The future of interactive systems and the emergence of direct manipulation

BEN SHNEIDERMAN

BEHAVIOUR AND INFORMATION TECHNOLOGY, 1982, VOL. 1, NO. 3, 237–256

The future of interactive systems and the emergence of direct manipulation†

BEN SHNEIDERMAN

Department of Computer Science, University of Maryland, College Park, Maryland 20742, U.S.A.

Abstract. This paper suggests three motivations for the strong interest in human factors' aspects of user interfaces and reviews five design issues: command language versus menu selection, response time and display rates, wording of system messages, on-line tutorials, explanations and help messages and hardware devices. Five methods and tools for system development are considered: participatory design, specification methods, software implementation tools, pilot studies and acceptance tests and evolutionary refinement based on user feedback.

The final portion of the paper presents direct manipulation, an approach which promises to become widely used in interactive systems. Direct manipulation involves representation of the object of interest, rapid incremental reversible actions and physical action instead of complex syntax.

1. Introduction

The rapid metamorphosis of large computers into small numerous distributed appliances was stimulated by cost reductions resulting from dramatic technological advances. Large-scale integrated circuitry reduced the cost of computation a millionfold during the last three decades while reliability and speed increased. Now that multiple vendors provide equivalent system functionality, the challenge of harnessing these computational resources has moved to the human factors arena. Manufacturers who have been able to sell poor designs by offering sheer power will find their products losing to those who have become sensitive to the needs of the human user.

Three forces underly the strong industry concern for human factors. The first is the expansion of the user population to include novice and non-technically-trained people. In the first decades of computer software development, senior programmers designed text editors, operating system control languages, programming languages and applications' packages for themselves and their peers. Now the user population for office automation, home and personal computing and point of sales terminals is so vastly different that the experience and intuition of senior programmers may be inappropriate. Designs must be validated through pilot and acceptence tests which can also provide a finer understanding of user skills and capabilities. The egocentric style of the past must yield to humility and a genuine desire to accommodate to the user's skills, wishes and orientation. Designers must seek more direct interaction with the users during the design phase, development process and throughout the system lifecycle. Corporate marketing departments are aware of these issues and are a source of constructive encouragement. When more than 200 suppliers provide similar word processing packages, human engineering is vital for product acceptance.

† This paper was originally presented as the Keynote Address at the N.Y.U. Symposium on User Interfaces, 26–28 May, 1982, New York, U.S.A. It will be published in *Human Factors and Interactive Computer Systems*, 1983, edited by Y. Vassiliou, Ablex Publishing Co., Norwood, New Jersey, U.S.A.

The second force which promotes human factors' issues is the increasing organizational dependence on interactive systems. While automation of payroll or accounting functions was originally promoted as a cost-beneficial improvement, contemporary interactive systems are central to the functioning of many organizations. Hotel, airline or car rental reservations systems, banking applications, mail order inventory, information retrieval services and decision support systems are often critical determinants of an organization's success and even survival. In such a demanding environment, frequent errors, poor response time, slow performance or operator confusion, which might have been tolerated in the past, are no longer acceptable. The Quality Inn motel chain, which recently invested heavily in developing an improved on-line reservation system, estimated that each second trimmed from the 150 s average for a reservation saved $40 000 per year.

Further support for the need to improve the human interface comes from several studies which indicate poor user performance. Even expert users of interactive editing or command languages were found spending one-third of all commands in making or correcting errors. The opportunities for improved performance and increased productivity are clear. For high-volume applications a 10 per cent reduction in time per transaction can lead to a 10 per cent reduction in staff, terminals and hardware.

The third source of pressure is the high performance requirements in life-critical applications such as medical intensive care, nuclear reactor, air traffic or utility control, fire or police dispatch, manned spaceflight and military command and control. In these applications error rates must be kept extremely low and emergency conditions eliminate the luxury of consulting manuals or expert users. Designers are interested in strategies for presenting complex information rapidly and clearly, and approaches to command languages that eliminate the need for tricky formats which cannot be recalled during the once a year emergency. Manufacturers of these life-critical systems know that they are liable for damages if the design is demonstrated to be a contributing factor to a disaster.

IBM and AT&T have several hundred professionals working on human factors issues and a major commitment to this topic. Other computer manufacturers and large user organizations are trying to increase their attention to this area because they recognize the need and potential for improvement. The academic community and professional societies are struggling to respond to the rapid growth of interest.

It is impossible to predict what novel ideas or unifying theories will appear, but it is possible to lay out apparently productive directions for research and development. Section 2 covers a set of research issues which are of interest to practitioners in shaping future designs. Section 3 reviews methods and tools for system development. Section 4 is a more detailed presentation of a basic and promising design strategy which I call direct manipulation.

This author believes that the route to rapid progress is through controlled psychologically-oriented experimentation (Shneiderman 1980) by academic and industrial researchers and through pilot and acceptance tests by commercial system developments. This disciplined approach complements and supports the intuition and experience of designers while replacing the vehement, but often empty, arguments over the 'user friendliness' of competing systems. Scientific measurement of learning time, performance speed, error rates, user satisfaction and retention provides valuable data for making design decisions, choosing among competing products and formulating integrative theories which can be used for prediction in novel circumstances.

The future of interactive systems will be shaped on how rapidly experimentation proceeds and how eagerly designers apply the resultant guidelines. There is hope that many poor designs will be eliminated and that the increasingly discriminating user population will exert pressure for higher quality.

2. Design issues

In my study of text and display editors, bibliographic retrieval systems, database query facilities, job control languages, graphics systems, computer-based education, business applications and other systems, I have taken on the role of a movie critic of interactive systems. As I return to a system after several weeks, I find my sensitivity to design issues has become heightened. My progression is from tolerance, to annoyance at inconveniences or poorly designed features, to clarity about the source of my discomfort, and finally to hypotheses about how to improve the user interface. Of course, the critic's role is easier than the producer's, but the effective critic provides insights for future work, the provocation to do better next time and a clearer focus on the issues. This part covers several design issues which have arisen in multiple-application domains and should be of general interest.

