality of the diagnoses[a]

Error-type	Condition	
	MM+	MM-
Semantic	20.48	22.52
Syntactic	21.69	21.31
Slip	22.50	20.50

Note. Diagnoses were registered on the Corrective Knowledge Test only. *n*=21 for both conditions.
[a] Higher rank means higher quality

Table 4.6
Mean rank scores of the quality of correction

Error-type	Condition	
	MM+	MM-
Corrective Knowledge Test[a]		
Semantic	20.17	22.83
Syntactic	25.00[*]	18.00
Slip	22.10	20.90
Corrective Skill Test[b]		
Semantic	20.98	22.02
Syntactic	21.00	22.00

Note. n=21 for both conditions.
[a] Maximum score = 9 [b] Maximum score = 6
[*] p<.05

Correction
The mean correction scores of both tests are shown in Table 4.6. On the whole, the MM+ users were not better at correcting errors on the corrective knowledge test. Manual type had no significant effect on the total correction score (U(42)=214.5). As can be seen from Table 4.6, there was an effect of manual on the correction of syntactic errors (U(42)=147.0, p<.05), indicating that the MM+ users were better at correcting syntactic errors than their MM- counterparts. No effect was found on correction of semantic errors (U(42)=192.5) and slips (U(42)=208.0).

On the corrective skill test, again no difference on the total correction score was found (U(42)=217.0). Apparently, MM+ users were as good at correcting errors as MM- users. The mean rank scores indicate that there was no significant difference between correction scores on semantic errors (U(42)=209.5), and syntactic errors (U(42)=210.0).

Table 4.7 presents the mean time (in minutes) subjects required to complete the corrective skill test. Overall, MM- users completed this test more than 7 minutes *faster* than MM+ users. This difference was statistically significant (t(40)=2.37, p<.05). Time and the correction score were combined into a measure of correction-efficiency (see Table 4.7). Although the MM+ users were expected to be more efficient with respect to this measure, the opposite turned out to be true. Manual type had a significant effect on correction-efficiency (U(42)=127.0, p<.05) indicating that the MM- group corrected errors more efficiently than the MM+ group.

Table 4.7
Mean time and recovey efficiency scores on the Corrective Skill Test

	Condition	
	MM+	MM-
Time	23.30 (12.40)*	16.00 (6.76)
Efficiency[a]	18.20 (9.80)*	23.80 (7.70)

Note. Time in minutes, Standerd deviations in parentheses.
[a] Efficiency = mean correction score per time × 100.
* *p*<.05

4.6.3 CONFIDENCE

Error-information provides users with a safety net. Therefore, MM+ users were expected to gain more self-confidence than MM- users. The mean differences in self-confidence are presented in Table 4.8.

In the MM+ group, confidence scores remained relatively constant. There was a small increase in confidence after practice, and a small decrease after the tests. None of these differences were statistically significant. Confidence changes were similar for MM- users. However, the difference between confidence scores after practice and after the tests was significant for this group ($Z(19)=-2.63$, $p<.01$). The MM- users' confidence after the tests was lower than after practice.

Table 4.8
Within-subject differences in self-confidence

	Condition	
	MM+	MM-
B — A_1	0.05 (0.56)	0.28 (0.67)
B — A_2	-0.09 (0.69)	0.03 (0.43)
A_1 — A_2	-0.10 (0.52)	-0.36 (0.48)*

Note. B = confidence before training; A1 = confidence after training; A2 = confidence after the tests. Standard deviations in parentheses.
* *p*<.01

4.7 Discussion

This study examined the effect of error-information on users' procedural skills and levels of self-confidence. Subjects who used a manual with error-information were expected to develop better constructive *and* corrective skill and to gain a higher level of self-confidence than subjects who used a manual without error-information. In general, there is no effect of error-information on these measures. However, some results reveal new and interesting insights into how error-information might affect user behavior.

The first hypothesis, which stated that MM+ users would develop better constructive skill, was not supported by the results. Subjects from both conditions performed equally well on the constructive skill test. There was no difference between the two groups regarding the time to complete the test items, the number of items successfully completed or the number of errors.

Why didn't the MM+ have a facilitative effect on subjects' constructive skill? Firstly, subjects' errors on the constructive skill test were not the kind of errors addressed by the error-information in the manual. Most error-information in the MM+ deals with syntactic errors. The MM's short chapters and action-oriented headings explicitly denote when commands have to be used. Information to recover semantic errors (i.e., the choice of an incorrect command) is therefore hardly ever presented. Post-hoc analysis of the constructive skill test indicated that no fewer than 84% of the errors subjects made were semantic errors. Only 15% of the errors were syntactic; 1% were slips. Since subjects' errors were for the most part *not* overcome by the error-information, MM+ users were not better trained to detect and correct most of their own errors. Consequently, MM+ users were not faster in completing the constructive skill test and produced as many correct solutions on test items as MM- users.

Secondly, the functionality of error-information is affected by its actual use. During practice, subjects can utilize error-information to correct an error or to explore the effect of a proposed correction method (see Van der Meij, 1992). This study provides no information as to whether the subjects have consulted the error-information. Subjects may not have made a given error, or they may not have explored the correction method.

The MM+ was further expected to be superior to the MM- for corrective skill. This hypothesis too was not supported by the results. On the corrective knowledge test, the MM- users detected as many errors as MM+ users. On the corrective skill test, the MM- group was faster and detected more semantic errors. Moreover, the two groups were equally proficient at diagnosing the cause of an error. With regard to correction, again no pronounced difference between the groups occurred on the two tests. The MM+ users were better at correcting syntactic errors on the corrective knowledge test, whereas the MM- users achieved higher correction efficiency scores on the corrective skill test.

The reason why MM+ users were better at correcting syntactic errors can be accounted for by the error-information. Since error-information mainly addressed syntactic errors, MM+ users were better trained in correcting these errors than MM- users. The fact that the scores on the corrective skill test do not support this explanation can be ascribed to the correction methods subjects used on this test. Nearly all subjects used re-constructive methods to correct errors, meaning that instead of undoing actions, they simply performed those actions again. For example, to undo an incorrect line spacing, subjects inserted a new line spacing code instead of removing the old, incorrect code. Although such methods will often be effective in working with WordPerfect, they are less likely to be applied on the corrective knowledge test. Consequently, corrective methods were used on this test, and MM+ users were better trained in using these methods for syntactic errors. The issue of how subjects correct their errors should be addressed in future studies.

There are several reasons why the other expected findings failed to appear. Firstly, the scores on the corrective skill test point at a ceiling-effect. Although the data show a significant difference between the two groups with regard to the detection of semantic errors, their true magnitude cannot be established. For detection, this ceiling-effect may be caused by system cues or the word processor's help function. These build-in resources may have biased the number of errors detected. The results from the corrective skill test can therefore not be seen as an adequate reflection of the actual number of detections.

Secondly, although preliminary checks on random allocation of subjects to conditions indicated that both groups were identical with respect to prior experience with computers, within-group differences existed. These differences may have affected the assessment of recovery skill. Because more experienced users have a richer, more elaborate conceptual model than less experienced users, their conceptual model allows for a better, more meaningful incorporation of new information. Consequently, they are assumed to benefit more from error-information. Future research should therefore focus on how prior experience with computers affects the development of recovery skill.

Thirdly, as with constructive skill, the actual use of error-information may have affected corrective skill. In case error-information is not used or explored, MM+ users are not better trained to detect and correct an error than MM- users. Because this experiment revealed no information on how subjects dealt with errors and error-information during practice, its true effect on subjects' constructive and corrective skill cannot be established. In future research on error-information, quantitative results should therefore be supported by (observational) data regarding subjects' activity during practice. Another question for future research is the effect of the cues and prompts generated by the software. During practice, the effect of error-information might have been overshadowed by the effect of the cues of the word processor. In future studies, this effect can be eliminated by removing all system cues or by counting the number of times a subject uses this information. Such experiments require an experimental setting that differs from the one that was used here. Individual subjects should be observed during practice as well as during the tests. Not only do these observations provide information on the use of system cues during practice and on the tests, they also reveal more about the actual use of the error-information.

The third hypothesis regarding subjects' self-confidence was partly supported by the results. The error-information in the MM+ did not cause users to develop higher self-confidence; confidence scores for MM+ users remained rather constant. In the MM- group, however, self-confidence scores after the tests were significantly lower than the scores after practice. So, although the expected increase in self-confidence failed to occur, error-information did have a positive effect on self-confidence.

The reasons for the MM+ users' self-confidence to remain constant rather than increase might be that subjects did not know in advance that manual usage was not allowed during the test phase. Intermittent confidence scores might therefore reflect subjects' self-confidence in word processing *with* the use of a manual. Confidence scores after the tests thus represent self-confidence *without* the use of the manual. Since most users had little or no experience with WordPerfect, the absence of a manual could have lowered their final confidence score.

The present study showed error-information to have hardly any effect on procedural skill (both constructive and corrective). From these findings, one might conclude that including error-information in a manual only yields an increase in amount of written information (something minimal manuals can do without!). This conclusion is premature, however. Although error-information had no effect on learning outcomes, it could have supported users in the *development* of procedural skills. As the effect of error-information on users' activity during practice is still unknown, it can not be decided as yet whether the inclusion of error-information in a minimal manual is indeed functional.

References

Allwood, C.M. (1984). Error detection processes in statistical problem solving. *Cognitive Science, 8,* 413 - 437.

Anderson, J.R. (1985). *Cognitive psychology and its implications.* San Francisco: Freeman.

Ashcraft, M.H. (1989). *Human memory and cognition.* Glenview: Scott Foresman.

Bailey, R.W. (1983). *Human error in computer systems.* Englewood Cliffs: Prentice-Hall.

Brown, J.S. (1983). Learning by doing revisited for electronic learning environments. In M.A. White (Ed.), *The future of electronic learning* (pp. 13 - 33). Hillsdale: Lawrence Erlbaum.

Brown, J.S., Burton, R.R., & deKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOFIE I, II and III. In D. Sleeman & J.S. Brown (Eds.), *Intelligent tutoring systems* (pp. 227 - 282). London: Academic Press.

Card, S.K., Moran, T.P., & Newell, A. (1983). *The psychology of human-computer interaction.* Hillsdale: Lawrence Erlbaum.

Carroll, J.M. (1990*a*). An overview of minimalist instruction. *Proceedings of the Twenty-Third annual Hawaii International Conference on System Science* (pp. 210 - 219). Washington: IEEE.

Carroll, J.M. (1990*b*). *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill.* Cambridge: The MIT Press.

Carroll, J.M., & Mack, R.L. (1984). Learning to use a word processor: By doing, by thinking and by knowing. In J.C. Thomas & M.L. Schneider (Eds.), *Human factors in computer systems* (pp. 13 - 51). Norwoord: Ablex.

Cuff, R.N. (1980). On casual users. *International Journal of Man-Machine Studies, 12,* 163 - 187.

Frederiksen, N. (1984). Implications of cognitive theory for instruction in problem solving. *Review of Educational Research, 54,* 363 - 407.

Glaser, R. (1965). Toward a behavioral science base for instructional design. In R. Glaser (Ed.), *Teaching machines and programmed learning, volume 2* (pp. 771 -809). Washington: Association for Educational Communications and Technology.

Glaser, R., & Bassok, M. (1989). Learning theory and the study of instruction. *Annual Review of Psychology*, *40*, 631 - 666

Graesser, A.C., & Murray, K. (1990). A question-answering methodology for exploring a user's acquisition and knowledge of a computer environment. In S.P. Robertson, W. Zachary & J.B. Black (Eds.), *Cognition, computing and cooperation* (pp. 237 - 267). Norwood: Ablex.

Guthrie, J.T., Bennett, S., & Weber, S. (1991). Processing procedural documents: A cognitive model for following written directions. *Educational Psychology Review, 3*, 249 - 265.

Horton, W. (1990). *Designing and writing online documentation: Help files to hypertext.* New York: Wiley.

Jelsma, O., & Bijlstra, J.P. (1990). Process: Program for research on operator control in an experimental setting. *IEEE Transactions on Systems, Man, and Cybernetics, smc-20,* 1221 - 1228.

Johannsen, G. (1988). Categories of human operator behavior in fault management situations. In L.P. Goldstein, H.B. Andersen & S.E. Olsen (Eds.), *Tasks, errors and mental models* (pp. 251 - 258). London: Taylor & Francis.

Kamouri, A.L., Kamouri, J., & Smith, K.H. (1986). Training by exploration: Facilitating the transfer of procedural knowledge through analogical reasoning. *International Journal of Man-Machine Studies, 24*, 171 - 192.

Lang, T., Lang, K., & Auld, R. (1981). A longitudinal study of computer-user behavior in a batch environment. *International Journal of Man-Machine Studies, 14*, 251 - 268.

Lazonder, A.W., & Van der Meij, H. (1992). *Towards an operational definition of the minimal manual* (Tech. Rep. No. IST-MEMO-92-02). Enschede, University of Twente, Dept. of Instructional Technology.

Lazonder, A.W., & Van der Meij, H. (1993). The minimal manual: Is less really more? *International Journal of Man-Machine Studies, 39*, 729 - 752.

Lewis, B.N. (1981). An essay on error. *Instructional Science*, 10, 237 - 257.

Lewis, C. & Norman, D.A. (1986). Designing for error. In D.A. Norman & S.W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction* (pp. 411 - 432). Hillsdale: Lawrence Erlbaum.

Mack, R.L., Lewis, C.H., & Carroll, J.M. (1987). Learning to use a word processor: Problems and prospects. In R.M. Baeker & W.S. Buxton (Eds.), *Reading in human-computer interactions: A multidisciplinary approach* (pp. 269 - 277). Los Altos: Morgan Kaufmann.

McCoy Carver, S., & Klahr, D. (1986). Assessing children's LOGO debugging skills with a formal model. *Journal of Educational Computing Research, 2,* 487 - 525.

Mizokawa, D.T., & Levin, J. (1988). Standards for error messages in educational software. *Educational Technology, 28*(4), 19 - 24.

Newell, A., & Simon, H.A. (1972). *Human problem solving.* Englewood Cliffs: Prentice Hall.

Njoo, M., & De Jong, T. (1993). Exploratory learning with a computer simulation for control theory: Learning processes and instructional support. *Journal of Research in Science Teaching, 30*, 821 - 844.

Norman, D.A. (1986). Cognitive engineering. In D.A. Norman & S.W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction* (pp. 31 - 61). Hillsdale: Lawrence Erlbaum.

Penrose, J.M., & Seiford, L.M. (1988). Microcomputer users' preferences for software documentation: An analysis. *Journal of Technical Writing and Communication, 18*, 355 - 366.

Pickthorne, B. (1983). Error factors: A missing link between cognitive science and classroom practice. *Instructional Science, 11*, 283 - 312.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering.* New York: Elsevier.

Redish, J.C. (1988). Reading to learn to do. *The Technical Writing Teacher, 15*, 223 - 233.

Roush, R. (1992). Taking the error out of explaining error messages. *Technical Communication, 39*(1), 56 - 59.

Scharer, L.L. (1983). User training: Less is more. *Datamation, 29*(7), 175 - 182.

Singer, R.N. (1978). Motor skills and learning strategies. In H.F. O'Neil (Ed.), *Learning strategies* (pp. 79 - 106). New York: Academic Press.

Stevens, A., Collins, A., & Goldin, S.E. (1982). Misconceptions in students' understanding. In D. Sleeman & J.S. Brown (Eds.), *Intelligent tutoring systems* (pp. 13 - 50). London: Academic Press.

Stillings, N.A., Feinstein, M.H., Garfield, J.L., Rissland, E.L., Rosenbaum, D.A., Weisler, S.E., & Baker-Ward, L. (1987). *Cognitive Science: An introduction.* Cambridge: MIT.

Van der Meij, H. (1992). A critical assessment of the minimalist approach to documentation. *SIGDOC'92 Conference Proceedings*, 7 - 17.

Van Joolingen, W.R. (1993). *Understanding and facilitating discovery learning in computer-*

based simulation environments. PhD thesis, Eindhoven University of Technology, The Netherlands.

Wærn, Y. (1991). On the microstructure of learning a wordprocessor. *Acta Psychologica, 78*, 287 - 304.

Wendel, R., & Frese, M. (1987). Developing exploratory strategies in training: The general approach and a specific example for manual use. In H.J. Bullinger, B. Schackel & K. Kornwachs (Eds.), *Proceedings of the second IFIP conference on human-computer interaction* (pp. 943 - 948). Amsterdam: Elsevier.

CHAPTER 5

Verifying the preconditions for error-based learning[9]

Human errors can play a useful role in learning to use software. However, whether people actually learn from their errors depends on the degree to which they are controlled in the learning process. In a minimal manual, such error control is assumed to be brought about by the error-information. An experiment was performed to validate this assumption. Eight subjects were given a minimal manual for a word processor. During practice, the experimenter recorded their learning activities. The results indicated that approximately 40% of all errors were supported by the error-information in the manual. Moreover, error-information was frequently consulted to detect and correct errors and to check if an error had occurred. In the discussion, suggestions to further improve the manual are identified.

5.1 Introduction

People make errors when they try to learn something new. This is particularly true of computer systems. No matter how user-friendly the software and the training manual, new users will undoubtedly make errors.

Although behaviorists consider(ed) errors to be undesirable in learning, the current view is that errors are a valuable opportunity to clarify misconceptions in the learner (Mory, 1992). However, errors do not necessarily play a useful role in learning. Errors will only have a positive effect when they are controlled in the learning process. That is, when the learners are supported in dealing with their own erroneous actions.

Such error control is possible by including error-information in the manual. To identify the conditions under which error-information will be most effective, a model of error-recovery was developed, explaining what happens

[9]Lazonder, A.W. (1994). Minimalist documentation and the effective control of errors. In M. Steehouder, C. Jansen, P. van der Poort & R. Verheijen (Eds.), *Quality of technical documentation* (pp. 85 - 98). Amsterdam: Rodopi. (with modifications)

when someone tries to recover an error (Lazonder & Van der Meij, 1994 [chapter 4]). According to this model, the process of undoing errors expires from detection through diagnosis to correction. In the detection phase, the user observes that 'something is wrong'. During the diagnosis phase, the user will reason about the exact nature of the error and its most likely cause. Based on the outcome of this phase, the user selects and executes a correction method.

