THE HUMAN FACTOR HAS KNOCKED THREE TIMES
Three books on the ergonomics of computing systems

B. Meyer

SOFTWARE PSYCHOLOGY: HUMAN FACTORS IN COMPUTER AND INFORMATION SYSTEMS, by B. Shneiderman. *Winthrop,* Cambridge (Mass.), U.S.A., 1980, ISBN 0-87626-816-5.

DIRECTIONS IN HUMAN FACTORS FOR INTERACTIVE SYSTEMS, by H. Ledgard, A. Singer and J. Whiteside. *Springer-Verlag,* Berlin, Heidelberg and New York, 1981, ISBN 3-540-10574-3 and 0-387-10574-3.

THE PSYCHOLOGY OF HUMAN-COMPUTER INTERACTION, by S.P. Card, T.P. Moran and A. Newell. *Lawrence Erlbaum Associates,* Hillsdale (NJ), U.S.A., 1983, ISBN 0-89859-243-7.

What is the best way of building user-friendly computing systems? This is undoubtedly a naive question but its importance will be recognized by any user of present-day systems. Specialists in this field of research give various names to it: the ergonomics of computing systems, the study of human factors, software psychology, human engineering, user psychology, user science, cognitive engineering, etc.

The lack of a single agreed term is a clear sign of how young the discipline is, and of the lack of concrete results produced in it to date. But, unquestionably, its current development meets a need which is increasingly widely felt.

The three books reviewed here approach this subject from rather different points of view. The earliest (1980) is Shneiderman's, which gives the broadest coverage and is concerned with all psychological problems linked with software, i.e. not merely with the use of programs (which is what concerns us here) but also with their production. Ledgard, Singer and Whiteside (1981) deal with the study of 'human factors'. Like Schneiderman, the authors are computer scientists concerned with making the systems they have produced more user-friendly. The most recent work, by Card, Moran and Newell — which had just appeared as I began this review — has a different slant: it is a very concise examination of the 'cognitive' aspects of the use of interactive systems, based on the results of current research in cognitive psychology. The approach is multidisciplinary: although at least one of the authors (Newell) is well known in the computer science field (in particular that of artificial intelligence), most of the references in their bibliography are extracts from reviews such as *Cognitive Psychology, Journal of Experimental Psychology,* etc.

1. Shneiderman

As I have already suggested, Shneiderman's work is broader in aim than the other two. The general subject is the 'psychological approach' to programming problems, and this covers topics as diverse as how to motivate a team of programmers, programming style, evaluation of software quality, data base management systems, use of natural language and design of interactive interfaces. It is no bad thing to find considerations of such very different fields brought together in a single book in this way; on the majority of questions covered, Shneiderman is more precise than [Weinberg 71], the first work on the psychology of programming, which today seems somewhat simplistic. However, a survey as broad as Shneiderman's can hardly avoid casting the net a little too wide.

In my view, the topic of the work should have been dealt with in two books. The book does, in any case, divide naturally into two parts (chapters 1–6 and 7–11, respectively). The first considers the psychological aspects of software production, and the second the psychological study of the use of computing systems, and the conclusions to be drawn by designers.

The first section is, in a sense, a summary of some current ideas on software engineering, and it covers control structures as well as 'the science of software' and reliability models. These are all subjects on which much has already been written; Shneiderman's original contribution is that he applies the psychological point of view to these subjects and makes systematic use of quantitative results, usually from statistical tests. I am not a great supporter of quantitative methods in this field; it appears to me that the most significant progress as regards programming methods and languages has been obtained by a deductive rather than an inductive approach (though against this view, one might quote Dijkstra's famous article on the dangers of branching, which began with the no less famous remark that the author had noticed for some time that the quality of programmers varies inversely with the number of GOTOs in their programs, which is certainly an inductive, though not really quantitative, argument). It can however be interesting, even if one is completely convinced, for example, of the guarantee of security provided by strongly typed languages, to read or re-read the results of the Gannon study [Gannon 77], reproduced by Shneiderman, which support this point of view and are based on systematic experiments.

The second section, whose subject falls more directly within the framework of this comparative review, is also rather too broad. There are two chapters dealing with data bases, a subject on which Shneiderman has already published several studies. Was it really necessary to include a whole discussion on the various data models? I don't think so; Shneiderman's view of these questions is valuable but it belongs in another book. Finally, of the fifty pages devoted to data bases, only a few are really relevant; these are the ones which discuss the form which a good query language should take.