2.1. *Command language versus menu selection*

The command language and menu selection approaches are distinguished by the demands they place on the user. The explicit list of options in menu selection eliminates the need for memorization and provides the operator with a clear indication of permissible actions. Well-written menus offer novice users familiar terminology and a step-by-step process for retrieving information or specifying procedures. Display rates play a key role in performance speed and user satisfaction with menu systems, since annoyance can build if the user must wait for lengthy menus. Users should be able to enter their choice before the menu is completely displayed and short cuts should be permitted for knowledgeable users. An appealing strategy is to number the menu choices, but provide alphanumeric names for each menu, allowing knowledgeable users to immediately specify the target menu. Experimental studies are beginnning to provide guidance for designers (Miller 1981, Dunsmore 1980) and commercial use of videotex systems is providing further feedback.

By contrast, command languages require users to memorize the options and to formulate possibly complex requests. Knowledgeable and frequent users prefer command languages because in many cases they permit faster task completion. Unfortunately, infrequent or novice users may find it difficult to maintain the permissible syntax in their memory, and it is difficult for designers to provide specific error messages when problems arise. Macrodefinition facilities are an important feature for expert users who prefer to define their own commands, and personalize their environment by encapsulating frequently-used command sequences in a new command. Issues for command language designers are hierarchical structure, congruence of command names, uniformity of syntax and abbreviations (Ledgard *et al.* 1980, Carroll 1980, Ehrenreich and Porcu 1982, Schneider 1982).

2.2. *Response time and display rates*

The speed of a computer system is a function of the response time, the number of seconds from the time the operator sends a command until the system begins displaying a response, and the display rate, the speed with which characters or graphics appear on a screen or hardcopy device. Miller's (1968) classic paper provided worth-while intuitive guidance which is being validated and refined by recent experimental studies (Goodman and Spence 1978, 1981, Bergman *et al.* 1981, Barber and Lucas 1982). An appropriate set of guidelines might be:

(1) Typing and cursor motion commands should generate results in 0·1 s.
(2) Frequent simple commands should take less than a second.
(3) Other commands may take longer, but the response time for similar commands should have a small range, say 20 per cent deviation from the mean. This gives users the sense of predictability, helps sensitize them to anaomalous conditions and permits planning of a sequence of commands. Users will modify, if possible, their command sequences to avoid slow operations.
(4) As the response time shortens, users pick up the pace of interaction and may make hasty decisions or learn more poorly. Error rates increase as the response time becomes very short. Faster is not always better, especially for novice users who may prefer slower operation.

Display rates also influence user performance and acceptance.

(1) Faster display rates are generally preferred if the material is familiar or if only a small part of the display must be read. Novices have more accurate performance and higher satisfaction with slower display rates (15–25 characters/s (Bevan 1981).
(2) Erratic display rates can be disturbing (Miller 1977).
(3) If the full text of the material must be read then display rates faster than normal reading speeds are of no advantage (Miller 1977).

2.3. *Wording of system messages*

In my use of and studies with users of interactive computer systems, I have become increasingly aware of the importance of system messages. Novice users are unimpressed with CPU speeds, disk storage capabilities, or elegant file structures. For them, the system appears only in the form of the messages on their screens or printers. When novices encounter violent messages such as 'FATAL, ERROR, RUN ABORTED', vague phrases such as 'ILLEGAL CMD', or obscure codes such as 'OC7' or 'IEH2191' they are understandably shaken, confused, dismayed and discouraged from continuing. The negative image that computer systems generate in many people is, I believe, largely due to the difficulties they have when they make mistakes or when they are unclear about what to do next.

There are many kinds of messages that should come under close scrutiny during the design process: menu-selection choices, prompts for command language or data entry, feedback indicating task completion, results from database searches, explanatory or tutorial information and error messages.

Since the error messages come at a moment of confusion or incomplete knowledge they have a more powerful impact. Experience and five experiments (Shneiderman 1982 b) have led me to these guidelines:

(1) Use positive tone indicating what must be done, rather than condemning the user for the error. Reduce or eliminate the use of terms such as 'ILLEGAL', 'INVALID', 'ERROR', or 'INCORRECT'. Instead of 'ILLEGAL PASSWORD' try 'Your password did not match the stored password. Please try again'.

(2) Be specific and choose terminology from the user's problem domain. Avoid the vague 'SYNTAX ERROR' or obscure internal codes. Use variable names and concepts known to the user. Instead of 'INVALID DATA' in an inventory application, try 'Dress sizes range from 5 to 16'.
(3) Place the user in control of the situation and provide enough information for the user to take action. Instead of the domineering 'ENTER NEXT REQUEST', try 'ready for next command'. Instead of 'INCORRECT COMMAND', try 'Permissible commands are: SAVE, LOAD, or EXPLAIN'.
(4) Have a neat, consistent and comprehensible format. Avoid lengthy numeric codes, obscure mnemonics and cluttered displays.

Writing good messages, like writing poems, essays or advertisements, takes experience, practice and a sensitivity to how the reader will react. It is a skill which can be acquired and refined by programmers and designers who are intent on serving the user. Yet, perfection is not possible and humility is the mark of the true professional. Therefore, system messages should be subjected to an acceptance test with an appropriate user community to assure comprehensibility.

2.4. *On-line tutorials, explanations and messages*

As computer use shifts from the expert to the novice and to the knowledgeable infrequent user, the need for on-line aids increases. Interrupting a terminal session to retrieve a manual and locate relevant material is so disruptive that it discourages use. Designers of many systems have sought to provide sufficient tutorial and reference material on-line so that printed manuals are unnecessary. This philosophy has merit, since the on-line material can focus on the current user's task, can be updated regularly, may be less difficult to locate and avoids the cost and delay of printing and shipping.