To allow for effective error control, the error-information in the manual should be designed in line with the stages of this model. Good error-information should therefore consist of: (a) a characterization of the system-state to detect and identify the error, (b) conceptual information about the likely cause of the error, and (c) action statements, for correcting the error.

The facilitative effect of error-information has yet to be proven. An early experiment showed that error-information exerts hardly any effect on learning outcomes (Lazonder & Van der Meij, 1994 [chapter 4]). The experiment produced some promising results, however. The authors therefore concluded that error-information in manuals should be further investigated, taking into account the fact that effect of error-information depends on a number of factors, some practical, some theoretical.

On the theoretical side, a distinction is necessary between correction methods. In general, an error can be corrected by undoing actions (corrective method) or by (adequately) performing those actions again (reconstructive method). Examples of both correction methods are presented in Lazonder and Van der Meij (1993 [chapter 6]) and in Van der Meij and Carroll (in press). If people mainly apply reconstructive correction methods, only the detection and diagnosis part of the error-information will contribute to its effect.

At least four considerations are important on the practical side. Firstly, one must be sure that the learners make (enough) errors. There is ample evidence that they do: research showed that new users spend 30 to 50% of their time on dealing with errors. (e.g., Bailey, 1983; Graesser & Murray, 1990; Van der Meij, 1992). Secondly, a significant part of these errors must be supported by the error-information. If not, the error-information can be of little use in error-recovery. Thirdly, the users must actually use the error-information to recover errors. Fourthly, the experiment must adequately assess the processing of error-information both during practice and during the test phase.

The effect of error-information can then be examined by contrasting a manual *with* error-information with a manual from which all error-information is removed. The main hypotheses of this experiment relate both to the learning activities and learning outcomes. Learners who use a manual containing error-information are hypothesized to produce higher scores on both measures. That is, they are expected to make less errors during practice and to recover errors faster. As a consequence, they are expected to need less time to complete practice. Moreover, these learners are hypothesized to perform better on test items measuring constructive and corrective skills.

A small-scale study was conducted to examine whether error-information does in fact support learning (i.e., the practical considerations). Subjects were given a minimal manual for a word processor. The manual contained error-information that was designed according to the criteria for effective error control. Subjects were expected to fulfil all necessary conditions, that is, they were expected to make errors (that are supported by the error-information) and to use the error-information to recover these errors.

5.2 Method

5.2.1 SUBJECTS

Eight subjects (2 men and 6 women) volunteered in this study. They were recruited by means of an advertisement in a local newspaper. The subjects' age varied: the mean age was 38,5, with a range of 23 to 55. The subjects' educational level was also widely divergent: 2 subjects had finished lower general secondary education while 1 subject had a university degree. All subjects had little or no computer experience and no experience with word processing.

5.2.2 MATERIALS

Technical equipment

The goal of the experiment was to teach basic word processing skills with the menu-driven version of WordPerfect 5.1. WordPerfect was run on an Olivetti 286 personal computer. To keep subjects from using system cues, WordPerfect's help-function was disabled. A registration program was installed on the computer. It generated a logfile of all of the subjects' keystrokes. Whenever a key was struck, time and keypress were recorded.

The experimenter used the ERR-system (Error-Recovery Registration System) to register subjects' actions in dealing with errors. The ERR-system is completely mouse-controlled and runs on an Apple Macintosh computer under HyperCard. By clicking icons, the experimenter can record when (a) an error is made, (b) an error is detected, and (c) the error state is ended. When a given icon is clicked, additional information (e.g., type of error, quality of the solution) can be entered for that measure.

Instructional manual

The manual was a minimal manual, designed especially for the experiment. In all, approximately 20% of the manual consisted of error-information. More specifically, error-information was included at a rate of about once after every 3 action steps. All the error-information was designed according to the criteria for effective error control. A detailed description of the manual's design principles can be found in Carroll (1990), Lazonder and Van der Meij (1992, 1993 [chapter 3]) and Van der Meij (1992).

5.2.3 PROCEDURE

All sessions took place in a quiet room. Each session lasted one day, with a maximum of 8 hours. There were short breaks for coffee and lunch.

At the outset of the session subjects received instructions. They were told to work through the manual at their own pace. The subjects were instructed to deal with errors themselves. The experimenter would not offer any help, except when a system error had occurred or when they were stuck for more than 15 minutes. In addition, they were asked to think aloud. This was rehearsed on a simple task (tie a knot in a rope).

After the instruction, subjects were seated at a desk with the computer and the printer. They received a manual and a diskette containing all documents to be used in practice. During practice, the experimenter sat at a table nearby to record a subject's corrective actions.

Coding and scoring

Three measures were used to assess whether the conditions for error-based learning are fulfilled: number of errors, type of errors and use of error-information.

An error was defined as every action that does not contribute to the user's goal. The user's goal is represented by the action steps in the manual. The total number of errors was registered during practice.

Similar to Lazonder and Van der Meij (1994 [chapter 4]), errors were classified as semantic, syntactic or slip. Errors were further classified as supported and not-supported by the error-information. An error is supported by the error-information when the error-state – and therefore the correction method – is specified by the error-information. When the error-information does not embody the error-state, the error is classified as not-supported.

With the use of error-information, a difference was made between correct and incorrect performance. During correct performance, the number of times error-information was read, used and explored was recorded. Error-information is read when the user merely reads the error-information. Error-infor-

Table 5.1
Number and type of errors

	Supported[a]	Not supported[b]	Total
Semantic	32	71	103
Syntactic	35	93	128
Slip	11	42	53
Total	78	206	284

[a] Number of errors supported by error-information [b] Number of errors *not* supported by error-information

mation is used when the users reads the error-information and looks at the screen to check whether the described error-state has occurred. Error-information is explored when the user reads the error-information and tries out the suggested correction method. In case of an error, it was registered whether error-information was used to detect or to correct the error.

All data were registered by the ERR-system. To increase reliability, these data were compared to the data from the logfiles.

5.3 Results

5.3.1 NUMBER AND TYPE OF ERRORS

Table 5.1 shows the number of errors. In all, the subjects made 284 errors during practice. The mean error score was 35.50 (*SD*=10.07), with a range of 20 to 50.

As the column totals indicate, there were 103 semantic errors (36%), 128 syntactic errors (45%) and 53 slips (19%).

Approximately one third of all errors were supported by the error-information in the manual. The remaining portion of errors were not-supported. The mean number of supported errors was 9.75. The mean number of errors that are not supported by the error-information was 25.75.

5.3.2 USE OF ERROR-INFORMATION

The frequency with which the error-information was used during correct performance (constructive use) as well as during incorrect performance (corrective use) are presented in Table 5.2.

As this table indicates, the subjects frequently read and used the error-

Table 5.2
Use of error-information

User activity	#	*M*	*SD*	Range
Constructive				
Read	159	19.88	5.87	11-31
Used	168	21.00	5.95	13-28
Explored	14	1.75	3.41	0-10
Corrective				
Detection	31	37.20[a]	23.73	13-78
Correction	55	71.63[b]	28.84	30-86

Note. The corrective use of error-information is computed for supported errors only.
[a] Percentage of errors detected with the use of error-information [b] Percentage of errors corrected with the use of error-information

information as part of their constructive activity. On average, error-information was read 20 times and was used 21 times during correct performance. Error-information was explored less frequently. The mean number of explorations was 1.75, with a range of 0 to 10.

During incorrect performance, error-information was used to detect or correct errors. Approximately 37% of the supported errors was detected with the use of the error-information in the manual. The subjects used the error-information to correct an error more frequently. The error-information was applied for this purpose in 70% of the occasions. The range for this measure was 30 to 86%. In all, subjects spent 25.33% of their time on error-recovery (*SD*=4.20).

5.4 Discussion

This study examined the practical conditions for error-based learning from minimalist documentation. As the main hypothesis, it was stated that subjects who used a manual with error-information would satisfy all the practical conditions for error-information to have a facilitative effect on learning.

The first hypothesis, which stated that the subjects would make errors, is clearly supported by the results. On average, the subjects made 36 errors; the number of errors varied from 20 to 50. When compared with the number of action steps in the manual, it turns out that approximately one out of four actions was performed incorrectly. Considering the fact that it took subjects

25% of their time to recover from these errors, this error frequency seems adequate to develop and practice error-recovery skills. Moreover, as practice is still primarily concerned with developing constructive skills (i.e., 75% of practice time), it seems unlikely that subjects will get discouraged by the abundance of their errors and, as a consequence, abandon training.

The second hypothesis regarding the number of errors that are supported by the error-information, is also supported by the results. Approximately thirty percent of the errors were supported. Although this percentage does in itself not seem extremely high, it is satisfactory. The error-information in a minimal manual is designed to cover the most frequently occurring errors. It is therefore unrealistic, if not impossible, to expect error-information to cover every possible error. In addition, subjects have different styles of using the manual. Some spell out everything, while others process the information in the manual in a more exploratory fashion. As the subjects who explore do not follow the manual step-by-step, they will encounter problems different from those covered by the error-information in the manual.

Moreover, the data from the observations provide valuable insight into the subjects' most prevalent errors. These findings suggest several ways to further improve the error-information. Firstly, the coding of supported errors should be modified for the 'Do it yourself' sections. Many errors were made in these sections. Due to the nature of these sections these errors were not supported 'on the spot', but in the section that is referred to. By coding these errors as supported, the number of supported errors will increase considerably. Secondly, error-information was designed to support the most prevalent errors. Which errors are prevalent mainly depends on a subject's prior computer experience. By selecting subjects whose computer experience corresponds with the errors that are supported in the manual, the number of supported errors is further increased. This study further revealed some of the typical problems users encounter in learning a word processor. Based on this information, the error-information in the manual can be further improved.

Subjects were further expected to use error-information to detect and correct errors. This hypothesis is also supported by the results. Subjects used error-information to detect errors as well as to correct them. The results showed that error-information was consulted more often to correct an error than to detect one (37% and 72% respectively). This may be due to the fact that, on executing a command, subjects almost immediately look at the screen to see its result. In case of an error, the subject will notice that "something is wrong" before they return to the manual. Errors are therefore more likely to be detected while the subject is looking at the screen than when he or she is reading the manual. When an error is detected by looking at the screen, the user often returns to the manual for its correction. The error-information is then used for correction only.

In addition, subjects frequently consulted the error-information to check whether the described error-state had occurred. The constructive use of error-information indicates that subjects evaluated their actions regularly. The error-information thereby provides them with a safety net. They are assured that nothing is wrong as long as the described error-state does not occur. The users are thus prevented from recovering so called false alarms (i.e., undoing correct actions they consider erroneous).

The present study showed all practical conditions for error-based learning to be fulfilled. The outcomes of this study therefore allow for a valid assessment of the functionality of error-information in minimalist tutorials.

References

Bailey, R.W. (1983). *Human error in computer systems.* Englewood Cliffs: Prentice-Hall.

Carroll, J.M. (1990). *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill.* Cambridge: MIT.

Graesser, A.C., & Murray, K. (1990). A question-answering methodology for exploring a user's acquisition and knowledge of a computer environment. In S.P. Robertson, W. Zachary & J.B. Black (Eds.), *Cognition, computing and cooperation* (pp. 237 - 267). Norwood: Ablex.

Lazonder, A.W., & Van der Meij, H. (1992). *Towards an operational definition of the minimal manual* (Tech. Rep. No. IST-MEMO-92-02). Enschede, University of Twente, Department of Instructional Technology.

Lazonder, A.W., & Van der Meij, H. (1993). *Error-information in tutorial documentation: Supporting users' errors to facilitate initial skill learning.* Paper submitted to International Journal of Human-Computer Studies.

Lazonder, A.W., & Van der Meij, H. (1993). The minimal manual: Is less really more? *International Journal of Man-Machine Studies, 39*, 729 - 752.

Lazonder, A.W., & Van der Meij, H. (1994). Effect of error-information in tutorial documentation. *Interacting with Computers, 6*, 23 - 40.

Mory, E.H. (1992). The use of informational feedback in instruction: implications for future research. *Educational Technology: Research and Development, 40*, 5 - 20.

Van der Meij, H. (1992). A critical assessment of the minimalist approach to documententition. *SIGDOC'92 Conference Proceedings*, 7 - 17.

Van der Meij, H., & Carroll, J.M. (in press). Principles and heuristics for designing minimalist instruction. *Technical Communication.*

CHAPTER 6

The effect of error-information in minimalist documentation[10]

People unavoidably make errors when they learn to use a computer program. The current view on errors is that they can either be helpful or disruptive, depending on the extent to which they are controlled in the learning process. Such error control can be brought about by including error-information in the training manual, supporting the stages users go through when dealing with an error. Good error-information should therefore enable users to detect, diagnose, and correct errors. To investigate the functionality of error-information, two minimal manuals were experimentally compared. One manual contained ample error-information, in the other error-information was entirely absent. Subjects who used the manual containing error-information were expected to perform better during practice as well as after practice. The results bore this out. Error-information resulted in superior corrective skill and did not obstruct the development of constructive skill. In addition, it gave users a more profound knowledge of how the software works. Based on these findings the value of error-information is discussed and ways to extend its potential are suggested.

6.1 Introduction

People's first impression of a computer program is vitally important for initial acceptance, for productivity, and for continued use of that program. Users will persevere with a program for only a limited amount of time. If during this 'honeymoon' period too many problems, failures, and misunderstandings occur, the program is rejected (Booth, 1991).

One of the main reasons first-time users reject software is that it is often confusing and (too) difficult to learn. This negative stance is caused by a mismatch between the program and the user's model of it (Cuff, 1980). Design team members frequently use themselves as models of the user, yet

[10] Lazonder, A.W., & Van der Meij, H. (1993). *Error-information in tutorial documentation: Supporting users' errors in initial skill learning.* Paper submitted for publication.

the gap between designer and user is considerable. As Norman (1986) has pointed out, users and designers have different mental models of the program. They have different goals and perceive the software differently. Prompted by this incongruity, users often misunderstand and misinterpret the software. As a consequence, they make errors and get into trouble. These problems may lead them to abandon their attempts to use the software.

Clearly, software designers are aware of the gap between their expertise and the users' initial ignorance. Iterative user-tests of draft versions are conducted to discover what elements of a program are problematic to users. Some of these problems can be solved by redesigning the program. For example, a new icon can be drawn to replace one that is frequently misunderstood. Other problems can be solved by means of the program's built-in support systems. Help facilities, warnings and error messages are examples of this kind of solution. Yet a third way to accommodate users is by designing tutorial documentation that enables them to master the program without getting in (too much) trouble. This type of support is the focal point in the present chapter.

One might wonder whether any paper support is called for at all. Gilb and Weinberg, quoted in Cuff (1980), stated that documentation should be used as a symptom of poor design, not as a solution to it. Redesigning the software and its programmatic support should overcome all of the users' misunderstandings and errors. However, the present technical know-how is not (yet) that advanced (Baber, 1991). Even with relatively user-friendly programs, such as word-processors, considerably high failure rates have been found to occur. For example, Carroll and Carrithers (1984) found that users spent 25% of their time on dealing with errors. Other studies suggest that this percentage may actually be as high as 50% (e.g., Arnold & Roe, 1987; Card, Moran & Newell, 1983; Graesser & Murray, 1990). Since many of the users' errors are not covered by the programs' error support systems, the task of helping users in error-recovery comes down to the manual.

Until recently, most training manuals did not give users adequate support in dealing with errors. Manuals concentrated on teaching users what they can do with the software and how to do it (i.e., constructive skill), without simultaneously training them to undo the things that have gone wrong (i.e., corrective skill) (Lazonder & Van der Meij, 1994 [chapter 4]). When the development of corrective skills is ignored, errors are likely to have mainly a negative impact on the user. They can, for example, give negative reinforcement which, in turn, can demotivate and frustrate the learner (Glaser, 1965). Errors may also strengthen the (possible) computer anxiety of the novice user and decrease self-confidence ("I am not smart enough to learn to use such a complicated machine"). In addition, when errors cause system-states that are difficult to leave, the user who lacks support may be forced to turn the power off, losing all the work that was produced (Frese & Altmann, 1989).

With adequate support, errors can be quite beneficial to learning. They may help to direct the user to a more appropriate, perhaps even more elaborate understanding of the program (e.g., Booth, 1991; Frese *et al.*, 1988; Pickthorne, 1983). They can help build confidence and skill because users learn to deal effectively with unexpected situations. Moreover, errors can also help stop premature automization of a skill (Frese & Altmann, 1989). For example, when users routinely employ a procedure that is incorrect for the given system state, this forces them to critically review their actions. Errors may also force creative solutions and invite users to explore new strategies (Frese & Altmann, 1989). For example, if a user accidentally activates the typeover mode and makes an error because of this, it might stimulate the user to explore the functions of this mode. Finally, the nature of the learning experience should reflect the intended training outcomes as much as possible. Users should therefore be taught constructive as well as corrective skills (Wendel & Frese, 1987).

In summary, two views on errors can be distinguished. Errors can be disruptive or helpful, depending on the extent to which they are supported during training. Given the fact of first-time users, detailed support is needed in order to enhance the positive effects of errors and minimize the negative ones. Such effective error control is possible by designing manuals that support users' actions with error-information.

6.2 A general model of error-recovery

Error-information in a manual is informative when it supports users' needs. Error-information should thus provide users with all the necessary information to recover an error. But, exactly what information does a user need? The answer to this question can be abstracted from models that describe the stages a user goes through in dealing with an error (Brown, 1983; Curry, 1981; Jelsma & Bijlstra, 1990; Reason, 1990; Wærn, 1991). The stages are: detection, diagnosis, and correction. These stages, which are assumed to be passed through in this particular order, are detailed below.

6.2.1 DETECTION

Users must first discover that they have made an error before they may initiate an attempt to recover it. Error detection is therefore conditional to the other stages in the model. Errors can be triggered in two ways: internally and externally (Allwood, 1984).

In case of internal triggering, a user feels that he or she has done something wrong, but there is no visible cue to confirm this notion (yet). The user may, for example, feel insecure with the selected method, the command(s), or its execution. In thinking-aloud protocols this state is typically signalled by questions like "Did I do this right?" or "I wonder, have I completed all the necessary steps?".

Internal triggering is not a sufficient condition for detection, however. Misconceptions about the appropriateness of a solution method may lead to undetected errors or to a delay in the detection of an error. On the other hand, triggering can also occur if no error has been made (i.e., correct performance is judged as erroneous). So, in addition to internal triggering, the user has to spot the error on the screen to actually detect it. Locating an error occurs by evaluating or reviewing the current system state and the actions that were performed.