One chapter is devoted to the use of natural language for communication with computers. Shneiderman has considerable reservations (and rightly so, I feel), and shows that there is no reason to assume a priori that the same type of formalism is suitable for communication with machines as for communication with human beings.

The two chapters devoted to the design of interactive systems and their external interfaces are, in my view, the most interesting in the book. Chapter 1 in particular (Designing Interactive Systems) gives the results of many studies, some of which are difficult to find otherwise. It is useful to examine and compare the methods proposed by the various authors for designing good quality, interactive systems; one is less struck by the disagreements which arise (although some are considerable), than by the still very empirical nature of much of the work in this field, and by the lack of general criteria for problems such as error processing and ease of learning.

In fact, some of the advice given appears questionable. Many authors, including Shneiderman himself, state, like Hansen [Hansen 71], that the first principle is to have a good knowledge of the prospective user (know the user). This precept seems dangerous to me, if followed to the letter. Many systems — in fact, all successful systems — end up being used by a far larger group than that for which they were initially designed; they therefore decrease in value if they are tailored too specifically to the needs of their first intended users. Two examples, from two very different fields, are Fortran and Unix: in each case, distribution has exceeded the initial target in an unexpected way. In such situations, the systems impose restrictions of use which make sense for the first generation of users but have no meaning for later ones. One could, of course, reply that if systems of this kind have 'succeeded' it is precisely because they suited their initial users perfectly. But it seems more useful to me to study the common characteristic of all users of computing systems, rather than to concentrate exclusively on the specific needs of a particular group, especially as that group will soon only represent a minority of all users, if the system is a success.

One part of chapter 10, on points such as the respective merits of selection by menu or by command, will, in my view, soon seem as outdated as the section which seriously ponders which of batch or interactive operation is to be preferred. Progress in hardware will soon settle this kind of question. In addition, users themselves will no longer accept just anything: the children of the video game era will certainly be less indulgent towards text editors with torturous syntax than their parents, brought up the hard way with batch processing and punched cards. Shneiderman's book gives a fairly good summary of the new perspectives provided by current methods of man-machine communication; it deserves to be read in conjunction with a more recent article by the same author [Shneiderman 83], devoted to what he calls direct manipulation handling. In this article, Shneiderman examines a number of interactive systems with satisfied, or even downright enthusiastic, users. The purpose of these systems are varied: computer assisted instruction, CAD, 'full page' text editing, video games. The common characteristic which the author has discovered in these different implementations is that they operate by direct manipulation: the user is always provided with an explicit (and often graphic) representation of the current state of the system; each of his actions leads to immediate modification of this representation. The user can see what he has, and what he is doing. It is the feeling of security and power provided by this operating mode which explains its success; one may assume that the interactive systems of the future will increasingly follow this pattern.

In the same way that underground trains always have a last carriage (which one comedian suggested removing since it is always the most crowded), books always have a last chapter. In the case of Shneiderman's book, its last chapter is a chapter too many: in fact, I cannot feel that his final comments on 'Computer power by and for the people' have much of a contribution to make. Virtuous sentiments abound, but rather too much in the 'mother's apple pie' tradition. The final straw for me was the appearance of a series of quotations from R. Pirsig's *Zen and the art of motorcycle maintenance*, which was very fashionable for a time with a fringe group of young American intellectuals, but is very difficult for a European reader to take seriously. I must confess that I read no further, but then there were only a few pages left.

As we have seen, the book has some irritating features. But, as the very length of this review may suggest, for all my reservations, I consider it to be significant and useful. Let us hope that the author will some day provide us with a more succinct and penetrating work on a better defined subject.

2. Ledgard, Singer and Whiteside

'Directions in Human Factors for Interactive Systems' is an attractive title, but unfortunately the book is riddled with faults. In the first place, it is made up of a series of articles which leaves it short on structure. Above all, however, it is often superficial and technically outdated.

Its superficial character is particularly apparent in the chapter on 'Formal specification and design'. This chapter begins with a plea for the use of formal methods, which is all to the good, but that's as far as it goes: The chapter comes to an end two pages later. What splendid formal methods are we talking about, how are they applied and who is already using them? Not a word. There isn't even an adequate bibliography; only two references (one to VDL and the other to CAM) are really relevant and there is nothing on VDM, for example, or HDM-Special, FDM, etc.