These advantages are appealing, but they can be compromised if the quality of the on-line material is poor, if the extra commands to access them increase confusion and if jumping from work to explanatory screens becomes disruptive. Both Relles (1979) and Dunsmore (1980), to their surprise, found that the on-line aid treatment in their experiment led to poorer performance than printed manuals. Apparently the switching of contexts and the additional commands disrupted novice users. Having a printed manual in hand while looking at a computer display is useful because the problematic terminology or task remains visible while the explanation is read. Many users of on-line aid facilities will make lengthy written notes about proper command forms before returning to the task.

An effective strategy with screen displays is to have a four or six-line window, so that helpful instruction can be presented while the task remains visible. A more ambitious alternative is to have a second screen for helpful information.

Just because on-line aids are available, does not ensure that they are effective. Too often, informally written screens attempt to serve all levels of users and confuse novices with complex terminology or annoy experts with lengthy tutorials. A well-designed on-line aid should provide separate tutorials for novices, command explanations for knowledgeable intermittent users and brief messages about specific problematic situations.

Screen formats should be uniform and orderly so that users will know where to look for specific information and can use positional cues when scanning for previously retrieved information. Graphic designers, book or advertising copy writers and technical writers should be included in developing extensive on-line aids. Pilot studies and acceptance tests should focus on the effectiveness of on-line aids by determining whether subjects can satisfy established criteria for learning and error recovery. A reasonable goal might be for 80 per cent of the selected subjects to accomplish a benchmark set of tasks within 30 min.

An enduring question is what to call these on-line aids? Since the term 'HELP' may have negative connotations for some users, I prefer the generic term 'EXPLAIN' for a keyboard label or command name.

2.5. *Hardware devices*

The profusion of hardware options is an inviting challenge and an overwhelming burden to designers. The hundreds of variations in the basic items such as keyboards, screens and hard copy printers must be reviewed for suitability. Then the numerous additional devices such as light pens, touch screens, graphics tablets, joysticks, programmed function keys and rotating knobs must be considered.

However, since new devices are appearing on the market regularly and since system portability is often important, a modular approach which avoids dependence on a specific device is appropriate. Often the hardware is the softest part of the system design. Foley and Wallace (1974) describe four logical device types which might be the basis for design:

(1) Pick: a mechanism for picking from a set of displayed entities.
(2) Valuator: a device for setting numeric values, e.g. a rotating knob with a potentionmeter, or keyboard entry of the value.
(3) Locator: a way of specifying a position in two- or three-space, e.g. a touchscreen, graphics tablet or arrow keys.
(4) Button: a selection device for initiating or terminating action.

By designing in a device-independent manner, new approaches or hardware can be easily incorporated.

Some designers are tempted to include novel devices, but fail to realize their additional complexity. Novel devices are attractive to users, but the interest wears off quickly unless the device is a genuine improvement. Experimentation with alternate devices can reveal their utility for specific tasks and users. A danger with multiple devices is the additional effort in switching from one device to another. Reaching up from a keyboard to a touchscreen is disruptive since the user's focus of attention must shift and then return to the keyboard. All else being equal, designers should limit the number of context shifts necessary.

3. Methods and tools

The issues raised in § 2 should be the subject of controlled experimentation so that practical guidelines can be uncovered. Implementers of interactive systems will need such guidelines, but the implementers must also deal with the process of system development. These lifecycle processes, methods and tools are increasingly the subject of research and exploration.

3.1. *Participatory design*

Many authors have urged participatory-design strategies (for a review see, for example, Olson and Ives 1981, Ives and Olson 1981), but their positive impact is not validated. The arguments in favour of user involvement include more accurate information about task performance, an opportunity to argue over design decisions, the sense of participation which builds ego investment in successful implementation and the potential for increased user acceptance of the final system.

On the other hand, extensive user involvement may be more costly and lengthen the implementation period, built antagonism with those who are not involved or whose suggestions are rejected, force designers to compromise their design to satisfy incompetent participants and simply build oposition to implementation (Ives and Olson 1981).

The implementation of a complex information system is beyond study by controlled experimentation or survey. The sensitive project leader must judge each case on its merits and decide on the right level of user involvement. The personalities of the design team and the users are such a critical determinant that experts in group dynamics and social psychology may be useful as consultants.

The experienced project leader knows that organizational politics and the preferences of individuals are more important than the technical issues in governing the success of an interactive system. The warehouse manager who sess his/her position threatened by an interactive system which provides senior managers with up-to-date information through their terminals, will ensure that the system fails by delaying data entry or being less than diligent in guaranteeing data accuracy. Proper system design should taken into account the impact on users and solicit their participation to ensure that all concerns are made explicit early enough to avoid 'counterimplementation' (Keen 1981).

3.2. *Specification methods*

One of the difficuties in gaining user involvement during system design is that it is hard to convey what the final system will look like. It is even difficult for designers to discuss among themselves or to write down what the system will look like when it is done. Augmented transition networks have been proposed by several people as a notation for system description (Wasserman and Stinson 1979, Feyock 1977) but this graphic notation can become cluttered and requires transformation to be put in machine-readable form. Reisner (1981) proposed a formal action grammar to describe general user activity with an interactive system.

Expanding from her concept, Shneiderman (1982 a) offers a multiparty grammar which describes the action of the operator and the response of the computer with the same notation. This Backus Naur Form (BNF) extension has labelled non-terminal symbols to represent either the user or the computer. The productions describing human commands are used to parse the input string and cause the value for the last parse to be assigned to the non-terminal, as is done in syntax-directed translation systems. Then the value of the non-terminal can be used by the computer to generate a response, for example:

```
<SESSION> ::=<H: HUMAN COMMAND> <C: COMPUTER RESPONSE>
<H: HUMAN COMMAND> ::=DELETE <H: FILENAME>
<H: FILENAME> ::=IDENTIFIER
<C: COMPUTER RESPONSE> ::=FILE [<H: FILENAME>] HAS
                               BEEN DELETED
```

The first production defines a session fragment as a human command (the 'H' indicates human action and the 'C' computer action) followed by the computer's response. The second production defines a simple delete command where a filename is specified. The third production shows that the filename is to be in the form of a previously defined systems identifier. When the human types the filename the non-terminal symbol ⟨H: FILENAME⟩ acquires as its value the filename. Then the computer's response shown in the fourth production can take the value of the filename to generate an appropriate message.