Detection can also be prompted by two kinds of external cues: program messages and program states. Program messages often relate to errors or potential errors. For example, messages like "Printer not selected" or "Text not found" signal the presence of a definite mistake. Other messages are less direct, merely prompting users to reflect before committing a possible error. The message "**Delete Block N**o(**Y**es)" nicely illustrates this. The second type of external triggering takes place when the user's actions result in an unex-pected outcome on the screen. For instance, when a user tries to increase the line spacing and finds that it decreases, he or she will probably wonder what went wrong.

Unlike internal triggering, external triggering can only occur when the user perceives the right place on the screen. That is, when he locates the place that shows that an error is made. For the two types of external cues these places differ. Most program messages are displayed at a fixed position, often at the bottom of the screen. Especially for novice users, these messages may go unnoticed because they appear too far from the focal field of attention.

Program states may likewise not prompt immediate detection. Error detection may be delayed when the appropriateness of a solution method does not directly dawn on the user. For instance, a first-time user who misplaces the cursor while marking a block of text often only perceives his mistake after having moved or changed that block. Misconceptions about the expected outcome can even cause errors to remain undetected. For example, selecting 'Small Caps' to change the typeface in a way alters the outlook of the characters. On looking at the screen, the user may infer that his or her actions have lead to the desired outcome.

In short, users often literally do not see that something on the screen is not as it is supposed to be. Occasionally this is because nothing can yet be seen. More often, however, users do not know where to look and what to look for. In these cases error detection may be triggered only at a stage in which error diagnosis and correction have become problematic.

6.2.2 DIAGNOSIS

After detection, users merely know that 'something is wrong'. They may, at that stage, decide the error is not important enough and abandon its pursuit. When a user decides to attend to it, he minimally attempts to establish the exact nature of the error. That is, the user tries to find out 'what is wrong'.

When triggering happens internally, diagnosing the nature of the error is difficult because the user does not yet have a visible cue. Consequently, the user has to compare the program's error-state with his original goal to reveal its nature. Their contrast imparts the discrepancy between the observed and the desired output, which, in turn, indicates 'what is wrong'.

In case of external triggering, information on the nature of the error often is inherent in the program state or message. For example, when a user misspells the name of a file he wants to retrieve, the computer might prompt "ERROR--File CHATPER1.TXT not found". In this example, the user can easily infer that the characters t and p have been reversed. But not all program states or messages (can) provide such detailed diagnostic information. For example, in changing the margins of a document the program state reveals little information on what is wrong. In addition, some system messages merely indicate the presence of an error (e.g., "Runtime error in line 327").

Having identified the nature of the error, users may reason about which actions caused the error (McCoy Carver & Klahr, 1986). This will lead them to know not only what went wrong but also why it went wrong. This last step in diagnosis is hardly ever supported by program messages; most of the time users have to infer what they did wrong on their own account. Such inferencing can be very helpful to a deeper understanding of the program.

6.2.3 CORRECTION

Correction contains four different kinds of user activities: (a) goal setting; (b) selection of the correction method; (c) planning the execution method; and (d) physical execution of the corrective actions.

The user begins with selecting a (repair) goal (Allwood, 1984; Arnold & Roe, 1987; Card *et al.*, 1983). Like constructive performance, corrective actions are always goal directed. The top level goal is obvious: the gap between the present and the desired outcome must be bridged. The user may break down this overall goal into a number of sub-goals (e.g., Anderson, 1985; Frederiksen, 1984; Rasmussen, 1986). For example, the goal 'correct a typo' may be subdivided into the sub-goals 'move the cursor', 'delete the incorrect text', and 'type the correct text'.

Next, the user decides on following a corrective or a (re)constructive method. In a corrective approach users really make changes in the document that correct the error(s). For example, they may remove obstacles that are blocking newly chosen options, or they may undo the error-state. Users can also opt for a (re)constructive method, meaning that they simply try to perform the constructive actions again. By paying more attention they hope to do it right this time.

Which of these two approaches will be employed, depends on the nature of the error. Corrective methods practically always work, they enable users to remedy any type of error. They are, however, also more difficult to learn because they are always situation-specific and tend to require more actions than (re)constructive methods. The latter method has the disadvantage that it does not always work or that it leads to a loss of information. In addition, they can leave behind some information (e.g., uncorrected hidden codes) that can affect later actions or they may lead to a loss of information.

The error-state thus seems to dictate which method is most appropriate. There are, however, many situations in which both approaches work. For example, when a new line spacing code is placed before the old one, the user can correct the error by deleting the old code (i.e., a corrective method), or by moving the cursor to the right place and inserting the new code again (i.e., a (re)constructive method). When both methods are applicable, method selection is handled by selection rules that allow the user to choose between methods (see Card *et al.*, 1983).

Third, the selected method is planned for execution. It is translated into a physical action-sequence. The user selects the commands that will be used and determines in which order they will be executed. The last action in the model is the execution of the commands. These last two steps may be executed simultaneously. That is, users may not plan the whole action sequence in advance but plan and execute step by step.

6.3 Principles for designing error-information

The error-recovery model describes a user's action(s) in dealing with an error. To adequately support these actions, the training manual should incorporate error-information that parallels this recovery process. In designing good error-information several principles should be taken into consideration. These design principles relate to the content and presentation of error-information.

6.3.1 CONTENT

Error-information should explicitly specify when an error has occurred and what must be done to get out of the error state (Lang, Lang & Auld, 1981; Allwood, 1986). Moreover, as novice users often find it difficult to get to know the exact nature and cause(s) of the error (e.g., to understand what the error message means, or to infer the steps that led to the error, see McKendree, 1990), it should also assist the user in diagnosis. Good error-information should therefore contain (a) a characterization of the system-state supporting the detection of the error, (b) conceptual information on the nature and likely cause(s) of the error, and (c) action statements for correcting the error (Lang *et al.*, 1981; Mizokawa & Levin, 1988; Roush, 1992).

This design principle suggests uniformity in how a user's corrective actions should be supported. This is not the case, however. Not every error should be addressed in the same way. Designing error-information that is adaptive to a user's actions therefore requires a more elaborate description of what is meant by an error.

A common classification of error is that into semantic errors, syntactic errors, and slips (sometimes referred to as typing errors) (e.g., Douglas & Moran, 1983; Lewis & Norman, 1986; Norman, 1983; Reason, 1990). Semantic errors are mistakes that occur at the level of the intention. That is, the user's intention to act is not appropriate for achieving his goal. Semantic errors thus occur when an inadequate command is chosen to achieve a given goal. For example, the user may select 'Create Horizontal Line' to try to underline a word. In case of syntactic errors and slips, the user's intention is adequate, but the performance is deficient. When a correct command is carried out improperly, it is called a syntactic error. A typical example of a syntactic error in WordPerfect is ending the search mode by pressing the ENTER key instead of the F2 key. Slips are small errors at the keystroke level (e.g., mistyping the word in the search mode).

These different error types require different handling by the error-information. For example, semantic errors are usually more difficult to detect than syntactic errors or slips (Frese & Altmann, 1989; Lewis & Norman, 1986; Rizzo, Bagnara & Visciola, 1987). In a semantic error, the user's intention is incorrect. When he or she compares the outcome to his goal, they will match. Consequently, there is no internal triggering, and visible contrasting evidence comes only when the selection of a path through the menus has been completed. For syntactic errors and slips, the discrepancy between the intended outcome and the actual outcome often can be observed immediately and easily on the screen (i.e., external triggering). Semantic errors therefore require specific information for detection, whereas a more general description of the error-state is satisfactory for syntactic errors and slips.

Unlike detection, diagnosis is often more complex for syntactic errors and slips because the user cannot suffice by deciding that a wrong method has been used. Instead, he or she must infer which particular command or action was executed incorrectly. The error-state of a syntactic error or a slip often is more open to multiple interpretations, and therefore requires a thorough understanding of the software to make good inferences. For example, when the screen remains empty after an attempt to retrieve a document, the user's first reaction may be to question the method and try again, this time succeeding because he or she does not make the typo that caused the error in his or her first attempt. Syntactic errors and slips thus require more support in diagnosis, especially since a good understanding of what caused an error can prevent the occurrence of this error in the future.

Correction is to a great extent independent of the type of error. Most programs nowadays enable at least two corrective strategies. Users may choose between a generic way to correct an error and a specific one. A typical example of a generic strategy in Apple Macintosh programs is the undo-command that cancels the latest command. In most MS-DOS programs, the ESC-key serves this function. Since generic correction strategies do not work for all errors, specific strategies (e.g., "Press the F7 key and type an N twice. When the screen is empty, retrieve the new document") may sometimes be necessary. Because generic methods can be repeated over and over again, they can be learned and applied more easily. Therefore, generic correction strategies should always be treated before specific ones.

6.3.2 PRESENTATION

Not every single keystroke should be accompanied by error-information. In a recent paper it has been argued that error-information should at least support actions that are error-prone (Lazonder & Van der Meij, 1994 [chapter 4]). Many such situations arise when automaticity leads to an error (Booth, 1991) or when a false analogy is used (Allwood & Eliasson, 1987). In WordPerfect an illustrative example of an automaticity error is that of users who learn to search a string of text. Instead of activating the search-command by pressing the F2 key, they automatically press the ENTER key, the default action to close off a command. An instance of an analogy error occurs when a user substitutes a "1" (the number one) with an "l" (the character l) in typing a filename. Although these characters are interchangeable on a typewriter, they have different meanings in word-processing.

Error-information should also be given when errors are difficult to correct (Van der Meij & Carroll, in press). That is, when the error-state calls for a specific correction strategy that involves many corrective steps. In the example of clearing the screen before retrieving a document, users should be told how to correct the error, for else they may never be able to remedy the situation themselves.

Perhaps the most important problem to novice users is that errors can accumulate, getting them deeper and deeper into an error-state, hence deeper and deeper in trouble. To prevent such accumulation problems, the right timing (i.e., placement) of the error-information in the manual is important. Error-information should be presented on the spot, directly after the actions it refers to (cf. McKendree, 1990; Mory, 1992). Error-information should thus allow for an early detection of errors, which, in turn, prevents errors from piling up, facilitates error-recovery, and minimizes the chance of loosing previous work.

Presenting the error-information 'in context' has the additional benefit that the program's cues can be exploited. By describing the program's (visible) cues in the detection part of the error-information, users receive an exact description of the error state. This facilitates the detection of an error, as users merely have to compare the described error-state to the current system state. Consequently, error-information should be given directly after actions that produce a distinct, observable message on the screen.

Finally, a clear textual and graphical presentation of error-information is indispensable as well. In keeping with the error-recovery model, error-information should be stated in the same detection-diagnosis-correction format throughout. This sequence is always maintained, although small stylistic variations are possible. In addition, error-information has to be signalled to indicate its distinct nature and to facilitate recognition. For this reason we propose to set the error-information in italics, not in the least place because italics tend to be read (a little bit) more slowly (Hartley, 1985), which may be helpful for having users execute the corrective steps

1. *If the text* ***A:\LETTER1.WP*** *does not appear, you have forgotten to clear the screen. Press the F7 key and type an* ***N*** *twice to clear the screen as yet.*	Error type: semantic Problem: detection and correction Solution: the error-state is explicitly specified because the text of the file letter1.wp will appear on the screen anyway. Specific correction information is necessary: this is the only way to correct the error.
2. *If the text* ***Document to be retrieved:*** *does not appear on the screen, you have selected the wrong command. Press the F1 key to rectify your choice.*	Error type: semantic Problem: detection and correction Solution: specific detection information to enable early detection. A generic correction method can be used to correct the error. Note that a specific diagnosis is fairly impossible here.
3. *If you have inserted the text at the wrong place, you have positioned the cursor wrongly before pressing the ENTER key. Remove the text again.*	Error type: syntactic Problem: diagnosis Solution: detection and correction can easily be inferred by looking at the screen. Because the cut and paste function often is elusive to new users, diagnosis of the cause of the error is explicitly specified.
4. *If you cannot entirely select the words "half a million dollars" you did not position the cursor at the beginning of this text before activating the block function. Press the F1 key to undo the block function.*	Error type: syntactic Problem: diagnosis and correction Solution: correctly positioning the cursor is frequently overlooked. Therefore, the cause of the error is explicitly included. Correction is a major problem as well. Unless the block function is switched off, the user cannot proceed. A generic correction method can be applied.
5. *If the text* ***Drive not ready reading drive A*** *appears, you have not inserted the diskette deep enough into the drive. Insert it again so that the button pops up. Then type a* ***1***	Error type: syntactic Problem: detection, diagnosis, correction Solution: detailed information for every error-recovery stage. A specific correction method is the only way out here. Also note that the full-stop at the end of the last sentence was omitted to prevent an accidental typing error.
6. *If the screen remains empty, you have probably made a typing error. Retype the name of the file and press the ENTER key.*	Error type: slip Problem: diagnosis and correction Solution: specific information on diagnosis is included because it is not at all clear what caused the screen to remain empty. A specific correction method is the most efficient way to correct the error.

Figure 6.1
The left column shows six examples of error-information that are extracted from a minimal manual for WordPerfect. The right column characterizes each example by specifying the type of error, the users' main problem(s) in recovering from that error and a rationale for the content of the error-information.

correctly.

Figure 6.1 presents a few examples of how the above design principles can be implemented. At this point, an important limitation should be mentioned. Errors on the task level cannot be anticipated by the error-information. A formula in a spreadsheet application can, for example, be entirely correct but still computes useless values with regard to the content. Similar-ly, the error-information in the manual cannot cover a user's grammar and punc-tuation errors in word-processing.

6.4 Investigating error-information

To our knowledge, the effect of error-information in manuals has hardly been studied. In a previous experiment we compared the learning outcomes of users who were trained either with or without error-information. The results showed that error-information did not improve or decrease the sub-jects' performance during and after training (Lazonder & Van der Meij, 1994 [chapter 4]). Post-hoc analyses revealed some shortcomings that might account for this non-effect. For example, during the experiment (and not in the pilot) a ceiling effect occurred for one of the corrective tests. In addition, most of the subjects' errors were semantic, whereas the error-information mainly supported syntactic errors. Furthermore, detailed analyses of the users' errors led to the principles for the design of error-information described earlier.

After making the necessary changes, another, exploratory study was conducted to find out whether the revised manual might lead to a better assessment (Lazonder, 1994 [chapter 5]). The results of this study indicated that much more errors were supported by the new manual (approximately 40%). Users frequently consulted the error-information to detect and correct errors. They also used the error-information as a means to check if they were still on the right track.

The observational data of this study revealed directions to further improve the manual. Tasks that were prone to errors because of their conceptual complexity to new users were removed (e.g., copying text, using function keys as shortcuts). Tasks and commands that were frequently looked up during practice (e.g., making a block of text, the REVEAL CODES command) were given distinct headings to allow for easy reference. One of the most striking changes with regard to the error-information is that its gradual fading (in presence as well as in content) was removed. As a consequence, both the frequency with which error-information is presented and the extensiveness of the directions to recover from an error is identical for all chapters in the manual. The positioning of error-information was refined as well. In the revised manual, error-information is always preceded by action steps that produce some visible cue on the screen.

This led to the present study in which the effect of error-information on user behavior was examined. In the experiment, half of the subjects were given a training manual with error-information (MM+) while the other half worked with a manual that contained no error-information (MM-). The main hypotheses relate both to the learning activities and the learning outcomes. Overall, MM+ users were expected to require less time to complete practice. This is so because, in the course of practice, they were supposed to commit fewer errors and to recover from their errors faster than MM- subjects. The inclusion of error-information should further be beneficial for the users' scores on performance tests. More specifically, MM+ users should have developed better constructive skills. It was also expected that they would be more knowledgeable and skilled in detecting, diagnosing, and correcting errors.

6.5 Method

6.5.1 SUBJECTS

Fifty adult volunteers (10 men and 40 women) participated in the experiment. They were recruited by means of an advertisement in a local newspaper. The subjects' mean age was 36 (*SD*=10.0). Their educational background varied from secondary education to university. All subjects had less than 100 hours of computer experience[11] and no experience with the experimental software.

The subjects were randomly assigned to one of the two experimental conditions. There were 25 subjects in the MM+ group and 25 subjects in the MM- group. Checks on the random allocation to conditions showed the two experimental groups to be essentially equivalent with regard to age, sex, educational level, intelligence, typing skill, and prior experience with computers.

6.5.2 MATERIALS

Technical equipment

The experiment was performed on an Olivetti 286 personal computer with the menu-driven version of WordPerfect 5.1. WordPerfect's help-function was disabled to restrain subjects from using system cues. A registration program was installed on the computer. It generated a logfile of all of the

[11] this measure conflicts with the definition of novice users presented in chapter 1. However, subjects with more than 50 hours of computer experience worked as data processors or bank employees. As their interaction with the computer merely involved entering data, they were considered novices.

subjects' actions. Whenever a key was pressed, time and keystroke were recorded.

The experimenter used the ERR-system (Error-Recovery Registration System) to log a subject's actions in dealing with errors. This system is completely mouse-controlled and runs on an Apple Macintosh computer under HyperCard. By clicking icons, the experimenter can record when (a) an error is made, (b) an error is detected, and (c) the error-state is ended. When a given icon is clicked, additional information (e.g., type of error, quality of the solution) can be entered for that measure. The clock time of both computers was synchronized to allow for crossreferencing between logfiles. Only measures with satisfactory inter-observer reliability scores (>.70) were used in the experiment. Measures with scores below .70 were rerated by checking the subjects' logfiles.

Manuals

Subjects received a manual (MM+ or MM-) and a training diskette. Both manuals were minimal manuals, designed especially for the experiment and refined on the basis of several pilot tests (see Lazonder, 1994 [chapter 5]). The two manuals differed only with regard to the error-information. In the MM+, error-information was presented frequently (i.e., 45 times). In all, over 35% of the MM+ consisted of error-information, designed according to the heuristics that were detailed above. The different types of error were equally addressed by the error-information. The MM- contained no error-information at all. A more detailed description of the MM+ can be found in Carroll (1990), Lazonder and Van der Meij (1993 [chapter 3]), and Van der Meij and Lazonder (1993).

Tests

Subjects' intelligence was assessed by means of a standardized intelligence test (Raven, 1986). Three tests were used to assess learning outcomes.