A large part of the book (almost a third) is devoted to designing and analysing the qualities of an 'annotated Pascal assistant', a kind of specialized text editor. This is a tool of the 'line by line' type, which hardly seems a field of enquiry appropriate to advanced research on 'human factors'. The authors' main contention is that interaction with computing systems must use natural language as far as possible. Shneiderman, as we have seen, was more reserved on this point. In the case of the 'Pascal assistant', the result of the design proposed is a system which appears cumbersome and verbose. No-one, in my view, would prefer a tool of this kind to a good 'full page' text editor or to the word processing systems available today on microcomputers, such as Microsoft WordStar (this column makes no plugs). Nor is the 'Assistant' particularly suitable for Pascal; it is not a 'structures editor' like Mentor, Gandalf or CPS, tools which the authors fail to mention.

In order to back up their theory, the authors give the results of tests during which different subjects used either a conventional text editor with unusual conventions for its command syntax, or another editor providing the same functions but with a syntax more like English (e.g. CHANGE ALL 'BOY' TO 'GARCON', etc.). The tests show that the latter scores better from every point of view, but I cannot help feeling that these results are of only limited value: the subjects started with no familiarity with either editor, and it is obvious that novices will find a syntax similar to natural language easier to master. The comparison would not necessarily give the same results after six months' experience. But, above all, I do not think that the difference between:

DELETE 8 (formulation required by the 'conventional' editor)

and

DELETE 8 lines (formulation required by the 'English' editor)

is earth shattering. It would have been more interesting to compare a line based editor with a full page editor, to find out whether selection from a menu is better than the use of explicit commands, to study the various pointing methods (cursor, mouse, etc.) and to measure the impact of an environment with several windows in overlay (an idea defended with enthusiasm by many present-day designers of programming environments, but whose real merits have not, as far as I know, been systematically evaluated; perhaps a reader will correct me).

The last two chapters cover, respectively, 'ten suggestions' for experiments relating to human factors, and advice on setting up experiments of this type.

Ledgard et al. give this book as a monograph in a series devoted, in principle, to the rapid publication of results in non-definitive form. Even presented in this way, however, I do not feel that it has much to contribute.

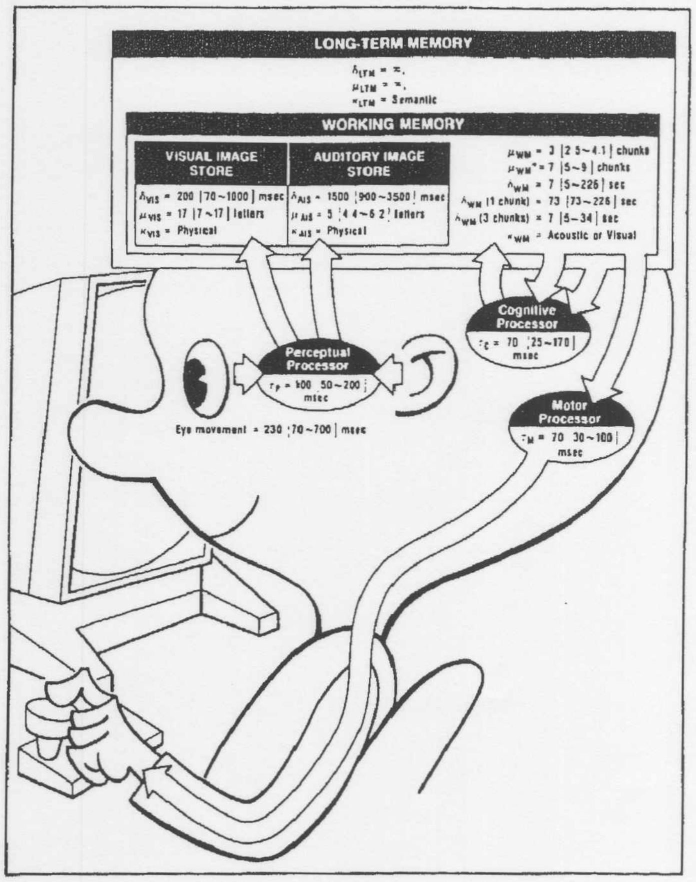
3. Card, Moran and Newell

'The Psychology of Human-Computer Interaction' is very different from the other two books. Card, Moran and Newell put forward a sort of treatise on the application of psychological research to man-machine interaction. The work (466 pages) is peppered with figures, graphs and references but is still a pleasure to read. I must admit that I found this book fascinating, overwhelming even, although I am not yet sure what to do with the wealth of material it contains.