Multiparty grammars can contain features to deal with unparsable input, screen formatting and dynamic actions. Using a BNF-like strategy means that interaction design must proceed in an orderly top-down tree-like manner with names for non-terminals to guide the reader. The specification can then be partially tested and implemented with compiler–compiler (also called translator writing systems) techniques.

This multiparty grammar approach is appealing for system implementers who are likely to be familiar with BNF productions, but non-programmer users and managers need a more comprehensible form. Some screen management systems allow construction of screens with display editors to generate a mock-up of the final system. This approach is especially appropriate in designing menu selection or on-line tutorial systems. Sample terminal sessions or output formats are also useful in specifying the system and giving users an idea of what is being planned.

3.3. *Software implementation tools*

Since standard high-level programming languages have meagre facilities for creating interactive systems, the code is often difficult to compose, comprehend, debug and modify. Special purpose languages exist for creating interactions in a computer-assisted instruction environment, but general implementation tools would be an improvement. Compiler–compilers appear to be useful as a basis for developing interactive system operators. Extending screen managers or display editors is another fruitful direction. The designer uses a display editor to produce a screen of information or menus and then specifies the network of screens which a user can traverse.

We have constructed a menu-builder/driver system to assist non-programmers in the creation of menu-selection-based information systems. Menus are constructed using the standard text editor and then stored on a direct-access file. The menu driver is a 600 line COBOL program which does all the traversal of the 'menuware', produces screen displays, handles user selections and generates simple error mesages. This same program can be used with the 100 screens in our department information system or with 50 000 menus. Decoupling the program from the data is a standard programming approach which allows non-programmers to prepare menu screens, simplifies maintenance of the code, permits portability of at least the menuware, and ensures consistency in the interaction. Developers at NASA (Helfer *et al.* 1981) and Bell Laboratories (Heffler 1981) have constructed similar systems.

3.4. *Pilot studies and acceptance tests*

Despite all the progress in understanding user interaction techniques it will always be necessary to validate new approaches by pilot studies and acceptance tests. Early prototypes tests can provide designers with the data to make design decisions or modifications before the cost and delay becomes prohibitive. Aircraft manufacturers build plywood mockups, orchestra conductors have many levels of rehearsals and architects produce multiple design drawings or models.

Similarily in designing interactive systems pilot studies are extremely beneficial, even if only part of the system can be tested. The first tests may be with typewritten versions of screen displays or paper and pencil attempts at writing command language statements. The early subjects will be the designers and their colleagues—it is surprising how much designers can learn by applying their novel interface syntax to a standard set of problems. The basic process of establishing a standard set of problems is often provocative. As system development progresses, pilot studies can be more rigorous and involve a wider class of subjects.

When the system is complete, a rigorous acceptance test should be carried out. Software developers expect to have their code validated against a benchmark set of inputs, so when a human interface is part of the system an acceptance test should be a natural process.

The acceptance test might specify, in a simple case, that 'after 75 min of training, 40 typical users should be able to accomplish 80 per cent of the benchmark tasks in 35 min with fewer than 12 errors'. Since most systems have multiple classes of users, multiple levels of commands, and a wide range of task difficulty, the acceptance test will consist of several subtests. Furthermore, additional constraints such as a lengthier training period, requirements for retention, or user satisfaction goals may inspire many acceptance-test variants.

The importance of precise acceptance tests is that they set clear goals for user performance which can have profound influences on design decisions. If an acceptance test is expected, then numerous pilot tests are likely to be carried out along the way to verify that the goal is attainable.

Critics may scoff at the very idea of acceptance tests or complain about the delays and expenses of such tests, but the savings in superior designs are extremely attractive. Just as rigorous clinical trials are required for pharmaceutical companies, I believe that common practice will be to require acceptance tests for interactive systems. Such testing will occur first in life-critical applications, but commercial purchasers will increasingly expect evidence of formal testing.

3.5. *Evolutionary refinement based on user feedback*

The complexity of interactive systems and the diversity of users mean that the initial implementation will have to be modified in response to user difficulties and suggested improvements. Knowledgeable designers recognize this evolutionary refinement process as an opportunity and seek out user feedback. They also design their systems in a modular way so that modifications can be easily implemented.

User feedback can come from direct discussions or interviews with users or their managers. Individuals or group discussions, on a regularly scheduled basis, give the users a chance to voice their concerns and make suggetions. If users come to expect such open forums, then they are more likely to make note of a problem they are having.

On-line user consultants are a wonderful source of assistance to the users and of information about user problems (Hiltz 1982). Electronic message exchange between users and consultants is an effective, personal and relatively inexpensive way to provide assistance and gauge performance. If on-line consultants are not a viable option, then users should be able to at least send messages which are regularly read by designers and maintainers.

Survey questionnaires can be useful in soliciting user satisfaction, but low return rates, which may bias the sample, are a problem. On-line questionnaires are an intriguing possibility which is now being tested by a consortium of library services. The questionnare can be lengthy or it may be a simple set of three questions asked just before logging off. If requiring every user to answer subjective satisfaction questions at every session seems too burdensome, than a sampling strategy could be tried. Regular use of evaluation scales can demonstrate improved satisfaction as system modifications are made or training is improved.