A constructive skill test measured subjects' constructive skill. It consisted of 9 items. All items addressed elementary word processing tasks that were trained during practice (e.g., changing the line spacing, underlining words).

Two tests assessed the subjects' capacities in dealing with errors: a knowledge and a skill test. The corrective skill test was performed on the computer. Subjects had to detect and correct 8 errors (5 semantic, 3 syntactic) in a task document. The corrective knowledge test was a paper and pencil test consisting of 9 items. Each item presented a word processing task and a screendump, showing the result of the actions that were performed to accomplish that task. For each item subjects had to mark all possible errors (i.e., detection). In case of an error, they also had to specify its most likely cause (i.e., diagnosis) and a way to correct it.

6.5.3 PROCEDURE

All experimental sessions took place in a quiet room. Each session lasted one day, with a maximum of 8 hours. During the first half hour subjects completed the intelligence test (personal data such as age, sex and computer experience were collected by telephone). The remaining 7.5 hours were spent on word processing. There were short breaks for coffee, lunch and tea.

The subjects were instructed to work through the manual in their own way and at their own pace. They were told that the experimenter would offer help only in case of a computer breakdown or when the subject was stuck for more than 10 minutes. In addition, they were asked to think aloud during practice. Thinking aloud was practiced on a simple task (tying a bowline knot).

Next, subjects were seated at a small desk with the computer and the printer in front of them. They were given their manual (MM+ or MM-) and a diskette, containing all documents to be used in practice. The manual managed the learning process by alternating short explanations with many practical exercises. During practice, the experimenter sat at a table nearby to record the subject's corrective actions, using the ERR-system.

After practice, the subjects were given the constructive skill test and the corrective skill test. A counterbalanced administration was used to control for order effects. After these tests, the subjects completed the corrective knowledge test. During all tests, the subjects were not allowed to consult their manual or to ask for help of the experimenter.

Coding and scoring

During practice, the following measures were scored: time, number and type of errors, the number of detected and corrected errors. Practice time was the time to read the manual and complete the training exercises. With this measure, a distinction was made between time spent on constructive and corrective actions. Errors were scored as semantic, syntactic, or slip. For each error, detection and correction were scored on a true-false scale.

Constructive skill was defined by test time, performance success, and the number of errors. Test time was scored as the time subjects' required to complete the constructive skill test. Again, the difference between time on constructive and corrective actions was made. Performance success was indicated by the number of successfully completed items on this test. For each subject, the number and type of errors in constructive performance was registered as well. All of these scores were assessed by examining the documents on diskette and the subject's logfile.

Corrective skill was defined by three measures: detection, diagnosis, and correction. Detection was scored as either right or wrong. Diagnosis was scored on the following 4-point ordinal scale: (a) both cause and effect are wrong; (b) wrong cause, right effect; (c) wrong effect, right cause; and (d) both cause and effect are right. For corrective knowledge, a similar scale was used. The correction method was scored as one that (a) obviously does not try to correct the error, (b) attempts to correct the error but is semantically and syntactically incorrect or incomplete, (c) is semantically correct but contains one or more syntactic errors, and (d) is both semantically and syntactically correct. The inter-rater reliability scores for all corrective measures were satisfactory (Cohen's Kappa $\geq .80$). Correction on the corrective skill test was defined as the number of adequately corrected errors.

Data analyses

The majority of the data were analyzed by means of (M)ANOVA's, using type of manual (MM+ or MM-) as independent variable. Mann-Whitney U tests were applied to analyze the ordinal data. Where appropriate, the effectsize (*ES*) was computed (in *SD*'s) to establish the magnitude of statistically significant results.

Due to a computer break-down, incomplete scores were registered for three subjects. In addition, one subject (MM-) did not get to the corrective knowledge test and one MM+ subject did not complete the corrective skill test. The data for these subjects were excluded on an analysis-by-analysis basis, causing variable group sizes.

6.6 Results

6.6.1 LEARNING ACTIVITIES

The mean time subjects required to perform constructive and corrective actions during practice is presented in Table 6.1. Overall, the MM+ users were nearly 8 minutes faster than MM- users. This difference was significant, $F(2,46)=3.22$, $p<.05$. Manual type also had a univariate effect on corrective time: MM+ subjects spent a significant 38% less time on dealing with errors ($F(1,47)=4.35$, $p<.05$, $ES=.53$). No effect was found for constructive time ($F(1,47)=.57$). Apparently, subjects from both conditions were equally fast at performing constructive actions.

Table 6.1 also shows the mean number of errors during practice There

Table 6.1
Mean learning activity scores

	Condition	
	MM+	MM-
Time		
Constructive[a]	132.2 (38.6)	124.4 (34.2)
Corrective[b]	27.0 (20.7)**	42.8 (31.0)
Errors		
Semantic	8.4 (5.5)**	12.7 (6.7)
Syntactic	8.8 (5.0)*	11.6 (6.1)
Slip	2.8 (1.8)	2.8 (1.9)

Note. Time in minutes, Standard deviations in parentheses.
[a] Time spent on constructive actions [b] Time spent on corrective actions
* *p*<.10 ** *p*<.05

was a marginal multivariate effect of manual type (*F*(3,46)=2.66, *p*=.06), indicating that, overall, MM+ users tended to make less errors than MM- users. Although there was a trend of the MM+ group having made less syntactic errors (*F*(1,48)=3.26, *p*<.10), a significant univariate effect was found for semantic errors only (*F*(1,48)=6.14, *p*<.05, *ES*=.64).

Table 6.2
Mean error-recovery scores during practice

	Condition	
	MM+	MM-
Detection		
Time	0.4 (0.3)**	0.6 (0.4)
Success[a]	90.1 (9.0)*	85.9 (8.5)
Correction		
Time	0.9 (0.5)**	1.2 (0.5)
Success[b]	86.1 (10.7)***	74.7 (14.2)
Effectiveness[c]	98.0 (3.8)	95.9 (6.6)
Efficiency[d]	2.8 (3.5)	1.7 (1.0)

Note. Time in minutes, Standard deviations in parentheses.
[a] % of detected errors [b] % of successfully corrected errors [c] Mean number of successful corrections to the number of attempted corrections x 100 [d] Mean number of successful corrections per time x 100
* *p*<.10 ** *p*<.05 *** *p*<.01

Table 6.3
Mean scores on the Constructive Skill Test

	Condition	
	MM+	MM-
Time		
Constructive[a]	25.1 (11.8)	28.3 (12.5)
Corrective[b]	8.1 (9.2)	8.1 (7.7)
Performance		
Success[c]	6.2 (1.5)	5.4 (1.8)
Efficiency[d]	30.9 (17.1)	24.6 (20.2)
Errors		
Semantic	5.2 (4.6)	5.8 (4.3)
Syntactic	2.5 (1.7)[*]	3.8 (2.2)
Slip	1.0 (1.1)	1.0 (1.1)

Note. The constructive skill test consisted of 10 items. Time in minutes, Standard deviations in parentheses.
[a] Time spent on constructive actions [b] Time spent on corrective actions [c] Number of items successfully completed [d] Number of items successfully completed per constructive time x 100
[*] $p<.05$

Clearly, subjects in the MM+ group were expected to detect and correct more errors during practice. The ratio of the number of detected errors to the total number of errors is shown in Table 6.2. Whereas MM+ users detected more errors, this difference was not statistically significant ($F(1,48)=2.85$, $p<.10$). They were significantly faster at detecting errors, however ($F(1,48)=4.51$, $p<.05$, $ES=.52$).

MM+ users were also more successful in correcting errors. Manual type significantly affected the number of successful corrections ($F(1,48)=10.28$, $p<.01$, $ES=.80$). It also affected the time for error correction ($F(1,48)=4.90$, $p<.05$, $ES=.60$). Apparently, MM+ users' were faster and better at correcting errors than MM- users. However, the two groups did not differ with regard to the effectiveness of correction (i.e., the number of successful corrections to the number of attempted corrections; $F(1,48)=2.04$). Their efficiency scores (i.e., number of successful corrections per time) were similar as well ($F(1,48)=2.61$).

6.6.2 LEARNING OUTCOMES

Constructive skill

The mean time to complete the constructive skill test is presented in Table 6.3. As with practice time, constructive and corrective activities were analyzed separately. There was no multivariate effect of manual type on time (*F*(2,45)=.63). As can be seen from the mean scores, subjects from both conditions were equally fast on this test.

Table 6.3 also shows the mean performance success scores for constructive skill. Overall, MM+ users produced as many correct solutions as MM- users. An ANOVA on manual type by the number of successfully completed items produced no significant effect (*F*(1,48)=2.51).

Time and performance success were combined into a measure of performance efficiency. As the mean efficiency scores in Table 6.3 indicate, there was no significant difference between the two groups (*F*(1,46)=1.37). Users from both conditions were equally efficient at performing constructive actions.

Finally, the number of errors were compared between groups. As the mean error scores in Table 6.3 indicate, manual type did not affect this measure (*F*(3,44)=1.62). Overall, MM+ users made as many errors as their MM- counterparts. There was, however, a univariate effect of manual type on the number of syntactic errors (*F*(1,46)=4.99, *p*<.05, *ES*=.58).

Corrective skill

Subjects' capacities in error-recovery were assessed by two tests: the corrective knowledge test and the corrective skill test. Their skill in recovering their own errors (i.e., errors on the constructive skill test) was examined as well.

The mean number of detected errors on both tests is shown in Table 6.4. On the corrective knowledge test, manual type had a marginal multivariate main effect on the number of detected errors (*F*(3,45)=2.50, *p*<.10). Overall, MM+ users tended to be better at detecting errors than MM- users. Manual type also had univariate effects on the number of detected semantic errors (*F*(1,47)=4.00, *p*=.05, *ES*=.48) and syntactic errors (*F*(1,47)=6.04, *p*<.05, *ES*=.70). There was no effect on slips (*F*(1,47)=1.36). On the corrective skill test, manual type did not affect the number of detected errors (*F*(2,46)=.90). Apparently, MM+ users detected as many errors as MM- users on this test.

On the corrective knowledge test, subjects had to give a diagnosis for each detected error. The mean rank scores for the quality of diagnoses are presented in Table 6.5. As these scores indicate, the quality of diagnosis differed in favor of the MM+ group. Overall, there was a significant effect

Table 6.4
Mean number of detected errors

	Condition	
Error-type	MM+	MM-
Corrective Knowledge Test[a]		
Semantic	3.9 (0.9)*	3.3 (1.3)
Syntactic	2.1 (0.9)**	1.5 (0.9)
Slip	0.4 (0.5)	0.2 (0.4)
Corrective Skill Test[b]		
Semantic	3.6 (1.3)	3.5 (1.4)
Syntactic	2.6 (0.7)	2.3 (1.1)

Note. Standard deviations in parentheses.
[a] Maximum score = 9 [b] Maximum score = 8
* *p*<.10 ** *p*<.05

of manual type on diagnosis (*Z*=2.02, *p*<.05). Moreover, manual type affected the quality of the diagnoses on syntactic errors (*Z*=2.75, *p*<.01). MM+ users also tended to be better at diagnosing semantic errors (*Z*=1.37, *p*<.10).

Table 6.6 reports the mean scores for error correction. On the written corrective knowledge test, MM+ subjects came up with better correction methods than MM- users. The overall quality of the correction method was significantly higher for the MM+ group (*Z*=1.70, *p*<.05). MM+ users were also better at correcting syntactic errors (*Z*=2.48, *p*<.01). No effect was found for the correction of semantic errors (*Z*=1.02) or slips (*Z*=.65).

Similar findings were obtained on the corrective skill test. Manual type had a multivariate main effect on the number of corrected errors

Table 6.5
Mean scores of the quality of the diagnoses[a]

	Condition	
Error-type	MM+	MM-
Semantic	27.7*	22.2
Syntactic	30.3**	19.5
Slip	25.8	24.2

Note. Diagnoses were registered on the Corrective Knowledge Test only.
[a] Mean rank scores are presented. Higher rank indicates higher quality
* *p*<.10 ** *p*<.01

Table 6.6
Mean scores of the quality of correction

	Condition	
Error-type	MM+	MM-
Corrective Knowlegde Test[a]		
Semantic	27.0	22.9
Syntactic	29.9 **	19.9
Slip	26.0	24.0
Corrective Skill Test[b]		
Semantic	2.2 (1.1)**	1.3 (1.2)
Syntactic	1.8 (1.1)*	1.0 (1.1)

Note. Mean rank scores are presented for correction on the Corrective Knowledge Test, Standard deviations in parentheses.
[a] Maximum score = 9 [b] Maximum score = 8
* $p<.05$ ** $p<.01$

($F(2,46)=4.35$, $p<.05$, $ES=.71$). There were univariate effects on the number of corrected semantic errors ($F(1,47)=8.41$, $p<.01$, $ES=.79$) and syntactic errors ($F(1,47)=5.23$, $p<.05$, $ES=.65$).

MM+ users were not better at recovering errors that occurred during constructive performance (see Table 6.7). Overall, manual type had no effect on the number of detected errors ($F(1,47)=.76$), nor on the number of corrected

Table 6.7
Mean error-recovery scores during constructive performance

	Condition	
	MM+	MM-
Detection[a]		
Semantic	67.6 (27.1)	66.4 (26.9)
Syntactic	76.3 (31.8)	72.3 (32.8)
Slip	92.3 (27.7)	100.0 (0.0)
Correction[b]		
Semantic	86.6 (24.5)	80.5 (28.5)
Syntactic	77.8 (37.1)	91.9 (17.1)
Slip	100.0 (0.0)	100.0 (0.0)

Note. Standard deviations in parentheses.
[a] % of detected errors [b] % of successfully corrected errors

errors (*F*(1,47)=1.68). Contrary to expectations, subjects from both conditions were equally skilled at detecting and correcting their own errors.

6.7 Discussion

People tend to make errors while learning to use software. Among others because these errors cannot be avoided, an approach was proposed that exploits users' errors. That is, the training manual was supplied with error-information that assisted users in dealing with errors. The general hypothesis was that error-information would allow users to develop better constructive and corrective skills.

The outcomes of the study were mixed. Some of the results support the expectations, others do not. In discussing these findings, the effects of error-information on users' constructive skills will be addressed first. Next the way in which subjects recover their own errors is discussed. Then the outcomes for error-recovery on the corrective knowledge and skill tests is considered.

Error-information does not affect the time users spent on constructive skills during practice. The fact that MM+ users were not faster at performing constructive actions may well be explained by the presence of error-information. Since more than 35% of the MM+ consisted of error-information, MM+ users might have needed additional time to work through their manual. Indeed, post-hoc analyses showed a positive correlation between the constructive use of error-information (i.e., consulting error-information in case no error had occurred) and constructive time (r=.41, p<.05). This suggests that the error-information in the manual was processed constructively and hence affected constructive training time.

No positive effect was found on constructive skills after practice either. Subjects from both conditions performed equally fast and equally skilled on the constructive skill test. This non-effect may be explained from the fact that the two manuals presented the same constructive content. Thus, the two groups had received similar constructive training. On the other hand, one could also argue that any positive effect of error-information on constructive skill can emerge only when users are tested later than immediately after practice. A delayed test might therefore be more appropriate to reveal the deeper model and better problem solving skill users in the MM+ condition may have developed (cf. Charney, Reder & Kusbit, 1990).

MM+ users were further expected to be more proficient at recovering their own errors. Some results support this hypothesis. MM+ subjects committed fewer syntactic errors on the constructive skill test, suggesting that the error-information in the manual helps to prevent some errors. Other hypotheses are not confirmed, however. MM+ users successfully detected and corrected as many of their own errors as MM- users.

Several reasons may account for the fact that the MM+ users did not outperform their MM- counterparts. Firstly, WordPerfect's monitoring options might have affected the number of detected errors. The users could, for example, consult the print preview or use the REVEAL CODES command to see if there was an error. We found that most subjects (i.e., 82%) actually did so. Post-hoc analyses indicated that consulting these options and the detection of syntactic errors were positively correlated (r=.31, p<.05). This, in turn, also affected the number of correctly solved items on the constructive skill test (r=.31, p<.05).

Secondly, the program's system cues may have affected this measure accordingly. System cues provide users with valuable information on the occurrence of an error. Cues like "ERROR--FILE CHATPER1.TXT NOT FOUND", "**Exit WordPerfect? N**o (**Y**es)", or "**Block on**" may prompt users to review their goal, their solution method, or the execution of some action step(s). The present study was not designed to examine this effect, but it will be interesting to address this issue in future studies as a means to assess the effect of on-line error-information.

Thirdly, the users' own errors may have been easy to correct. In this respect it is important to recall that the ease or difficulty of correction is independent of the type of error. Instead, it seems to depend on the following factors: (a) whether a generic or specific corrective strategy can be applied; (b) the number of corrective actions required; and (c) whether the user has already made a particular error during practice. The latter factor points at yet another explanation. Because the constructive skill test did not contain transfer items, users were tested on problems that were treated in the manual. In the new situations posed by transfer items, users are likely to face errors they have not dealt with during practice, and hence here the expected effects on error-recovery might emerge.

With a few exceptions, the experiment clearly showed a number of beneficial effects of error-information on subjects' corrective skills. During practice it helped MM+ subjects to make fewer errors and enabled them to detect errors faster. It also resulted in better and faster error-correction. After practice the MM+ group tended to be better at detecting errors on the corrective knowledge test. No effect was found for detection on the corrective skill test. With regard to diagnosis and correction, significant effects in favor of the MM+ group were found on both error-recovery tests. The overall conclusion, therefore, is that error-information does facilitate the development and improves the quality of corrective knowledge and skills. Its benefits, however, mainly relate to diagnosis and correction.

The finding of higher corrective capacities on the corrective knowledge test is probably an important signal that MM+ users have developed a better mental (working) model of the program. The knowledge test is critical in that it is the only test in which users cannot profit from cues from the program. It gives an assessment of unprompted recall; users have to rely on their own knowledge to recover an error. In all other (skills) situations, users can employ system cues for error-recovery. That is, they may receive system warnings or error messages, task progression may be blocked, or they may request information on the screen to reveal possible errors (e.g., they can ask for a print preview or activate the REVEAL CODES command). Post-hoc analyses substantiated this. On the corrective skill test, the use of WordPerfect's monitoring options and error detection were positively correlated (r=.28, p<.05).