Card et al. are, it could be said, 'modellers of cognitive activity'. Their aim is to construct adequate descriptions of what is going on in the human brain when it is engaging in certain activities, particularly, of course, man-machine communication. Models constructed in this way, and duly validated by experiments should, in turn, guide the designers of new systems.

The basic model is reproduced here (it is in figure 2.1 of the work). Tremble humanists! Come back, Descartes! Those of us who have always been bothered by the C in the acronym of the Association which publishes the French version of this journal (Afcet) can draw comfort here: for it is indeed cybernetics, dear to our grandparents, that we are talking about, even though the term never appears in the book. So, when facing a computer, each of us has a 'perception processor' with its aural and visual memories; a 'movement processor'; and a 'knowledge processor' which shares a primary ('short term') memory and a secondary ('long term') memory with the movement processor. The three can of course function in pipeline; their physical characteristics, given in the figure, show that we come nowhere near the standard of a Vax or an Apple (T is the cycle time of a processor, δ the access time to an element in the memory, μ the capacity and K the coding type).

Simple as it is, this model provides the answers to a number of problems which we shall ask the reader to solve (answers are given after the references). In a video game, a ball collides with another ball; how much time is available after the collision for calculating the initial displacement of the second ball if it is regarded as being caused by the


LONG-TERM MEMORY
$\delta_{LTM} = \infty$,
$\mu_{LTM} = \infty$,
κ_{LTM} = Semantic
WORKING MEMORY
VISUAL IMAGE STORE
δ_{VIS} = 200 [70~1000] msec
μ_{VIS} = 17 [7~17] letters
κ_{VIS} = Physical
AUDITORY IMAGE STORE
δ_{AIS} = 1500 [900~3500] msec
μ_{AIS} = 5 [4.4~6.2] letters
κ_{AIS} = Physical
μ_{WM} = 3 [2.5~4.1] chunks
μ_{WM}^* = 7 [5~9] chunks
δ_{WM} = 7 [5~226] sec
δ_{WM} (1 chunk) = 73 [73~226] sec
δ_{WM} (3 chunks) = 7 [5~34] sec
κ_{WM} = Acoustic or Visual
Perceptual Processor
τ_P = 100 [50~200] msec
Eye movement = 230 [70~700] msec
Cognitive Processor
τ_C = 70 [25~170] msec
Motor Processor
τ_M = 70 [30~100] msec

impact? At what speed must one read? How much time will be gained per operation by making the numeric keys more similar to the 'change function' keys of a calculator? And other questions besides.

Armed with this model, which is of course discussed and justified in detail, the authors study what happens during a session with an interactive system. Their analysis is pitched at four different levels of detail (task to be accomplished, basic function, command argument, keystroke); this division into several levels enables a more refined study to be carried out than in the article published by the same authors a few years ago [Card 80]. One of the important contributions of the work is a formalism of the algorithmic type (which, unfortunately, does not conform syntactically to the conventions of the most common programming languages), to describe the strategies applied consciously or unconsciously by the users of an interactive program, such as a text editor, for example. This description itself has several levels: Goals, Operators, Methods and Selection Rules (GOMS). If we take a low level example (selection rules), the following rule describes the strategy observed in the case of a number of users of a full page text editor using a mouse to designate an object to which a certain operation is to be applied (elimination, modification, etc.):

```
Selection rules for GOAL; POINT-TO-TARGET
    CHAR-POINT-RULE =
        if VisualSearchTarget isa # CHARACTER
        then CHOOSE (CHAR-POINT-METHOD)
    WORD-POINT-RULE =
        if VisualSearchTarget isa # WORD
        then CHOOSE (WORD-POINT-METHOD)
```

```
TEXT-SEG-RULE =
    if VisualSearchTarget isa # TEXT-SEG
    then CHOOSE (TEXT-SEG-POINT-METHOD)
```

The very detailed studies contained in this work refer to subjects as diverse as the comparison of different types of text editor, the design of integrated circuits, the design of a phototypesetting system, the analysis of various devices for on-screen selection, etc.