There is a good opportunity for researchers in psychological test scale construction to apply their skills in developing a standardized instrument for evaluting interactive systems. Semantic differential scales with a 1–7 range have been used in our studies (Shneiderman 1982 b) and by many others (Lyons 1980), but a uniform and validated set of test items would benefit developers and permit comparisons across systems.

The items on such a questionnaire would cover user satisfaction with such items as the legibility of the material, comprehensibility of the displays, understandability of the command language, clarity of the prompts or error messages, effectiveness of the system in providing the desired service, adequacy of the response time or display rates and desire to use the system again.

4. Direct manipulation

In talking to users of interactive systems, I have become aware of a pattern of excitement. Certain systems generate a glowing enthusiasm among users which is in marked contrast with the more common reaction of grudging acceptance or outright hostility. The enthusiastic users' reports are filled with the positive feelings of:

(1) Mastery of the system.
(2) Competence in performance of their task.
(3) Ease in learning to use the system originally and acquire new features.
(4) Confidence in their capacity to retain mastery over time.
(5) Enjoyment in using the system.
(6) Eagerness to show it off to novices.
(7) Desires to explore more powerful aspects of the system.

These feelings are not universal, but this amalgam is meant to convey an image of the truly pleased user. As I examined the systems used by these enthusiasts, I began to develop a model of the features which produced such delight. The central ideas seemed to be visibility of the object of interest, rapid reversible actions and replacement of complex command langage syntax by direct manipulation of the object of interest—hence the term 'direct manipulation'.

4.1. *Examples of direct manipulation*

No single system has all the attributes or design features that I admire—that may be impossible—but each has enough to win the enthusiastic support of many users.

4.1.1. *Display editors*

Users of full-page display editors are great advocates of their systems as compared with line-oriented text editors. A typical comment was 'Once you've used a display editor you will never want to go back to a line editor—you'll be spoiled'. Similar comments came from users of stand-alone word processors such as the WANG system or display editors such as EMACS on the MIT/Honeywell MULTICS system or 'vi' (for visual editor) on the UNIX system. A beaming advocate called EMACS 'the one true editor'.

Roberts (1979) found overall performance times with line-oriented editors were twice as long as with display editors. Training time with display editors is also reduced, so there is evidence to support the enthusiasm of display-editor devotees. Furthermore, office automation evaluations consistently favour full page editors for secretarial and executive use.

The advantanges of display editors include:

(1) Full display of a page of text in the form that it will appear when the final printing is done. This gives the user a clear sense of context for each paragraph, permits simpler reading and scanning of the document and reveals where page breaks or section headings will fall. By contrast the one line at a time view offered by line editors is like seeing the world through a narrow cardboard tube.

(2) Cursor action which is visible to the user. Seeing an arrow, underscore or blinking box on the screen gives the operator a clear sense of where to focus attention and apply action.

(3) Cursor motion through physically obvious and intuitively clear means. Arrow keys or a cursor motion devices such as a mouse, joystick or graphic tablet provide natural physical mechanisms for moving the cursor. This is in marked contrast to commands such as 'UP 6' which require an operator to convert the physical action into a correct syntactic form, which may be difficult to learn, hard to recall and a source of frustrating errors.

(4) Labelled buttons for action. Many display editors have buttons with commands etched onto them, such as INSERT, DELETE, CENTER, UNDERLINE, SUPERSCRIPT, BOLD or LOCATE. These buttons act as a permanent menu-selection display to remind the operator of the features and to avoid the need to memorize a complex command-language syntax. On some editors only 10 or 15 labelled buttons provide the basic functionality. A specially-marked button may be the gateway to the world of advanced or infrequently used features which are offered on the screen in menu form.

(5) Immediate display of the results of an action. When a button is pressed to move the cursor or centre text, the results are shown immediately on the screen. Deletions are immediately apparent since the character, word or line are erased and the remaining text is rearranged. Similarly insertions or text movements are shown after each keystroke or function button press. This is in contrast to line editors in which print or display commands must be issued to see the results of changes. In many line-oriented editors, commands to skip lines, centre headings, or skip pages must be included in the text body and the impact of these commands is not apparent until a final print command is issued and the entire document is printed. On many display editors, by contrast, the final document form can be easily seen by single button presses to turn pages.

(6) Rapid action and display. Most display editors operate at high speed: 120 characters/s (1200 baud), a full page in a second (9600 baud) or even faster. This high display rate coupled with short response time produces a thrilling sense of power and speed. Cursors can be moved quickly, large amounts of text can be scanned rapidly and the results of commands can be shown almost instantaneously. Line editors operating at 30 characters/s with 3–8 s response times seem dragged down in the mud.

(7) Easily reversible commands. When entering text an incorrect keystroke is repaired by merely backspacing and overstriking. Simple changes can be made moving the cursor to the problem area and overstriking, inserting or deleting characters, words or lines. Many display editors offer a simple 'UNDO'

command to return the text to its state before the previous command or command sequence. The easy reversibility reduces user anxiety about making a mistake or fear of destroying the file.

4.1.2. *VISICALC*

This innovative system was the product of a Harvard MBA student who was frustrated at the time necessary to carry out the multiple calculations in a graduate business course. He built an 'instantly calculating electronic worksheet' (as the user manual describes it) which permits computation and display of results across 254 rows and 63 columns. The worksheet can be programmed so that column 4 displays the sum of columns 1 through 3, then every time a value in the first three columns changes the fourth column changes as well. Complex dependencies among manufacturing costs, distribution costs, sales revenue, commissions and profits can be stored for multiple sales districts and months so that the impact of changes on profits can be immediately seen.

By simulating an accountant's spread sheet or worksheet, VISICALC makes it easy for novices to comprehend. This display of 20 rows and up to nine columns, with the provision for multiple windows, gives the user sufficient visibility for easy scanning of information and comprehension of relationships among entries. The command language for setting up the worksheet can be tricky for novices to learn and infrequent users to remember, but most users need learn only the basic commands. The distributor of VISICALC attributed its appeal to the fact that 'it jumps', meaning that the delight is in watching the propagation of changes across the screen.