What exactly contributes to the development of better error-recovery strategies may not simply be a matter of a better understanding of the program at hand. Surely, it is important for users to learn what to look for on the screen, or what key(s) to press to correct an error. It may, however, be just as important to develop good regulatory processes. Developing superior error-recovery skills thus probably also hinges on teaching users how to monitor their behavior. The presence of error-information may have affected the development of this regulatory skill.

The present study was not designed to address this issue. But there is one finding that suggests that this effect may indeed have occurred. The error-information in the manual always directs the users' attention to the screen to check whether an error has occurred (hence the "if ... then" construction). Observational data from the ERR-system revealed that users frequently consulted the error-information for this purpose. More than 70% of the error-information in the MM+ was used for monitoring. Subjects used it to check the system state with the error-state in the error-information. Taking into consideration that the error-information is not needed for task progression, the high consultancy rate suggests that frequent monitoring is indeed stimulated.

The users' evolving understanding of the program might have important implications for the design of error-information. When the users' knowledge of the program increases, their need for error-information is likely to decrease. That is, users will increasingly employ their own knowledge to recover errors. As a result, error detection, diagnosis, and correction will shift from an external to an internal level. From this, one might conclude that the error-information in a manual should also adapt and, as with constructive information, become increasingly less explicit. However, such a fading technique may not work for error-information because of its optional character. Users who do not make a mistake can skip it without hampering their performance. Thus, some users may really need the full information in the error-information late in the manual. How the content and presentation of error-information can be accommodated to the users' changing needs should be the subject of future research.

More broadly, it is important to expand the advantages of error-information to different user groups. The present study focussed on the unexperienced computer users. Since this audience will become scarce in the near future, it is interesting to speculate whether error-information can be equally beneficial for experienced computer users who simply need to learn another new program. A recent study showed that this might be the case. Ex-perienced users have developed action routines that cause many errors when inconsistent software is used (Prümer, Zapf, Brodbeck & Frese, 1992). In learning a new software package, error-information could thus minimize negative transfer by preventing and supporting analogy errors.

Yet another audience that might benefit from error-information is that of casual users. This user group works with the software so infrequently that they do not go through the regular training stages. In fact, they are not interested in learning the program at all; they just want to use it to achieve a personal goal (Brockmann, 1990). Due to their preferences, casual users have a high propensity for error which, as with first-time users, can easily lead them to abandon the program. Consequently, they have a need for a safety net that guides them in their constructive actions and helps them to get out of problems (Cuff, 1980). As was previously mentioned, the software cannot take care of this need alone; error-information could support casual users in achieving their goals.

Another interesting issue concerns the applicability of the error design principles for on-line documentation. Most principles regarding the content and presentation of error-information can probably be easily applied. Only the guidelines regarding its physical presentation need to be adjusted to the new situation. In addition, due to the interactive nature of on-line documentation, new possibilities for supporting users' errors arise. For example, some types of error-information need not necessarily be imposed on the user. Rather, the program could present it only when users request it to check if an error has occurred. Thus, the program could assist users in monitoring their behavior and support error detection (e.g., Bradford, 1990; Dayton, Gettys & Unrein, 1989). In short, our principles for designing error-information seem fit for designing on-line documentation. The effect of error-information in on-line documentation on user behavior will have to be investigated in the future.

References

Allwood, C.M. (1984). Error detection processes in statistical problem solving. *Cognitive Science*, *8*, 413 - 437.

Allwood, C.M. (1986). Novices on the computer: A review of the literature. *International Journal of Man-Machine Studies, 25*, 633 - 658.

Allwood, C.M. & Eliasson, M. (1987). Analogy and other sources of difficulty in novices' very first text-editing. *International Journal of Man-Machine Studies, 27,* 1 - 22.

Anderson, J.R. (1985). *Cognitive psychology and its implications*. San Francisco: Freeman.

Arnold, B. & Roe, R. (1987). User-errors in human-computer interaction. In M. Frese, E. Ulrich & W. Dzida (Eds.), *Psychological issues of human computer interaction in the workplace* (pp. 203 - 220). Amsterdam: Elsevier.

Baber, R.L. (1991). *Error-free software: Know-how and know-why of program correctness.* Chichester: Wiley.

Booth, P.A. (1991). Errors and theory in human-computer interaction. *Acta Psychologica, 78,* 69 - 96.

Bradford. J.H. (1990). Semantic strings: A new technique for detecting and correcting user errors. *International Journal of Man-Machine Studies, 33,* 399 - 407.

Brockmann, R.J. (1990). *Writing better computer user documentation: From paper to hypertext.* New York: Wiley.

Brown, J.S. (1983). Learning by doing revisited for electronic learning environments. In M. A. White (Ed.), *The future of electronic learning* (pp. 13 - 33). Hillsdale: Erlbaum.

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

Carroll, J.M. (1990). *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill*. Cambridge: MIT.

Carroll, J.M. & Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors, 26*, 377 - 389.

Charney, D., Reder, L. & Kusbit, G.W. (1990). Goal setting and procedure selection in acquiring computer skills: A comparison of tutorials, problem solving and learner exploration. *Cognition and Instruction, 7*, 323 - 342.

Cuff, R.N. (1980). On casual users. *International Journal of Man-Machine Studies, 12*, 163 - 187.

Curry, R.E. (1981). A model of human fault detection for complex dynamic processes. In J. Rasmussen & W.B. Rouse (Eds.), *Human detection and diagnosis of system failures* (pp. 171 - 184). New York: Plenum Press.

Dayton, T., Gettys, C.F. & Unrein, J.T. (1989). Theoretical training and problem detection in a computerized database retrieval task. *International Journal of Man-Machine Studies, 30*, 619 - 637.

Douglas, S.A. & Moran, T.P. (1983). Learning text editor semantics by analogy. In A. Janda (Ed.), *Human factors in computing systems: Proceedings of the CHI'83 conference* (pp. 207 - 211). Amsterdam: Elsevier.

Frederiksen, N. (1984). Implications of cognitive theory for instruction in problem solving. *Review of Educational Research, 54*, 363 - 407.

Frese, M., Albrecht, K., Altmann, A., Lang, J., Von Papstein, P., Peyerl, R., Prümper, J., Schulte-Göcking, H., Wankmüller, I. & Wendel, R. (1988). The effect of an active development of the mental model in the training process: Experimental results in a word processing system. *Behaviour and Information Technology, 7*, 295-304.

Frese, M. & Altmann, A. (1989). The treatment of errors in learning and training. In L. Bainbridge & S. A. Ruiz Quintanilla (Eds.), *Developing skills with information technology* (pp. 65 - 86). Chichester, Wiley.

Glaser, R. (1965). Toward a behavioral science base for instructional design. In R. Glaser (Ed.), *Teaching machines and programmed learning,* vol. 2 (pp. 771 - 809). Washington: AECT.

Graesser, A.C. & Murray, K. (1990). A question-answering methodology for exploring a user's acquisition and knowledge of a computer environment. In S. P. Robertson, W. Zachary & J. B. Black (Eds.), *Cognition, computing and cooperation* (pp. 237 - 267). Norwood: Ablex.

Hartley, J. (1985). *Designing instructional text.* London: Kogan Page.

Jelsma, O. & Bijlstra, J.P. (1990). Process: Program for research on operator control in an experimental simulated setting. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-20*, 1221 - 1228.

Lang, T., Lang, K. & Auld, R. (1981). A longitudinal study of computer-user behavior in a batch environment. *International Journal of Man-Machine Studies, 14*, 251 - 268.

Lazonder, A.W. (1994). Minimalist documentation and the effective control of errors. In M. Steehouder, C. Jansen, P. van der Poort & R. Verheijen (Eds.), *Quality of technical documentation* (pp. 85 - 98). Amsterdam: Rodopi.

Lazonder, A.W. & Van der Meij, H. (1993). The minimal manual: Is less really more? *International Journal of Man-Machine Studies, 39*, 729 - 752.

Lazonder, A.W. & Van der Meij, H. (1994). Effect of error-information in tutorial documentation. *Interacting with Computers, 6*, 23 - 40.

Lewis, C. & Norman, D.A. (1986). Designing for error. In D. A. Norman & S. W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction* (pp. 411 - 432). Hillsdale: Erlbaum.

McCoy Carver, S., & Klahr, D. (1986). Assessing children's LOGO debugging skills with a formal model. *Journal of Educational Computing Research, 2*, 487 - 525.

McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer Interaction, 5*, 381 - 413.

Mizokawa, D.T. & Levin, J. (1988). Standards for error messages in educational software. *Educational Technology, 28*, 19 - 24.

Mory, E.H. (1992). The use of informational feedback in instruction: Implications for future research. *Educational Technology: Research and Development, 40*, 5 - 20.

Norman, D.A. (1983). Design rules based on analyses of human error. *Communications of the ACM, 26*, 254 - 258.

Norman, D.A. (1986). Cognitive engineering. In D.A. Norman & S.W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction* (pp.

31 - 61). Hillsdale: Erlbaum.

Pickthorne, B. (1983). Error factors: A missing link between cognitive science and classroom practice. *Instructional Science, 11*, 281 - 312.

Prümer, J., Zapf, D., Brodbeck, F.C. & Frese, M. (1992). Some surprising differences between novice and expert errors in computerized office work. *Behaviour and Information Technology, 11*, 319 - 328.

Rasmussen, J. (1986). *Information processing and human-machine interaction: An approach to cognitive engineering*. New York: Elsevier.

Raven, J.C. (Ed.), (1986). *Standard progressive matrices and vocabulary scales*. London: Lewis

Reason, J. (1990). *Human error*. Cambridge: Cambridge University Press.

Rizzo, A., Bagnara, S. & Visciola M. (1987). Human error detection processes. *International Journal of Man-Machine Studies, 27*, 555 - 570.

Roush, R. (1992). Taking the error out of explaining error messages. *Technical Communication, 39*, 56 - 59.

Van der Meij, H. & Carroll, J.M. (in press). Principles and heuristics for designing minimalist instruction. *Technical Communication.*

Van der Meij, H. & Lazonder, A.W. (1993). An assessment of the minimalist approach to computer user documentation. *Interacting with Computers, 5*, 355 - 370.

Wærn, Y. (1991). On the microstructure of learning a wordprocessor. *Acta Psychologica, 78*, 287 - 304.

Wendel, R. & Frese, M. (1987). Developing exploratory strategies in training: The general approach and a specific example for manual use. In H.J. Bullinger, B. Schackel & K. Kornwachs (Eds.), *Proceedings of the second IFIP conference on human-computer interaction* (pp. 943 - 948). Amsterdam: Elsevier.

CHAPTER 7
General discussion

7.1 Introduction

There is something paradoxically about looking back on a research project like the one described. As Carroll (1993) pointed out: "It is unavoidably unfair to look back in judgement on earlier work; what is obvious today is obvious *because* earlier work advanced and clarified matters. Yet we must look back if we want to understand where a field has been and where it might be going." (p. 4).

Prompted by this plea for reflection, this chapter reviews the research project that was described in this thesis. As detailed discussions were included at the end of each chapter, the present discussion is rather brief. Its main objective is to give a broader perspective to the work that was presented. The discussion therefore focuses on how the results of the experiments compare to the work in other, related areas.

The chapter starts with a discussion on the functionality of the minimalist approach. In chapter 2 this approach was introduced as a coherent set of design principles. In chapter 3 a minimal manual designed according to these principles proved to lead to significantly higher learning outcomes when compared to a state-of-the-art self-study manual. The minimalist approach was therefore considered to be effective for teaching basic computer skills. In section 2 of this chapter the generalizability of this conclusion is viewed from a broader perspective. Among others, the limitations of the minimalist approach to first-time user documentation are discussed.

The experiments in chapter 4 to 6 were designed to uncover whether even a single minimalist principle might affect learning and performance. The experiment discussed in chapter 6 presented the most salient results. This study showed that the inclusion of error-information resulted in superior corrective knowledge and skills of users, without obstructing their constructive skills development. Alternative approaches to error-based learning are presented and discussed in section 3.

The chapter closes with a discussion on manual design in general and minimal manual design in particular. So far, this thesis has paid little attention to the way in which minimalist instruction is designed. Although several principles and heuristics for design were identified, the overall methodology for designing minimalist instruction has largely been passed over. In the field of instructional design, this issue has gained importance, especially since researchers have discovered a discrepancy between how design is represented in prescriptive models and how it is actually carried out in practice (e.g., Pieters & Bergman, 1993; Tripp & Bichelmeyer, 1990). Based on these new insights and on the experiences from designing the WordPerfect manuals, the processes involved in designing minimalist instruction were identified. They are discussed in section 4.

7.2 Investigating minimalist tutorials

In chapter 2 the minimalist approach was introduced as a new design theory for designing tutorial documentation. The experiment reported in chapter 3 bore this out. Similar to Carroll's original study (Carroll, Smith-Kerker, Ford, Mazur-Rimetz, 1987), significant and considerable gains of a minimal manual over a state-of-the art self-study manual were found. Prompted by these findings, the use of a minimal manual was considered a preferred solution to the problem of teaching basic computer skills to novice users. In this section, some questions concerning the generalizability of this conclusion are addressed.

Firstly, the subjects in the experiment presented in chapter 3 were university students, a very homogeneous group of highly educated, intelligent young adults. Clearly, these students are not a representative sample of the population of adult computer novices. As a result, one may wonder whether the findings obtained in the experiment described here also apply to the broader population of adult users.

There is some evidence that this indeed is the case. There is, first, the original study of Carroll *et al.* (1987) in which the subjects were secretaries. In addition, Carroll mentions positive effects with various audiences in his book *The Nürnberg Funnel* (Carroll, 1990). Moreover, in a study using exactly the same manuals as in the experiment in chapter 3, nearly the same considerable, positive results were found for a substantially more heterogeneous sample of subjects (Van der Meij & Lazonder, 1993; see also Van der Meij, 1992). In that study the subjects were adult volunteers aged between 17 and 63 with a highly variable educational background; the sample included two subjects with a university degree but also a number of subjects that had not completed secondary school.

The second question relates to the software. The experiments in this thesis used the menu-driven version of WordPerfect 5.1. This choice for a word processor was prompted by the fact that the purpose of the first study was to replicate Carroll's study. The choice for WordPerfect was inspired by the high popularity of this package in the Netherlands. Using the menu-driven version of WordPerfect helped expel the skepticism regarding the functionality of the minimalist approach for screen-based programs. What remains in question is whether minimal manuals are equally beneficial for users who must learn to use more complex computer programs or computer equipment.

A review of the literature indicates that, among others, minimalist tutorials have been developed for learning to use a faxmodem (Scholtz & Hansen, 1933), HyperCard (Anderson, Knussen & Kibby, 1993), Smalltalk (Rosson, Carroll & Bellamy, 1990), a computer-aided design (CAD) system (Vanderlinden, Cocklin & McKita, 1988), an interface construction toolkit (Vanasse, 1994), and a safety application for predicting physical stress (Gong & Elkerton, 1990). Nearly all of these studies report considerable advances of the minimal manual over a state-of-the-art self-study manual. In short, they support the notion that the minimalist approach can be effective in domains other than word processing.

On the other hand, these studies do not address the question how minimalism compares to other instructional methods for teaching (basic) computer tasks. Nor does the experiment described in chapter 3. All of these studies contrasted a minimal manual with a state-of-the-art tutorial. The minimalist approach was not compared to other paper instructions such as the Leittext method (Teurlings, 1993) or the information mapping approach (Horn, 1989, see also Steehouder, 1990). Moreover, as the medium of instruction tends to shift to computerized support, a comparison of a minimal manual with, for example, an on-line tutorial might be informative.

In addition to expanding the minimalist approach, it is equally important to come to an understanding of *why* the minimalist approach is successful. Carroll has introduced minimalism as a set of design principles and heuristics that were shown to be effective when used in combination. The question is whether these principles will also work in isolation. Does each principle have a unique contribution to the manual's effect? Or is it just a fine-tuning of all principles and heuristics together? If the latter is true, this will severely limit the usefulness of minimalism, as one must opt for using all principles and heuristics. Clearly, the idea is that the four major principles can have a unique effect. There is also some research to support this.

For example, Black, Carroll and McGuigan (1987) investigated the effect of 'slashing the verbiage', a feature previously classified as one of the design principles that facilitates text optimization (see Table 2.1). Their study revealed a positive correlation between manual length and time. Subjects who were given less to read were significantly faster during practice and needed significantly less time to complete test exercises.

Another study in which distinct minimalist principles were examined was conducted by Gong and Elkerton (1990) who manipulated task orientation, 'slashing the verbiage', and support of error-recovery. Contrary to Black *et al.* (1987), they found no effect of shortening the manual on practice time. However, it did cause a significant decrease of the number of errors during practice. Their study further revealed that the manual's task-oriented nature affected practice time, whereas error-information significantly reduced both the number of errors during practice and the time to complete transfer tasks.

The experiments reported in this thesis further support the idea that the inclusion of error-information significantly enhances the effects of the minimal manual. Strictly speaking, it then remains to be shown whether error-information might also help improve a standard self-study manual, but this is a moot issue in view of the proven superiority of the minimal manual. The overall impression from the studies on minimalism thus supports the idea that each minimalist principle contributes to the manual's positive overall effect on learning and performance. It is therefore probably correct to conclude that the effect of these principles is not merely a synergistic one.

7.3 Error-based learning

The key assumption of the work presented in the chapters 4 to 6 was that errors can help people learn to operate a computer program if these errors are controlled in the learning process. The demands for such effective error control were identified in chapter 4, whereas the practical conditions for learning from errors were described in chapter 5. This was substantiated in the experiment presented in chapter 6. This study showed that subjects who used a manual that contained error-information required significantly less time to complete practice because they made fewer errors and were faster and better at recovering errors. After practice they were better at diagnosing and correcting errors.

In this section, other approaches to error-based learning are considered as alternatives to the approach presented in the experiments. These approaches can roughly be classified as instruction-driven or software-driven. Instances of both modes are discussed in view of their efficacy in controlling errors and their capacity to allow people to learn from errors.

A typical example of an instruction-driven approach is Frese's error management (Frese & Altmann, 1989; Frese *et al.*, 1991). Error management departs from the idea that users should learn when errors are likely to occur and how to deal with them effectively. According to Frese, this can be accomplished by including an explicit error training in the manual. Such error training involves presenting some kind of error and asking the learner to recover it. Error training is best given in the middle of the instruction; when offered at the beginning of the learning process it is assumed to cause information overload.