All this is remarkably well documented. A whole area of research in 'cognitive psychology' is summarized here for the benefit of computer scientists, who generally know nothing at all about this field (with the exception of some specialists in artificial intelligence who are concerned with knowledge models on which they can base their programs). In the view of the authors,this book will therefore be most useful to computer scientists, although it is also intended for psychologists. The authors maintain that the application of psychological principles is an issue for the former, rather than the latter. Their argument is that it is scarcely realistic to imagine a situation where psychologists and ergonomists could exercise real power over the way in which computing systems are designed and implemented: in practice, psychological and ergonomic criteria inevitably conflict with the other criteria governing the design of a software product (cost, efficiency, security, etc.) and it is the computer science engineer who must make the necessary compromises and who therefore requires access to the necessary information. Card, Moran and Newell therefore argue for the inclusion of ergonomics courses in software engineering teaching programmes, rather than for the training of psychologists and ergonomists as computer specialists.

Although I was very impressed by all this, I have two serious reservations. First of all, I did not like the way the emphasis was placed almost exclusively on two factors, time and quantity of information. The authors sometimes sound like time and motion engineers, or like a foreman standing over a worker with a stopwatch in his hand. Undoubtedly, a good system must be able to do simple things quickly; but what about the other qualities of interactive software, such as ease of use and learning, user friendliness, etc.? Can they all be reduced to measurements of time and volume? Can we not find some more subtle criteria? As I write this, I know that I am partly contradicting myself, because previously I criticized Shneiderman and the research he quotes for its lack of rigour and precision. But is it not possible to find measurements which, without lapsing into vagueness and subjectivity, describe something other than the time required for each operation and the number of elements handled?

My other reservation is associated with the form of the book, which is a research work. It appears to me that a designer of interactive systems cannot fail to be interested in this book and to feel that he still has much to learn; but much still remains to be done to put all this material into practice and to ensure that it really influences everyday design and implementation work. Alongside this book by Card, Moran and Newell, which I think will become a basic reference work, we need a more applied manual containing useful advice for the practitioner.

Nevertheless, I cannot recommend this work too highly to all those who are involved with interactive systems and have to construct them, whether the systems are text editors, operating systems, software engineering tools, CAD or CAM programs, etc.

References

[Gannon 77] J.D. GANNON: *An experimental evaluation of data type conventions;* Communications of the ACM, **20** (8), 584-595, Aug. 1977.

[Hansen 71] W.J. HANSEN: *User engineering principles for interactive systems;* Proceedings of the Fall Joint Computer Conference, **39,** 423-532, 1971.

[Shneidermann 83] B. SHNEIDERMANN: *Direct manipulation: a step beyond programming languages;* Computer (IEEE), **16** (8), 57-69, Aug. 1983.

[Weinberg 71] G.M. WEINBERG: The Psychology of Computer Programming; 1971, Van Nostrand-Reinhold, New York.

Answers to questions in section 3

100 milliseconds
652 words/minute
90 milliseconds

LE MICROPROCESSEUR 16 BITS 8086 (The 8086 16-bit microprocessor), by A.B. Fontaine. *Masson,* Paris, France, 1983, 200 pp., ISBN 2-225-79-960-1.

Don't attach too much importance to the title of this book: after reading it you will not have specialist (hardware or software) knowledge of the 8086. As A.B. Fontaine states in his introduction, the book makes reference to the characteristics of 16-bit microprocessors, and particularly to those of the Intel 8086, to introduce specialists in microcomputing to new computing concepts (all of which 16- and 32-bit microcomputers have made easier to implement).

The first two lessons (the book is organized in six relatively independent lessons) are concerned with the special characteristics of the 8086:

(a) separation, at hardware level, between logical and physical addresses;

(b) high-performance indexed addressing;

(c) the possibility of bus blocking during the execution of an instruction in multiprocessor systems;

(d) a very powerful instruction set which, however, is insufficiently standardized at register level.

These then link up with the new concepts introduced in the following lessons:

(a) multiprogramming and virtual memory (relocatable programs);

(b) data structures of high level languages like Pascal (easier implementation);

(c) multi-task operating systems (flag management);

(d) use of algorithmic languages (easier translation).