VISICALC users can easily try out many managerial plans and rapidly see the impact on sales or profit. Changes to commissions or economic slowdowns can be quickly added to the worksheet. The current status of the worksheets can be saved for later review.

4.1.3. *Spatial-data management*

The developers of the prototype spatial-data management system (Herot 1980), attribute the basic idea to Nicholas Negroponte of MIT. In one scenario the user is seated before a colour graphics display of the world and can zoom in on the Pacific to see markers for military ship convoys. By moving a joystick the screen is filled with silhouettes of individual ships which can be zoomed in on to display detailed data or ultimately a full colour picture of the captain.

In another senario, icons representing different aspects of a corporation such as personnel, an organizational chart, travel information, production data or schedules are shown on a screen. By moving the joystick and zooming in on objects of interest the user is taken through complex 'information spaces' or 'I-spaces' to locate the item of interest. A building floor plan showing departments might be shown and when a department was chosen, individual offices become visible. On moving the cursor into a room details about that individual appear on the screen. If you choose the wrong room, merely back out and try another. The lost effort is minimal and there is no stigma of error.

The success of a spatial-data management system depends on the skill of the designers in choosing icons, graphical representations and data layouts which are natural and comprehensible to the user. The joy of zooming in and out or of gliding over data with a joystick entices even anxious users who quickly demand additional power and data.

4.1.4. *Video games*

Maybe the most exciting, well-engineered and certainly successful application of these concepts is in the world of video games. The early, but simple and successful, game called PONG required the user to rotate a knob which produced horizontal motion of a white rectangle on the screen. A white spot acted as a ping-pong ball which ricocheted off the wall and had to be hit back by the movable white rectangle. The user developed skill involving speed and accuracy in placement of the 'paddle' to keep the increasingly speedy ball from getting by, while the speaker emitted a ponging sound when the ball bounced. Watching someone else play for 30 s is all the training that is needed to become a competent novice, but many hours of practice are required to become a skilled expert.

Contemporary games such as Missile Command, Asteroids, Pac Man, Galaxyian, Centipede or Space Invaders are far more sophisticated in their rules, colour graphics and sound effects. The designers of these games provide stimulating entertainment, a challenge for novices and experts and many intriguing lessons in the human factors of interface design—somehow they have found a way to get people to put coins in the sides of computers. The strong attraction of these games is in marked contrast to the anxiety and resistance that many users have for office automation equipment.

These games provide a field of action which is simple to understand since it is an abstraction of reality—learning is by analogy. Watching a knowledgeable player for several minutes is sufficient to learn the basic principles, but there is ample complexity to entice coins from experts for many hours. The range of skill that is accommodated is admirable. The commands are physical actions such as button presses, joystick motions or knob rotations whose results are shown immediately on the screen. There is no syntax to remember and therefore no syntax-error messages. If users move their spaceships too far left, then they merely move back to the right. Error messages are unnecessary because the results of actions are obvious and can be easily reversed. These principles can be applied to office automation, personal computing or other interactive environments.

Other principles are interesting to note, but may not be applicable to non-game applications. Every game that I have seen keeps continuous score so that users can measure their progress and compete with their previous performance, with friends or with the highest scorers. Typically, the 10 highest scorers get to store their initials in the game for regular display. This is one form of positive reinforcement that encourages mastery. Most games are heavily dependent on hand–eye co-ordination, an approach which seems inconsistent with non-game applications. Finally, the random events which occur in most games is meant to challenge the user, but in non-game designs predictable system behaviour is preferred.

4.1.5. *Computer-aided design/manufacturing*

Many computer-aided desing systems for automobiles, electronic circuitry, architecture, aircraft or newspaper layout use principles of direct manipulation. The operator may see a circuit schematic on the screen and with light-pen touches can move resistors or capacitors into or out of the proposed circuit. When the design is complete the computer can provide information about current, voltage drops or fabrication costs and warnings about inconsistencies or manufacturing problems. Similarly newspaper layout artists or automobile-body designers can easily try multiple designs in minutes and record promising approaches until a better one is found.

The pleasure in using these systems stems from the capacity to manipulate the object of interest directly and to generate mutliple alternatives rapidly. Some systems have complex command languages, but others have moved to using cursor action and graphics-oriented commands.

Another related direction is the world of computer-aided manufacturing and process control. The Honeywell process-control system provides the oil refinery, paper mill or power-utility plant manager a schematic view of the plant. The colour schematic may be on eight displays with red lines indicating a sensor value which is out of normal range. By pressing a single numbered button (there are no commands to learn or remember), the operator can get a more detailed view of the troubling component and with a second press the operator moves down the tree structure to examine individual sensors or to reset valves and circuits.

A basic strategy for this design is that there are no complex commands to be recalled in once-a-year emergency conditions. The schematic of the plant facilitates problem solving by analogy since the linkage between real-world high temperatures or low pressures and screen representations is so close.

4.1.6. *Further examples*

My favourite example of direct manipulation is driving an automobile. The scene is directly visible through the front window and actions such as braking or steering have become common knowledge in our culture. To turn to the left, simply rotate the steering wheel to the left. The response is immediate and the scene changes, providing feedback to refine the turn. Imagine trying to turn by issuing a command 'LEFT 30 DEGREES' and then having to issue another command to see where you are now—but this is the level of operation of many office automation tools of today.

The term direct manipulation is most accurately applied to describe the programming of some industrial robot tools. The operator holds the robot 'hands' and guides it through a spray painting or welding task while the controlling computer records every action. The control computer can then operate the robot automatically and repeat the precise action whenever necessary.