Although the error management method seems rather straightforward, error training can take many different forms. Frese described the possibility of the following strategies: (a) explicitly describe potential errors and their correction methods in the manual; (b) ask one learner to get out of the errors of a second learner; (c) reproduce error states on paper and ask learners to describe how the error has come about and how it can be corrected; (d) let the learners perform extremely complex tasks, and ask them to try to get out of the errors that are bound to appear.

Another instruction-driven approach to error-based learning can be found in the Leittext method. Leittext is an individualized training method that was first developed in the field of technical training in Germany. Teurlings (1993) successfully applied this method in learning to use a word processor. The basic idea behind the Leittext method is that learners individually perform a realistic task or assignment (e.g., typing out a text, styling a text, creating a text) under support of didactic aids. For example, learners may be given regulation questions, checklists, and technical instructions that structure the learning process according to the following six phases: informing, planning, deciding, executing, checking, and evaluating.

Learning from errors takes place in the last two phases. Firstly, learners are prompted to self-check their performance. They must consider their learning outcomes and reflect on the method they have applied. This self-check is assumed to enable learners to detect errors and identify their possible cause(s). Secondly, the trainer evaluates the learning process and learning outcomes. Among others, the trainer discusses how the errors can be prevented in future.

Although the error management and Leittext method pay considerable attention to corrective skills development, there is at least one major draw-back: they do not control users' errors during learning. Frese's error management approach is similar to one in which a distinct chapter in the manual is devoted to dealing with errors. This basically comes down to considering error-recovery as a distinct task users have to learn, similar to, for example, retrieving a document or changing the typeface. In the Leittext method, errors are considered at the end of the learning process. Users are not supported in detecting, diagnosing, and correcting errors during task execution. Learning from errors is therefore mainly directed at preventing errors in future task execution.

In contrast, software-driven approaches to error-based learning typically focus on error control by blocking the users' errors during (the first few hours of) learning. A typical example of this kind of error prevention can be found in training-wheels technology (Carroll & Carrithers, 1984; Carroll & Kay, 1985). In a training-wheels system, advanced functions of a program are disabled, and so are some of the options where mistakes may severely hamper task continuity. When users do select disabled commands, a message informs them that that particular command is unavailable and task execution can continue without any corrective action.

Carroll and Carrithers (1984) experimentally compared a training-wheels system to a complete system. Overall, they found that subjects who used the training-wheels system were faster and more successful. Training-wheels subjects needed less time to complete training, made significantly fewer errors, and were significantly faster at error-recovery. Despite these positive findings, one should keep in mind that the training-wheels system was not designed to reduce the number of errors per se. In fact, users could still make an excessive amount of errors. The main advantage of the system is that it extenuates the consequences of (some) errors, which, in turn, makes it more trackable for users to detect and correct the 'unblocked' errors.

A more advanced version of a training-wheels system is reported in Biemans and Simons (1992). In their system, a concurrent instructional shell was built around a word processor. This shell monitored subjects' performance during learning and provided feedback. In case of a correct solution, the subjects actions were carried out and the message "OK" appeared. When the subjects' input was incorrect, their actions were blocked and they were given feedback. Unfortunately, this training-wheels system was used to study the effect of self-regulation activities. Consequently, no information was gathered with regard to its capacity to control errors or to allow for error-based learning (Biemans, personal communication).

While these training-wheels systems control errors during learning, and probably to a higher degree than is accomplished with error-information in a manual, they seem limited in their capacity to exploit errors. By blocking errors and their consequences, users cannot fully capitalize on their effects and learning from them is restricted. Little attention is given to corrective skills development, a shortcoming that is likely to reveal itself after practice. When users make an error that was blocked during practice, they are not trained at either recognizing the error-state (i.e., detecting the error) or getting out of it (i.e., correcting the error).

In short, the instruction-driven and the software-driven approach mainly support one aspect involved in error-based learning. The instruction-driven approach facilitates corrective skills development but fails to control errors during learning, whereas the software-driven approach controls errors but offers users no opportunity to learn from them. Therefore, both approaches would complement each other well, at least in theory. In practice, their combination will inevitably lead to a deadlock.

It is interesting to note that one of the ways in which the two approaches are 'balanced' occurs when manuals contain both safety-information and error-information. The safety-information (i.e., dangers, warnings and cautions) resembles a training-wheels system in that it intends to prevent users to make errors. As such it seeks to control errors. Like in the training-wheels technology, safety-information is presented only in situations where errors are costly. That is, when errors involve a risk to the product or the person (e.g., Klauke, 1994; Venema, 1990).

7.4 Designing minimalist instruction[12]

Over the past decade, the number of handbooks on manual design has grown considerably (e.g., Brockmann, 1990; Cohen & Cunningham, 1984; Grimm, 1987; Hendrix & Van der Spek, 1993; Weiss, 1991). In each of these books, the development process is described on the basis of a systematic design model. Roughly speaking, these models contain five main activities, each of which is to be carried out after completing the previous one. These five activities are: (a) analyze the context, audience, and content; b) design the manual; (c) develop the manual; (d) evaluate and revise the manual; and (e) implement the revised manual.

Such models represent the activities involved in developing tutorial documentation in a fixed, linear order. This is not to say, however, that such a linear portrayal of steps and activities is typical of the way in which manuals are actually designed. In fact, it is generally acknowledged that (most) design efforts progress in repeated iterative cycles rather than in one single sweep carried out in a linear fashion (e.g., Banathy, 1987; Gayeski, 1991*b*; Pieters, 1992; Rowland, 1992). It might therefore be appropriate to represent the design process in more detail, that is, by taking its cyclic nature into account.

By prescribing explicit guidelines for each of these phases, linear models further suggest that design merely comes down to applying well-tried

[12] to appear in Van der Meij, H., & Lazonder, A.W. (in press). Het ontwerpen van ‘minimal manuals’ [Designing minimal manuals]. *Tijdschrift voor Taalbeheersing, 16*(3)

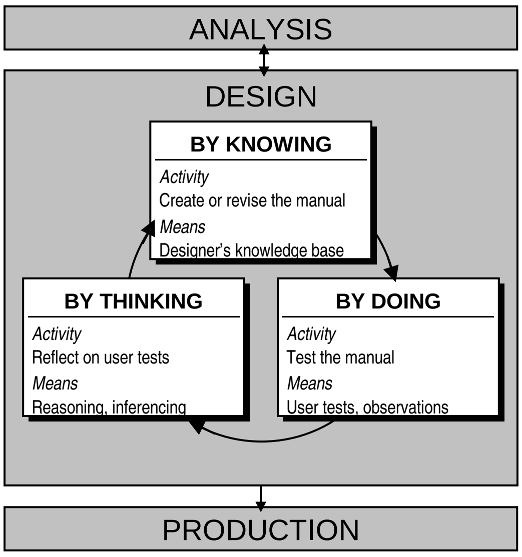


Figure 7.1
Minimalist design process

solutions. In this respect, the representation of the design process requires some differentiation as well. In addition to performing 'trusty' techniques like task analysis, needs assessment, and stating the objectives, design always involves a certain level of creativity. That is, it involves a constant balancing between approved ideas and new, original input or expressions that arise from the need to optimally adjust the manual to the features of the program, the task domain, and the target population.

In short, there seems to be a discrepancy between how design is represented in models and how it is carried out in practice. In describing the process of designing minimal manuals, an attempt is made to provide a more genuine description of the design process (henceforth referred to as the minimalist design process). The core activities of this process can be summarized as shown in Figure 7.1. As this figure indicates, the steps involved in the minimalist design process progress in iterative, recurrent cycles that are especially noticeable in the design phase. They are detailed below.

7.4.1 ANALYSIS

The first step in designing a minimal manual involves analyzing the components of the instructional setting. In this respect, the minimalist design process progresses in line with linear design models: it starts with an analysis of the software, of the relevant knowledge, skills, and idiosyncratic features of the intended audience and of the context, leading to a tentative statement of objectives. There are, however, two striking differences that relate to the way in which these analyses are carried out.

The first difference is that the designers focus on how the user interacts with the program, going slightly beyond a task-oriented approach and certainly way beyond a software-oriented analysis. In most systematic design models the nature of the learning outcomes tends to dictate the design process. In designing minimal manuals, the instruction is 'written around' the user. This act of user centeredness causes a shift in the central activity of the analysis phase from a pure subject-matter analysis to an analysis of how the target audience relates to the software. As a consequence, designers cannot suffice by giving a mere description of the users' demographic features. Instead, they should look for detailed information or make assumptions on the users' information needs, learning preferences, computer skills, and domain knowledge in relation to the program at hand.

The second difference relates to the analysis of the software. In the minimalist design process, task analysis is more comprehensive. As in linear design models, a fine-grained, top-down analysis of the actions required to operate the program is performed to identify the constructive skills the learner should get to know. But, given the fact that minimalist instruction supports error-recovery, this analysis is complemented with one of corrective skills. That is, the most prevalent errors are identified and so are the actions to recover them.

7.4.2 DESIGN

During design, three steps are carried out iteratively. In that respect, minimalist design resembles a recent design methodology called rapid prototyping (e.g., Gayeski, 1991*a*; Tripp & Bichelmeyer, 1990). According to this methodology, research and development should be conducted as concurrent processes, leading to prototypes, which are then tested, and which may or may not evolve into a final product. The rationale for this approach is that full understanding of needs, content, and objectives tends more often to be a result of the design process rather than an input to it.

Its flexible nature makes rapid prototyping appropriate to describe the minimalist design process. As users' actions can never be fully anticipated, the designer repeatedly alternates constructive acts of design with practical user tests. In performing these tests, a salient difference between rapid prototyping and the minimimalist design process appears. In rapid prototyping – as well as in linear design models – user tests are conducted with a protoype of the instruction. In the minimalist design process, the 'minimal' size of the manual allows for users tests with a complete version of the manual.

These iterative cycles of the minimalist design process are illustrated below. In using the same labels that were applied to classify the activities of users, the prevailing activities of designers can be characterized as design by knowing, by doing, and by thinking.

Designing by knowing

Each iterative design cycle starts with *designing by knowing*. Expert designers often use their rich body of knowledge as their starting point for design (Banathy, 1987; Gayeski, 1991*b*; Rowland, 1992; Schön, 1983). They integrate their general knowledge (i.e., notions from scientific disciplines like psychology, learning theory, or technical writing) and specific knowledge (i.e., information from the analysis phase) to construct a first draft of the manual.

An instance of designing by knowing can be found in the styling of the different types of information in a manual (see section 2.4.4). The designer knows that a minimal manual should contain at least four different types of information: action information, background information, error-information, and linkage information. The designer also knows that different information types should be presented differently. Graphically speaking, there are numerous options to differ between information types. However, reading research shows that italics are read slightly more slowly than a roman typeface (e.g. Hartley, 1985). Other studies indicate that readers perceive numbers as a representation of a sequence (e.g., Feinberg, 1989). In designing by knowing, these facts are integrated: the designer decides to number the action information and to put the error-information in italics.

Another example of designing by knowing relates to how the position of keys on the keyboard should be explained. From experience (or from observing users during the analysis phase) the designer knows that novices often have some trouble identifying and locating special keys, such as F1, ENTER, DELETE, and BACKSPACE. Therefore, the manual has to support users in finding these keys. Given the fact that the WordPerfect manual was designed for internal use only (i.e., for one type of keyboard), the position of keys could easily be identified by highlighting the relevant keys on the keyboard through illustrations in the manual. Following from the design principle to encourage exploration and problem solving (see section 2.3.1), displaying only the 'looks' of each key and omitting information about its location was considered preferable because it would encourage users to actively search for that key. This, in turn would help them to more fully understand the structure of the keyboard.

Designing by doing

The second step in the design phase is *designing by doing*. In designing by doing, provisional versions of the manual are subjected to pilot tests during which the designer records the users' actions. These user tests reveal the strengths and weaknesses of the manual and suggest points for revision. Designing by doing is a crucial step in the design process because what once worked well in one situation may very well be dysfunctional under different circumstances. Moreover, users tend to show unanticipated behaviors, which, if not adapted to, can turn an otherwise well-written manual into an unfit or incomprehensible set of directions.

Designing by doing is, among others, necessary to achieve text optimization. Comprehensible text cannot be designed without extensive pilot testing. With some basic knowledge of the target audience, the designer can start with weeding out all the excess and replacing jargon (when rewriting a manual) or with writing a simple first draft (when designing from scratch). After that, user tests are needed to find the right balance between presenting (additional) information users need to work with the program and deleting information the users already know or can infer.

The need for designing by doing is also revealed by the provision of error-information. Error-information should be given when errors block task progression. That is, when actions are error-prone or when errors are difficult to correct. Experienced designers can anticipate when such situations arise (Van der Meij & Carroll, in press). Notwithstanding their knack to locate the 'hot spots' in a manual, user tests remain indispensable. Observing users and tracking their errors helps better assess their needs for support in dealing with errors. More specifically, the designer can better identify the right content and presentation of error-information. For example, in the first experiment on error-information, the need for repeated, full presentation of error-information was underestimated. User tests revealed this shortcoming, and so, in a subsequent manual error-information was always given when mistakes were expected, regardless of the users' possible prior knowledge for correction (i.e., no gradual fading).

Designing by thinking

Designing minimalist instruction comes down to more than just knowing or doing things. It also involves *designing by thinking*. Practitioners reason about their design, they reflect and make inferences and generalizations (cf.

Schön, 1983). Such activities are indispensable in view of the above-mentioned fact that designing often implies more than rigidly following ready-to-apply prescriptions. As environments, programs and audiences vary, there will always be a need for alternative ways to implement a design principle, to incorporate new ideas, or to *not* follow some of the heuristics (Van der Meij & Carroll, in press).

A typical example of designing by thinking appears when designers interpret the outcomes of user tests. Test outcomes only suggest local optimizations of the manual. That is, they 'merely' show where additional error-information is needed, what metaphors are dysfunctional, or which chapters are too long to work through. Only reflection or thought can reveal the underlying principles behind these problems, making it possible to transfer their solution to other parts of the manual. For example, if tests indicate that most users end the search mode by pressing the ENTER key, the designer may infer that this is due to the fact that users have automized this routine. The designer may then infer that additional support may be needed *every time* a command is ended unconventionally, even though the tested users did not show these problems.

Mindlessly applying known design principles can also be dysfunctional. Design is creative, abidingly seeking ways to optimally adjust the instruction to the users' knowledge, skills, needs, and preferences. Thus, it constantly forces designers to develop new ideas or to reconsider certain design principles. This is nicely illustrated in a recent design project in which a minimal manual for an advanced laser robot was to be created (Laret, 1993). Given the inherent dangers of inviting users to explore some options of the laser robot, users were not stimulated to strike out on their own. The risk of working with a laser beam also prompted the designer to incorporate another type of information that is uncommon in most software manuals, namely warnings.

After the third step, some additional analyses may take place, the design cycle may start again, or production may begin. When a new cycle starts, the insights that were revealed in the previous design cycle(s) are part of the specific knowledge base of the designer. After designing the next version of the manual, user tests are conducted and the outcomes of these tests are reflected upon. The iterative design cycles end when user tests signal that the quality of the manual is satisfactory. In that case, the final phase of the design process starts: the production of the manual.

7.4.3 PRODUCTION

With commercially developed documentation, the actual production of the manual is preceded by a series of 'pre-press' activities like proofreading, finishing the rough illustrations, and preparing the final DTP-version of the manual. As these activities were not performed as part of the present research project, they will not be discussed here. An elaborate description of the production phase can be found in Brockmann (1990), Pakin (1984), Sullivan (1988), and Wright (1988).

7.5 Epilogue

In working on this research project, I sometimes wondered whether scientific findings are actually used in practice. This question seemed relevant, espe-cially since research into computer user documentation is considered to be applied research. In addition to increasing scientific understanding and gratifying the researcher's personal curiosity, its goal is to provide designers with empirically verified principles and heuristics on creating better manuals.

A glance at the past proves that this concern is not entirely unfounded. Document design has its origins in the 1930s. Since then, thousands of pages have been written on manual design. Practitioners admit to agree on these notions, but, ironically, incomprehensible manuals have long been the rule rather than the exception.

Fortunately, things have started to change. As Schriver (1989) pointed out, manual design is yet an emerging discipline and much of its development in theory, research, and practice has occurred in the past 10 years. In my view, it is vitally important that this development continues. The use and complexity of software and computer equipment is growing steadily these days, and the need for good user manuals is likely to increase accordingly. Clearly, this need cannot be met through design that is based on intuition or trial-and-error. Rather, as the minimalist design process indicated, design should be iterative, abidingly seeking ways to integrate empirical findings into the design work. If design is performed in this way, I am convinced that the infamous saying "if all else fails, read the manual" will belong to the past before the next century.

References

Anderson, A., Knussen, C.L., & Kibby, M.R. (1993). Teaching teachers to use Hyper-

Card: A minimal manual approach. *British Journal of Educational Technology, 24*, 92 - 101.

Banathy, B.H. (1987). Instructional systems design. In R.M. Gagné (Ed.), *Instructional technology: Foundations* (pp. 85 - 112). Hillsdale: Erlbaum.

Biemans, H.J.A., & Simons, P.R.J. (1992). Learning to use a word processor with concurrent computer-assisted instruction. *Learning and Instruction, 2*, 321 - 338.

Black, J.B., Carroll, J.M., & McGuigan, S.M. (1987). What kind of minimal instruction manual is the most effective? In H.J. Bullinger, B. Shackel & K. Kornwachs (Eds.), *Proceedings of the second IFIP conference on human-computer interaction* (pp. 159 - 162). Amsterdam: Elsevier.

Brockmann, R.J. (1990). *Writing better computer user documentation: From paper to hypertext* (2nd. edition). New York: Wiley.

Carroll, J.M. (1990). *The Nürnberg Funnel: Designing minimalist instruction for practical computer skill*. Cambrigde: MIT.

Carroll, J.M. (1993). Creating a design science of human-computer interaction. *Interacting with Computers, 5*, 3 - 12.

Carroll, J.M., & Carrithers, C. (1984). Blocking learner error states in a training-wheels system. *Human Factors, 26*, 377 - 389.

Carroll, J.M., & Kay, D.S. (1985). Prompting, feedback and error correction in the design of a scenario machine. *Proceedings of the CHI'85 Conference on Human Factors in Computing Systems* (pp. 149 - 154). San Francisco: ACM.

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R., & Mazur-Rimetz, S.A. (1987). The minimal manual. *Human-Computer Interaction, 3*, 123 - 153.

Cohen, G., & Cunningham, D.H. (1984). *Creating technical manuals: A step-by-step approach*. New York: McGraw-Hill.

Feinberg, S. (1989). *Components of technical writing*. New York: Holt, Rinehart & Winston.

Frese, M., & Altmann, A. (1989). The treatment of errors in learning and training. In L. Bainbridge & S.A. Ruiz Quintanilla (Eds.), *Developing skills with information technology* (pp. 65 - 86). Chichester: Wiley.

Frese, M., Brodbeck, F., Heinbokel, T., Mooser, C., Schleiffenbaum, E., & Thieman, P. (1991). Errors in training computer skills: On the positive function of errors. *Human-Computer Interaction, 6*, 77 - 93.

Gayeski, D.M. (1991*a*). Rapid prototyping: A new model for developing multimedia. *Multimedia Review, 2*(3), 18 - 23.

Gayeski, D.M. (1991*b*). Software tools for empowering instructional developers. *Performance Improvement Quarterly, 4*(4), 21 - 36.

Gong, R., & Elkerton, J. (1990). Designing minimal documentation using a GOMS model: A usability evaluation of an engineering approach. In J. Carrasco Chew & J. Whiteside (Eds.), *Proceedings of the CHI'90 conference* (pp. 99 - 106). New York: ACM.

Grimm, S.J. (1987). *How to write computer documentation for users* (2nd. edition). New York: Van Nostrand Reinhold.

Hartley, J. (1985). *Designing instructional text*. London: Kogan Page.

Hendrix, W., & Van der Spek, E. (1993). *Gids voor het schrijven van software handleidingen* [Guide for writing software manuals]. Groningen: Martinus Nijhoff.

Horn, R.E. (1989). *Mapping hypertext: The analysis, organisation, and display of* *knowledge for the next generation of on-line text and graphics.* Lexington: The Lexington Institute.

Klauke, M. (1994). National standards: Their impact on text production and quality. In M. Steehouder, C. Jansen, P. van der Poort & R. Verheijen (Eds.), *Quality of technical documentation* (pp. 161 - 170). Amsterdam: Rodopi.

Laret, C. (1993). *Ontwerp en constructie van een handleiding voor een technische robot* [Design an development of a manual for an industrial robot]. Unpublished master's thesis, University of Twente, Enschede, The Netherlands.

Pakin, S.A. (1984). *Document development methodology.* Englewood Cliffs: Prentice Hall.

Pieters, J.M. (1992). *Het ongekende talent: Over het ontwerpen van artefacten in de instructietechnologie* [The unprecedented talent: On designing artefacts in instructional technology]. Enschede: University of Twente.

Pieters, J.M., & Bergman, R. (1993). *The empirical basis of designing instruction: What practice can contribute to theory.* Unpublished manuscript, University of Twente, Department of Instructional Technology.

Rosson, M.B., Carroll, J.M., & Bellamy, R.K.E. (1990). Smalltalk scaffolding: A case study of minimalist instruction. *Proceedings of the CHI'90 Conference on Human Factors in Computer Systems* (pp. 423 - 429). New York: ACM.

Rowland, G. (1992). What do instructional designers actually do? An investigation of expert practice. *Performance Improvement Quarterly, 5*(2), 65 - 86.

Scholtz, J., & Hansen, M. (1993). Usability testing a minimal manual for the Intel SatisFAXtion™ faxmodem. *IEEE Transactions on Professional Communication, 36*, 7 - 11.

Schön, D.A. (1983). *The reflective practitioner: How professionals think in action.* New York: Basic Books.

Schriver, K.A. (1989). Document design from 1980 to 1989: The challenges that remain. *Technical Communication, 36*, 316 - 331.

Steehouder, M.F. (1990). *Gids voor het schrijven van computerhandleidingen* [Guide for writing computer documentation] (4th. edition). Enschede: Universiteit Twente, Vakgroep Toegepaste Taalkunde.

Sullivan, P. (1988). Writers as total desktop publishers: Developing a conceptual approach to training. In E. Barrett (Ed.), Text, context, and hypertest (p.. 265 - 278). Cambridge: MIT.

Teurlings, C.C.J. (1993). *Leren tekstverwerken: Een nieuw perspectief* [Learning to use a word processor: A new perspective]. Ph.D. thesis, Katholieke Universiteit Brabant, Tilburg, The Netherlands.

Tripp, S.D., & Bichelmeyer, B. (1990). Rapid prototyping: An alternative instructional design strategy. *Educational Technology: Research and Development, 38*(1), 31 - 44.

Vanasse, S. (1994). Minimal manual and on-line examples for learning how to use interface construction toolkits. *Performance Improvement Quarterly, 7*(1), 80 - 96.

Vanderlinden, G., Cocklin, T.G., McKita, M. (1988). Testing and developing minimalist tutorials: A case history. *Proceedings of the 35th International Technical Communications Conference*, 196 - 199.

Van der Meij, H. (1992). A critical assessment of the minimalist approach to documentation. *SIGDOC'92 conference proceedings* (pp. 7 - 17). New York: ACM.

Van der Meij, H., & Carroll, J.M. (in press). Principles and heuristics for designing minimalist instruction. *Technical Communication.*

Van der Meij, H., & Lazonder, A.W. (1993). Assessment of the minimalist approach to computer user documentation. *Interacting with Computers, 5,* 355 - 370.

Venema, A. (1990). *Produktinformatie ter preventie van ongevallen in de privésfeer: Gevaars- en veiligheidsinformatie in handleidingen van gebruiksprodukten* [Product-information to prevent domestic accidents: Safety-information in user manuals]. Leiden: Stichting Wetenschappelijk Onderzoek Konsumentenaangelegenheden (SWOKA).

Weiss, E.H. (1991). *How to write usable user documentation* (2nd. edition). Phoenix: Oryx Press.

Wright, P. (1988). Issues of content and presentation in document design. In M. Helander (Ed.), *Handbook of human-computer interaction* (pp. 629 - 647). Amsterdam: Elsevier.

DUTCH SUMMARY

Nederlandse samenvatting

1. Inleiding

De computer is zo langzamerhand niet meer uit onze samenleving weg te denken. Dit komt voor een groot deel door de populariteit en het veelvuldig gebruik van tekstverwerkers. Tekstverwerken is voor veel volwassenen een belangrijke, zo niet de belangrijkste vorm van computergebruik. Bovendien is een tekstverwerker voor veel mensen hun eerste, en vaak enige kennismaking met de computer. Sterker nog, voor veel mensen *is* de tekstverwerker de computer.

Helaas verhult het schijnbare gemak waarmee ervaren computergebruikers met een tekstverwerker omgaan vaak de moeilijkheden die gepaard zijn gegaan met het *leren* omgaan met zo'n programma. Leren tekstverwerken lijkt daardoor gemakkelijker dan het is. Althans, op het eerste gezicht. Ondanks het feit dat de huidige generatie tekstverwerkers duidelijk verbeterd is ten opzichte van de vroegere line-editors ervaren beginnende computergebruikers nog steeds talloze problemen.

Een aantal van deze problemen wordt veroorzaakt door de software. Voor beginnende gebruikers is een tekstverwerker niet bepaald het meest gebruikersvriendelijke programma. In veel gevallen geeft wat op het beeldscherm verschijnt een slecht beeld van wat je als gebruiker met het programma kunt doen. Ditzelfde geldt ten aanzien van de commando's. Zeker voor beginnende gebruikers zijn de namen van commando's soms net een cryptogram en worden de resultaten van die commando's al even raadselachtig op het scherm weergegeven.

De handleiding die bij de computer of de software geleverd wordt biedt zelden een kant-en-klare oplossing voor deze problemen. Integendeel, de handleiding zelf veroorzaakt – ironisch genoeg – ook vaak de nodige problemen. Deze problemen ontstaan doordat handleidingen onvoldoende aansluiten bij de wensen, behoeften en leerstijlen van beginnende gebruikers. Zo bevatten deze handleidingen bijvoorbeeld te veel uitleg en te weinig oefeningen. Verder wordt aangenomen dat de gebruiker de handleiding altijd van A tot Z doorwerkt, de voorgeschreven acties stap-voor-stap uitvoert en steeds foutloos handelt.

In dit proefschrift wordt een oplossing gezocht voor de problemen die beginnende computergebruikers met de handleiding hebben. Onderzocht wordt welke instructie-ontwerp principes moeten worden toegepast in een (les)handleiding voor beginnende computergebruikers. Anders gezegd: aan welke eisen moet een handleiding voor het leren tekstverwerken voldoen. Bij de beantwoording van deze onderzoeksvraag wordt uitgegaan van een theorie die bekend staat als minimalisme. Het minimalisme is een recente benadering, speciaal bedoeld voor het voor het ontwerpen van zelfinstructie materialen voor het leren werken met computer programmatuur.

2. De minimale handleiding

In het begin van de jaren tachtig ontwikkelden John Carroll en zijn collega's bij IBM de minimalistische benadering voor het ontwerpen van computer-handleidingen. Op basis van uitgebreide observaties van beginnende computergebruikers kreeg deze benadering gestalte in de *minimal manual*. Deze minimale handleiding onderscheidde zich van conventionele handleidingen doordat zij gebruikers meer vrijheid gaf om naar eigen inzicht te handelen. Het 'minimale' karakter van deze handleiding blijkt uit de inhoud: zij bevat bondige uitleg en geeft gebruikers steeds de gelegenheid om aan betekenisvolle taken te werken.

Een minimale handleiding wordt gedefinieerd door vier minimalistische principes: (1) actiegerichtheid, (2) optimaal gebruik van tekst, (3) ondersteuning van fouten, en (4) modulariteit.

2.1 ACTIEGERICHTHEID

Een minimale handleiding is actiegericht; zij stelt de gebruikers in staat vrijwel direct aan betekenisvolle taken te werken. Er wordt vrijwel geen aandacht besteed aan onderwerpen als het installeren van het programma of het veranderen van de standaardinstellingen. Ook wordt weinig uitleg gegeven over bijvoorbeeld de werking van rolmenu's of speciale toetsen. In plaats daarvan ondersteunt de minimale handleiding bekende en voor de gebruikers relevante taken zoals het typen van een uitnodiging voor een feest, het vormgeven van een brief aan de PTT of het corrigeren van de notulen van een ledenvergadering.

De actiegerichtheid komt ook tot uitdrukking in de hoofdstuk- en paragraaftitels. Deze geven altijd aan wat een gebruiker met het programma kan doen (bijvoorbeeld "Typen", "Een tekst bewaren", of "De kantlijn verschuiven"). Dit in tegenstelling tot kopjes als "Van en naar DOS" en "Diverse handige hulpmiddelen". Er zal best wat belangrijks in deze secties staan, maar de kopjes maken niet op voorhand duidelijk welke acties zij bevatten.

Vrijwel elk hoofdstuk bevat bovendien een "Zelf proberen" sectie waarin gebruikers aangespoord worden de werking van 'extra' opties te proberen. Deze extra's sluiten nauw aan bij het geleerde in dat hoofdstuk. Zo worden gebruikers nadat zij het onderstrepen hebben geoefend, aangespoord om diverse andere mogelijkheden om tekst te accentueren te proberen.

2.2 OPTIMAAL TEKSTGEBRUIK

Om de tekst in een minimale handleiding zo optimaal mogelijk op de gebruikers af te stemmen gelden twee basisregels. Ten eerste moet de hoeveelheid tekst tot het minimum beperkt worden. Een minimale handleiding heeft daarom geen voorwoord, inleiding, index, samenvattingen en overzichten. Daarnaast wordt ook weinig tot geen conceptuele informatie gegeven. Sterker nog, zelf de actie-informatie is niet altijd volledig. Aanwijzingen die op het scherm te vinden zijn of gemakkelijk afgeleid kunnen worden, zijn vaak opzettelijk weggelaten.

Verder moet de tekst in een minimale handleiding zo eenvoudig mogelijk zijn. De reden hiervoor ligt voor de hand. Omdat het leren tekstverwerken al ingewikkeld genoeg is, moet het lezen en begrijpen van de tekst als het ware vanzelf gaan. Daarom is de gemiddelde zinslengte in een minimale handleiding kort, ongeveer 12 tot 14 woorden. Er worden geen samengestelde zinnen gebruikt. Bovendien zijn jargon en technische termen vervangen door hun alledaagse synoniemen.

2.3 ONDERSTEUNING VAN FOUTEN

Doordat een minimale handleiding gebruikers de vrijheid geeft naar eigen inzicht te handelen, en hen daartoe zelfs stimuleert, neemt ook de kans op het maken van fouten toe. Een minimale handleiding bevat daarom veel informatie voor het ontdekken en herstellen van fouten. In hoofdstuk 1 staan de algemeen toepasbare manieren om fouten te herstellen (bijvoorbeeld het UNDO commando of de ESC toets). Daarnaast bevat elk hoofdstuk fouten-informatie. Fouten-informatie is een soort vangnet; het ondersteunt zowel handelingen waarbij in de regel veel fouten gemaakt worden als acties waarbij het risico bestaat dat gebruikers na een fout niet meer verder kunnen.

Gebruikers doorlopen gewoonlijk drie fasen bij het herstellen van een fout: detectie, diagnose en correctie. De fouten-informatie in de minimale handleiding bestaat daarom uit informatie die al deze fasen ondersteunt. Daarbij wordt een vaste volgorde aangehouden. Eerst wordt informatie gegeven om een fout te ontdekken, daarna informatie over de meest waarschijnlijke oorzaak van de fout en tenslotte informatie om de fout te herstellen. Omdat gebruikers die een bepaalde fout *niet* hebben gemaakt de fouten-informatie ook zullen lezen, wordt deze als een voorwaardelijke conditie geformuleerd: "Als de tekst ... op het scherm verschijnt (detectie), dan ... (diagnose) ... dan ... (correctie)".

Ondanks de aanwezigheid van fouten-informatie blijven nog (te) veel fouten onopgemerkt. Dit komt doordat beginnende gebruikers hun aandacht vaak niet goed over handleiding, toetsenbord en beeldscherm weten te verdelen. Ze kijken te weinig naar het scherm, waardoor fouten zich opstapelen en het steeds moeilijker wordt om ze te herstellen. Om dit te voorkomen bevat een minimale handleiding coördinerende informatie. Deze richt de aandacht van de gebruiker van tijd tot tijd op het scherm, bijvoorbeeld om na te gaan waar een melding van het programma verschijnt. Een voorbeeld van coördinerende informatie is "De tekst **A:\TEKST1.DOC** verschijnt op het scherm. Ga na of dat zo is."

2.4 MODULARITEIT

De modulaire opbouw van een minimale handleiding komt het best tot uitdrukking in de hoofdstukken. Elk hoofdstuk vormt een op zichzelf staand, afgerond geheel. Omdat een minimale handleiding de gebruikers in staat stelt om te doen wat ze zelf willen is dit essentieel. Sommige gebruikers zullen zomaar middenin de handleiding willen beginnen, bijvoorbeeld omdat ze al over enkele basisvaardigheden (denken te) beschikken. Ook degene die na een pauze wil herstarten met de handleiding moet niet gedwongen worden weer van voren af aan te beginnen.

De hoofdstukken in een minimale handleiding zijn kort, hun lengte varieert tussen twee en vier bladzijden. Zij geven de gebruiker de indruk dat het doorwerken ervan – en daardoor het leren tekstverwerken – geen bovenmenselijke inspanning kost. Als vuistregel geldt dat elk hoofdstuk door vrijwel alle gebruikers in dertig minuten door te werken moet zijn.

3. Het effect van de minimale handleiding

Deze principes illustreren een geheel nieuwe visie op het ontwerpen van leshandleidingen voor beginnende computergebruikers. Hoewel de minimalistische ideeën zeker veelbelovend zijn, is het de vraag of een minimale handleiding in de praktijk ook inderdaad beter werkt dan een conventionele handleiding. Onderzoek naar de effectiviteit van minimale handleidingen toont aan dat dit inderdaad het geval is. Zo vonden Carroll en zijn collega's bijvoorbeeld dat beginnende gebruikers in 40% minder tijd 58%

meer dingen leerden met een minimale dan met een conventionele handleiding. Bovendien maakten zij 20% minder fouten en waren efficiënter in het herstellen van fouten.

Ook uit andere onderzoeken blijkt de effectiviteit van een minimale handleiding. Toch is het moeilijk om de resultaten uit deze studies op de juiste waarde te schatten. Veel onderzoeken (inclusief dat van Carroll c.s.) beschrijven namelijk onvoldoende gedetailleerd welke principes gebruikt zijn voor het ontwerpen van de minimale handleiding. Bovendien geven de methodologische condities waaronder sommige resultaten verkregen zijn aanleiding tot enige voorzichtigheid.

Deze en andere methodologische kritieken vormden de aanleiding voor een eigen onderzoek naar de werking van de minimale handleiding. Het primaire doel van dit onderzoek was het valideren van de resultaten uit het oorspronkelijke onderzoek van Carroll. Daarnaast is nagegaan of computerervaring de effectiviteit van een minimale handleiding beïnvloedt.

Om deze vragen te beantwoorden zijn de prestaties van twee groepen gebruikers vergeleken. De ene groep leerde tekstverwerken met een minimale handleiding, de andere groep met een conventionele (controle) handleiding. De inhoud van beide handleidingen was identiek: de basisbeginselen van het tekstverwerken met WordPerfect 5.1. Zij verschilden alleen ten aanzien van de minimalistische principes. De minimale handleiding was een zo exact mogelijke kopie van Carroll's originele handleiding; de controle handleiding was afgeleid van een veelgebruikte, bekroonde Nederlandse handleiding voor WordPerfect.

De resultaten van dit onderzoek kwamen overeen met die van Carroll. Gebruikers die met de minimale handleiding leerden tekstverwerken waren sneller tijdens de training en tijdens de test. Zij maakten meer testopgaven goed, maakten daarbij minder fouten en waren vaker in staat hun fouten succesvol te herstellen. Deze gegevens ondersteunen de conclusie dat een minimale handleiding een effectieve en efficiënte manier is om te leren tekstverwerken.