A large part of the success and appeal of the query-by-example (Zloof 1975) approach to data manipulation is due to the direct representation of the relations on the screen. The user moves a cursor through the columns of the relational table and enters examples of what the result should look like. There are just a few single-letter keywords to supplement the direct manipulation style. Of course, complex booleans or mathematical operations require knowledge of syntactic forms. Still, the basic ideas and facilities in this language can be learned within half an hour by many non-programmers. Query-by-example succeeds because novices can begin working with just a little training, yet there is ample power for the expert. Directly manipulating the cursor across the relation skeleton is simple and showing the linking variable by giving an example is intuitively clear for someone who understands tabular data.

Designers of advanced office automation systems have made use of direct-manipulation principles. The Xerox Star (Smith *et al.* 1982) offers sophisticated text-formatting options, graphics, multiple fronts and a rapid, high resolution, cursor-based user interface. Researchers at IBM's Yorktown Heights Laboratories (Schild *et al.* 1980) propose a future office system, called PICTUREWORLD, in which graphic icons represent file cabinets, mailboxes, notebooks, phone messages, etc. The user could compose a memo with a display editor and then indicate distribution and filing operations by selecting from the menu of icons.

Hatfield (1981 and personal communication) has recognized many of these principles and included them in preliminary designs for an advanced office automation system. He describes the general approach as 'What you see is what you get'. Another imaginative observer of interactive system designs, Nelson (1980), perceives user excitement when the interface is constructed by what he calls the 'principle of virtuality'. Much credit also goes to the individual designers who have created systems which exemplify aspects of direct manipulation.

4.2. *Understanding direct manipulation*

The attraction of systems that use principles of direct manipulation is apparent in the enthusiasm of the users. The designers of the examples in §4.1 had an innovative inspiration and an intuitive grasp of what users would want. Each example has features that could be criticized, but it seems more productive to construct an integrated portrait of direct manipulation:

(1) Continuous representation of the object of interest.
(2) Physical actions or labelled button presses instead of complex syntax.
(3) Rapid incremental reversible operations whose impact on the object of interest is immediately visible.

Using these three principles it is possible to design systems which have these beneficial attributes:

(1) Novices can learn basic functionality quickly, usually through a demonstration by a more experienced user.
(2) Experts can work extremely rapidly to carry out a wide range of tasks, even defining new functions and features.
(3) Knowledgeable intermittent users can retain operational concepts.
(4) Error messages are rarely needed.
(5) Users can immediately see if their actions are furthering their goals, and if not, they can simply change the direction of their activity.
(6) Users have reduced anxiety because the system is comprehensible and because actions are so easily reversible.

My understanding of direct manipulation was facilitated by considering the syntactic/semantic model of user behaviour. The cognitive model was first developed in the context of programming-language experimentation (Shneiderman and Mayer 1979) and has been applied to database query language questions (Shneiderman 1981).

The basic idea is that there are two kinds of knowledge in long-term memory:

(1) *Syntatic knowledge:* the details of command syntax. In a text editor, syntactic knowledge would include permissible list delimiters, inserting a new line after the third line is 13 (or is it 31?), or the keystroke necessary for erasing a character. This knowledge is arbitrary and therefore acquired by rote memorization. Syntactic knowledge is volatile in memory and easily forgotten unless frequently used. This knowledge is system dependent with some possible overlap among systems.
(2) *Semantic knowledge:* the concepts or functionality. Semantic knowledge is hierarchically structured from low-level functions to higher-level concepts. In text editors, lower-level functions might be cursor movement, insertion,

deletion, changes, text copying, centring or indentation. These lower-level concepts are close to the syntax of the command language. A middle-level semantic concept for text editing might be the process for correcting a misspelling: produce a display of the misspelled word, move the cursor to the appropriate spot and issue the change command or key in the correct characters. A higher-level concept might be the process for moving a sentence from one paragraph to another: move the cursor to the beginning of the sentence, mark this position, move the cursor to the end of the sentence, mark this second position, copy the sentence to a buffer area, clean up the source paragraph, move the cursor to the target location, copy from the buffer, check that the target paragraph is satisfactory and clear the buffer area.

The higher-level concept in the problem domain (moving a sentence) are decomposed, by the expert user, in a top-down way into multiple lower-level concepts (move cursor, copy from buffer, etc.) closer to the program or syntax domain. Semantic knowledge is largely system independent; text-editing functions such as inserting/deleting lines, moving sentences, centring, indenting, etc., are generally available in text editors although the syntax varies. Semantic knowledge, which is acquired through general explanation, analogy and example, is easily anchored to familiar concepts and is therefore stable in memory.

The process of command formulation in the syntactic/semantic model is from the user's perception of the task in the high-level problem domain to the decomposition into multiple lower-level semantic operations and the conversion into a set of commands. The syntax of text editors may vary, but largely the decomposition from problem into low-level semantics is the same. At the level of syntax the user must recall whether spaces are permitted, whether program function keys are available, or whether command abbreviations are permitted. As a user of half a dozen text editors over a week, I am very aware of the commonality of my thought processes in problem solving and the diversity of syntactic forms that I must specify. Especially annoying are syntactic clashes such as the different placement of special characters on keyboards, the multiple approaches to backspacing (backspace key, CONTROL-H or printed rub-out characters) and the fact that one text editor uses 'K' for keeping a file while another uses 'K' for killing a file.

Novices begin with a close linkage between syntax and semantics; for them the command syntax is the focus of their attention as they seek to remember the command function and syntax. As they gain experience they increasingly think in higher-level semantic terms which are freer from syntactic detail. Novices may have a hard time figuring out how to move a sentence of text, even if they understand each of the commands. Novices using editors which have a 'CHANGE/old string/new string' command must still be taught how to use this command to delete a word or insert a word in a line.

The syntactic/semantic model suggests that training manuals should be written from the more familiar higher-level problem-domain viewpoint. The titles of sections should describe problem-domain operations that the user deals with regularly, and then the details of which commands are used to accomplish the task can be presented. Finally, the actual syntax is shown. Manuals which have alphabetically arranged sections devoted to each command are very difficult for the novice to learn from because it is difficult to anchor the material onto familiar concepts.