Verder bleek uit dit onderzoek dat computerervaring deze resultaten niet beïnvloedt. De belangrijkste conclusie die hieruit getrokken kan worden is dat een minimale handleiding net zo effectief is voor beginnende als voor meer ervaren computergebruikers.

Alleen weten *dat* de minimale handleiding werkt is echter niet voldoende; weten *waarom* dat zo is, is minstens zo belangrijk. Anders gezegd, op welke wijze dragen de afzonderlijke minimalistische principes bij aan het totale effect van de minimale handleiding? Om deze vraag te beantwoorden moeten de minimalistische principe afzonderlijk bestudeerd worden. Het onderzoek naar het effect van één van deze principes, het ondersteunen van fouten door fouten-informatie, wordt in de volgende paragraaf beschreven.

4. Het effect van fouten en fouten-informatie

Van fouten kun je leren. Bijvoorbeeld hoe je een gemaakte fout in het vervolg kunt voorkomen, hoe je een fout kunt herstellen of waarom bepaalde acties niet tot het gewenste resultaat hebben geleid. Fouten zijn echter niet automatisch effectief. Dit hangt af van de mate waarin ze door de instructie, in dit geval de handleiding, ondersteund worden. Alleen wanneer gebruikers directe feedback op gemaakte fouten krijgen, zullen zij daar wat van opsteken.

Een minimale handleiding biedt deze ondersteuning in de vorm van fouten-informatie. Wil fouten-informatie effectief zijn, dan zal ze zo veel mogelijk moeten aansluiten bij wat gebruikers doen wanneer ze een fout herstellen. In paragraaf 2.3 werd gesteld dat gebruikers daarbij drie fasen doorlopen: eerst ontdekken ze de fout (detectie), bepalen de oorzaak daarvan (diagnose) en tenslotte herstellen ze de fout (correctie).

In de detectie-fase moeten gebruikers de fout signaleren. De fouten-informatie in de handleiding wijst de gebruikers daarom op een zichtbare, maar misschien (nog) niet waargenomen mededeling op het scherm. Om een fout snel te kunnen ontdekken staat fouten-informatie altijd zo dicht mogelijk bij de mogelijk foute handeling. Snelle detectie is belangrijk; het voorkomt een opeenstapeling van fouten. Bovendien liggen de handelingen dan nog vers in het geheugen waardoor een goede diagnose en een spoedig herstel mogelijk zijn.

In de diagnose-fase stelt de gebruiker vast welke fout gemaakt is. Wanneer het programma dit aangeeft, vallen detectie en diagnose samen. De aanwezigheid en het soort fout worden dan gelijktijdig opgemerkt. Dit is bijvoorbeeld het geval met een melding als "FOUT -- bestand **TEKT1.DOC** niet gevonden". Het programma geeft aan dat er een fout gemaakt is en vermeldt bovendien welke fout dat is: een typefout (de letter "s" in "tekst" is vergeten).

In de diagnose-fase kan ook een analyse van de oorzaak van de fout plaatsvinden. In de meeste gevallen zal de gebruiker zelf de melding op het scherm moeten interpreteren om de aard van de fout vast te kunnen stellen. Hoewel dit bij de bovengenoemde foutmelding relatief eenvoudig en eenduidig is, is dit eerder uitzondering dan regel. Omdat er vaak verschillende mogelijkheden zijn waarom een fout gemaakt is, valt de precieze oorzaak slechts in een beperkt aantal gevallen te voorspellen.

In de correctie-fase stelt de gebruiker zich een nieuw doel. Dat kan zijn het herstellen van de gemaakte fout en daarna verder gaan met waar men gebleven was. In de praktijk blijkt echter dat gebruikers alleen in het herstel van een fout geïnteresseerd zijn als dit nodig is om verder te kunnen werken. Bij de meeste tekstverwerkers is dit vaak niet het geval en kan volstaan worden met het opnieuw uitvoeren van de actie(s). Correctie komt dan neer op het herpositioneren van de cursor en het nogmaals, nu correct, uitvoeren van de eerder gevolgde procedure.

Het effect van fouten-informatie is in drie experimenten onderzocht. In het eerste onderzoek zijn de leerprestaties van twee groepen gebruikers vergeleken. De ene groep leerde tekstverwerken met een minimale handleiding mèt fouten-informatie. De andere groep deed dit met een minimale handleiding waaruit alle fouten-informatie was verwijderd. De leerprestaties werden gemeten met drie testen: één test voor het uitvoeren van taken met WordPerfect (constructieve vaardigheden) en twee testen voor het ontdekken en herstellen van fouten (correctieve vaardigheden).

Fouten-informatie bleek de leerprestaties nauwelijks te beïnvloeden. Zowel tijdens de training als tijdens de testfase waren gebruikers uit beide condities even snel. Ook de scores op de constructieve test liet geen duidelijk verschil tussen de twee groepen zien. Op de correctieve testen liepen de prestaties uiteen, echter niet altijd in het voordeel van de fouten-informatie groep: op sommige onderdelen scoorde de controle groep zelf beter.

Dit onderzoek suggereert dat de aanwezigheid van fouten-informatie niet bijdraagt aan het totale effect van de minimale handleiding. Deze conclusie is echter niet geheel gerechtvaardigd. Hoewel fouten-informatie in dit onderzoek de leerprestaties niet beïnvloedde, kan het een effect op het leer*proces* gehad hebben. Anders gezegd, tijdens de training kan de fouten-informatie de ontwikkeling van constructieve en correctieve vaardigheden ondersteund hebben. Deze veronderstelling kan echter pas getoetst worden wanneer vast is komen te staan dat fouten-informatie tijdens de training daadwerkelijk geraadpleegd wordt.

Het feitelijke gebruik van fouten-informatie is in een kleinschalig experiment onderzocht. Acht beginnende computergebruikers leerden tekstverwerken met een minimale handleiding. Ongeveer twintig procent van deze handleiding bestond uit fouten-informatie. Tijdens de training werden de gebruikers geobserveerd. Omdat in dit onderzoek uitsluitend naar het leerproces werd gekeken, werden na afloop van de training geen testen afgenomen.

De observatiegegevens duiden op een frequent gebruik van fouten-informatie. Verder bleek het raadplegen van fouten-informatie zich niet te beperken tot situaties waarin een gemaakte fout ontdekt of hersteld moest worden. Ook wanneer er geen fout was gemaakt controleerden de gebruikers aan de hand van de fouten-informatie of de beschreven fout zich had voorgedaan. Deze en andere gegevens leidde daarnaast tot een aantal aanwijzingen voor verdere verbetering van de handleiding.

Met de verbeterde versie van de handleiding is een derde onderzoek uitgevoerd. In feite is dit onderzoek een combinatie van de twee hiervoor beschreven experimenten. De opzet van het onderzoek is vrijwel identiek aan die van het eerste onderzoek naar fouten-informatie. Het enige verschil is dat behalve naar leerprestaties ook naar leeractiviteiten gekeken is.

Uit dit onderzoek bleek dat gebruikers van de minimale handleiding mèt fouten-informatie tijdens de training minder fouten maakten en sneller en beter waren in het herstellen van fouten. Hierdoor hadden deze gebruikers minder tijd nodig voor de training. Ook de prestaties op de correctieve testen verschilden in het voordeel van de fouten-informatie groep. Gebruikers uit deze groep waren beter in het aangeven van de oorzaak van een fout (diagnose) en het herstellen ervan (correctie).

Deze resultaten leidden enerzijds tot de conclusie dat het ondersteunen van fouten door fouten-informatie gebruikers helpt bij het leren tekstverwerken. Anderzijds geeft dit onderzoek aan op welke wijze fouten-informatie bijdraagt aan de werking van een minimale handleiding. Fouten-informatie heeft met name effect op de ontwikkeling van correctieve vaardigheden. Gebruikers die tijdens de training fouten-informatie konden raadplegen waren zowel tijdens als na de training beter in staat met fouten om te gaan. Daarnaast heeft de aanwezigheid van fouten-informatie geen negatief effect op de ontwikkeling van constructieve vaardigheden. Dit maakt fouten-informatie een onmisbaar element in een (minimale) leshandleiding voor het leren omgaan met een computerprogramma.

Appendices

APPENDIX 1

Basic word processing tasks

Getting started

Turning the computer on
Starting the word processor
Ending the word processor

Creating a document

Typing text
Saving text

Revising an existing document

Clearing the screen
Retrieving a document
Moving the cursor
Correcting typing errors
Inserting and removing a blank line

Printing

Setting up the printer
Consulting the 'print preview'
Printing a document

Browsing through a document

Moving the cursor: shortcuts
Searching a text

Rearranging text

Deleting text
Moving text
Copying text

Formatting characters

Underlining text
Removing the underlines
Formatting text 'differently'

Changing the layout of a document

Changing the base font
Enlarging and reducing characters
Enlarging and reducing the line spacing

Changing the margins

Indenting the first line
Centering text
Alligning text 'flush right'

APPENDIX 2

Illustrative pages of the minimal manual[13]

7. Formatting characters

Underlining text

With WordPerfect you can emphasize parts of the text, for example by underlining them.

1 Retrieve the document COFFEE.WP

*If the text COFFEE.WP does not appear on the status line, you did not clear the screen first. Press the F7 key and type an **N** twice to clear the screen as yet.*

2 Block the words 'half a million dollars'

You can undo the block function by pressing the F1 key.

3 Go to the menubar and select the option FONT.
4 Press the ↓ key until you reach the command APPEARANCE

A so-called submenu appears on the screen.

5 Press the • key once to enter the submenu.
6 Press the ↓ key to select the command UNDERLINE
7 Press the ENTER key to underline 'half a million dollars'

You can consult the 'Print preview' to see if 'half a million dollars' has actually been underlined.

7.1

formatting characters

[13] translated from the (Dutch) minimal manual that was used in the experiment in chapter 3. Due to this translation, some minimalist principles may not show up well.

Removing the underlines

WordPerfect uses so-called hidden codes. Strictly speaking, if you underline something, you command WordPerfect to 'underline everything that is selected'. To remove these underlines, you must remove the hidden codes.

1 Go to the menubar, select the option EDIT and choose the command REVEAL CODES

The screen is now split in half. In the upper half you see the text in the typing area. The lower half shows the same text *with* the hidden codes. This part is called the underwater screen.

2 Position the cursor at the underlined words 'half a million dollars'
3 Position the cursor in the underwater screen on the code **[UND]** or **[und]** (it makes no difference which code you choose).
4 Press the DELETE key.

Look at the underwater screen. The codes for underlining text have disappeared: the words are no longer underlined.

5 Choose the REVEAL CODES command again to restore the normal screen.

On your own

With WordPerfect there are numerous ways to highlight text. For example, **boldface**, *italic* or shadow printing. These commands work just like the underline command.

Try these techniques yourself. You can see the results by consluting the ‘Print preview’.

7.2

formatting characters

APPENDIX 3

Illustrative pages of the self-study manual[14]

5. Editing text

This chapter deals with formatting (or editing) characters, words and lines. Among others, topics like underlining text, changing the margins and centering text will be discussed.

Please note (1): with most of these options, hidden codes are placed in the text. As was explained in section 4.4, these codes can be displayed by using the REVEAL CODES command

Please note (2): in this chapter the term 'base font' is used. The base font determines the size and style of the basic text. Typefaces like Helvetica, Times Roman and Courier are different base fonts. When the program uses, for example, Times Roman with a font size of 11 points as its base font, all text will be printed in 11-point Times Roman.

Please note (3): the size of the characters that will be printed is called the font size. The font size is expressed in points. One point equals 1/72 inch. Its notation is **1p**. (So: 72 p = 1 inch, or 2.54 cm).

5.1 Modifying characters

Modifying characters means: changing the appearance of characters without changing the base font and font size. Examples of such modifications are boldface, underline, or italic. The methods to modify text are highly similar; only the corresponding commands differ. All

41

editing text

[14] translated from the (Dutch) self-study manual used as control manual in the experiment in chapter 3. Due to a different sequencing of the content of this manual, the reveal codes command is discussed in a different chapter

commands are listed under the option FONT. The procedure to activate them is as follows.

Modifying new text

When new text has to be underlined, boldfaced, put in italics or highlighted in another way, you should do as follows:

1 Press [ALT].
2 Press [•] to select the option FONT.
3 Press [↓] to select the command APPEARANCE; a so-called submenu appears.
4 Press [•] to enter the submenu.
5 Press [↓] to select the desired command.
6 Press [ENTER].
7 Type the text; it will be modified as you type.
8 Choose the same command again to return to the standard textmode.

Please note: codes to modify text are paired codes (see section 4.3). To return the text to normal, you can also press [•] once. The cursor is then placed right after the code for the modification. In the typing area, this action is not visible; to see what happens, check the underwater screen.

Modifying previously typed text

To boldface, underline or italicalize existing characters, take the following steps:

1 Block the text you want to modify (see section 3.2).
2 Press [ALT].
3 Press [•] to select the option FONT.
4 Press [↓] to select the command APPEARANCE; a so-called submenu appears.
5 Press [•] to enter the submenu.
6 Press [↓] to select the desired command.
7 Press [ENTER].

Inserting text to modifications

When text is, for example, underlined, it is enclosed by the following codes:

- **[UND]** as the starting code (text is underlined from here).
- **[und]** as the ending code (underlining stops here).

These codes can be envisioned by using the command REVEAL CODES. The appearance of text that is inserted between these two codes is automatically modified as the already existing text. This implies that:

- inserting text between two words that are already underlined does not require additional modification. Just position the cursor between the two words and type the addition.
- inserting text at the beginning of underlined text requires the cursor to be positioned after the starting code **[UND]**. Look at the status line or consult the underwater screen to verify if the cursor is right after the starting code.
- inserting text at the end of underlined text requires the cursor to be positioned right before the ending code **[und]**. Look at the status line or consult the underwater screen to verify if such is the case. If so, the cursor is positioned correctly and the new text will be underlined.

Undoing modifications

Markings like boldface or underline are negated by removing their codes. As with every paired code, only one of them has to be deleted; the remaining code is automatically removed by the program. To remove codes, it is best to display them first with the REVEAL CODES command. Then position the cursor on one of the codes and press [DELETE].

APPENDIX 4

Motivational questionnaires experiment 1[15]

3.1 Initial motivation

RELEVANCE (□= 0.77)

1. I type my letters, memos, reports and the like.
2. I pay attention to the lay-out of letters, memos, reports and the like.
3. I type or write official letters.
4. Having one's own computer is useful.
5. I find it important that my letters, memos, reports and the like look fine.
6. There is no need to type letters, memos, reports and the like.
7. I write official letters.
8. I would buy a computer if I had the money.
9. I will need to do a lot of typing this year.
10. At the utmost I will be preparing two reports this year.

CURIOSITY (□= 0.84)

1. I read about computers.
2. I am fascinated by modern technology.
3. I want to know how technical apparatuses work.
4. I pay attention to technology.
5. Technical developments fascinate me.
6. I am absolutely not interested in how technical apparatuses work.
7. I am interested in computers.
8. I find it important to keep in touch with technological developments.

REFERENCE GROUP (□= 0.70)

1. I know many people who own a computer.
2. My relatives, friends and acquaintances are interested in computers.
3. The people that I know are fascinated with computers.
4. My relatives, friends and acquaintances own a computer.

[15] all items are translated from Dutch.

CONFIDENCE/PRIOR KNOWLEGDE (□= 0.91)

1. I find it difficult to learn how to operate a technical apparatus.
2. I know almost nothing about technical apparatuses.
3. I quickly understand how technical apparatuses work.
4. I can easily handle technical apparatuses.
5. My friends are better at handling technical equipment than me.
6. Compared with the people that I know I am handy with technical apparatuses.
7. I quickly know how a technical apparatus works.
8. It takes me a long time before I understand how to use a technical apparatus.
9. I cannot handle technical apparatuses very well.

PERSISTENCE (□= 0.77)

1. When I am having a problem, I keep working on it until it is solved.
2. I finish what I start.
3. If I cannot solve a problem quickly, I will leave it be.
4. If I start something it does not mean that I will also finish it.

3.2 Motivation after training

SATISFACTION (□= 0.81)

1. I like word processing.
2. I find working with WordPerfect stimulating.
3. Word processing is boring.
4. It didn't come easy today.
5. It took me longer to learn WordPerfect than I expected.
6. My knowledge of WordPerfect is very useful.
7. I get annoyed from word processing.
8. Word processing is unpleasant.
9. Word processing is frustrating.
10. Word processing is scary.

CONFIDENCE (□= 0.85)

1. Now I know how to use a wordprocessor.
2. WordPerfect is difficult.
3. I find it scary to work with the computer.

4. I find word processing scary.
5. Now I can do my word processing without help.
6. I feel that I can work well with WordPerfect.
7. Word processing is simple.
8. I am confident with regard to word processing.

RELEVANCE (□= 0.78)

1. Wordprocessors are handy.
2. My knowledge of WordPerfect comes in handy for my study, work or hobby.
3. I rather type my letters, memos, reports and the like on a typewriter than with a wordprocessor.
4. I don't think that I will be using WordPerfect shortly.
5. I would like to use WordPerfect as quickly as possible.

ATTENTION (□= 0.73)

1. The manual directs your attention to important things.
2. The manual gives good cues.
3. The manual gave good suggestions what to look for.

APPENDIX 5

Confidence questionnaires experiment 2[16]

4.1 Initial confidence

CONFIDENCE ITEMS

2[17]. I find it difficult to learn how to operate a technical apparatus.
4. I know almost nothing about technical apparatuses.
6. I quickly understand how technical apparatuses work.
9. I can easily handle technical apparatuses.
11. My friends are better at handling technical equipment than me.
14. Compared with the people that I know I am handy with technical apparatuses.
15. I quickly know how a technical apparatus works.
18. It takes me a long time before I understand how to use a technical apparatus.
20. I cannot handle technical apparatuses very well.

FILLER ITEMS

1. I work very concentrated.
3. My relatives, friends and acquiantances think working with a computer is difficult.
5. Having one's own computer is useful.
7. When I am having a problem, I keep working on it until it is solved.
8. I find it importantn that my letters, memos, reports and the like look fine.
10. If I cannot solve a problem quickly, I leave it be.
12. I am fascinated by modern technology.
13. I would buy a computer if I had the money.
16. I am not interested in computers.
17. My official letters always look fine.
19. When something interests me, I want to know all the ins and outs.

[16]all items are translated from Dutch.

[17]numbers indicate the order in which the items appeared in the questionnaire.