The success of direct manipulation is understandable in the context of the syntactic/semantic model. The object of interest is displayed so that actions are directly in the problem domain. There is little need for the decomposition into multiple commands with a complex syntactic form. On the contrary, each command produces a comprehensible action in the problem domain which is immediately visible. The closeness of the problem domain to the command syntax reduces operator problem-solving load and stress.

Dealing with representations of objects may be more 'natural' and closer to innate human capabilities: action and visual skills emerged far before language in human evolution. Psychologists have long known that spatial relationships and actions are more quickly grasped with visual rather than linguistic representations. Furthermore, intuition and discovery are often promoted by suitable visual representations of formal mathematical systems.

It is easy to envisage direct manipulation in cases where the physical action is confined to a small number of objects and simple commands, but complex applications may be unsuitable to this approach. On the other hand, display editors provide impressive functionality in a natural way. The limits of direct manipulation will be determined by the imagination and skill of the designer.

4.3. *Potential applications of direct manipulation*

The trick in creating a direct-manipulation system is to come up with an appropriate physical model of reality. For example, if the application is a personal- or business-address listing, then a display of a 'Rolodex'-like device seems natural. The most recently retrieved address card appears on the screen and the top line of the next two appear behind, followed by the image of a pack of remaining cards. As the joystick is pushed forward the Rolodex appears to rotate and successive cards appear in front. As the joystick is pushed further, the cards pass by more quickly or as the joystick is reversed, the direction of movement reverses. To change an entry, merely move the cursor over the field to be updated and type the correction. To delete an entry merely blank out the fields. Blank cards might be left at the top of the file, but when the fields are filled in, proper alphabetic placement is provided. To find all entries with a specific zip code, merely type the zip code in the proper field and enter a question mark.

Cheque-book maintenance and searching might be done in a similar fashion. Display a cheque-book register with labelled columns for cheque number, date, payee and amount. The joystick might be used to scan earlier entries, changes could be made in place, new entries would be made at the first blank line, and a tick could be made to indicate verification against a monthly report. Searches for a particular payee could be made by filling in a blank payee field and then typing a question mark.

Bibliographic searching has more elaborate requirements, but a basic system could be built by first showing the user a wall of labelled catalogue index drawers. A cursor in the shape of a human hand might be moved over to the section labelled 'Author Index' and to the drawer labelled 'F–L'. Depressing the button on the joystick or mouse would cause the drawer to open up and reveal an array of index cards with tabs offering a finer index. By moving the cursor-finger and depressing the selection button the actual index cards could be made to appear. Depressing the button while holding a card would cause a copy of the card to be made in the user's notebook, also represented on the screen. Entries in the notebook might be edited to create a printed bibliography or combined with other entries to perform set intersections or unions. Copies of entries could be stored on user files or transmitted to colleagues by electronic mail. It is easy to visualize many alternate approaches, so careful design and experimental testing will be necessary to sort out the successful comprehensible approaches from the idiosyncratic ones.

It is possible to apply direct manipulation to environments in which there is no obvious physical parallel to draw on. Imagine a job control language where the directory of files is shown continuously along with representations of computer components. Creating a new file is accomplished by merely typing the name of the new file in the first free spot in the directory listing. Deletion occurs when a file name is blanked out. Copies can be made by locking a cursor onto a file name and dragging it to a picture of a tape drive or a printer. For a hierarchical directory, the roots are displayed until a zoom command causes the next level of the tree to appear. With several presses of the button labelled 'ZOOM' a user should be able to find the right item in the directory, but if he/she goes down the wrong path, the 'UNZOOM' button will return to the previous level.

Why not do airline reservations by showing the user a map and prompting for cursor motion to the departing and arriving cities? Then use a calendar to select the date and a clock to indicate the time. Seat selection is done by showing the seating plan of the plane on the screen with a diagonal line to indicate an already reserved seat.

Why not do inventory by showing the aisles of the warehouse with the appropriate number of boxes on each shelf? Why not teach students about polynomial equations by letting them bend the curves and watch how the coefficients change or where the x-axis intersect occurs or how the derivative equation reacts?

The ideas are sketches for real systems. Competent designers and implementers must complete the sketches and fill in the details. Direct manipulation has the power to attract users because it is comprehensible, natural, rapid and even enjoyable. If actions are simple, reversibility ensured and retention easy, then anxiety recedes and satisfaction flows in.

5. Conclusion

The tremendous growth of interest in interactive system-design issues in the research community is encouraging. Similarly, the increased concern for improved human engineering in commercial products is a promising sign. Academic and industrial researchers are applying controlled psychologically-oriented experimentation to develop a finer understanding of human performance and to generate a set of practical guidelines. Commercial designers and implementers are eagerly awaiting improved guidelines and increasingly using pilot studies and acceptance tests to refine their designs.

Research efforts must be balanced between gaining an understanding of basic issues covered in § 2, the methods and tools described in § 3 and the innovative approaches such as direct manipulation discussed in § 4. The development of cognitive models of human behaviour is a basic research direction that will provide a framework for understanding and predicting human performance as well as for organizing principles for designers.

Acknowledgments

I am grateful to the Control Data Corporation for partial support (Grant 80 M15) of my work and to the University of Maryland Computer Science Center for computer resources to prepare this paper. I thank Jim Foley, John Gannon, John Lovgren, Sherry Weinberg and Mark, Weiser for their constructive and supportive comments on draft versions of this paper.

References

BARBER, R. E., and LUCAS, H. C., 1982, System response time, operator productivity and job satisfaction (to be published).

BERGMAN, H. BRINKMAN, A., and KOELEGA, H. S., 1981, System response time and problem solving behavior. *Proceedings of the Human Factors Society, 25th Annual Meeting*, Rochester, New York (12–16 October 1981), 749–753.

BEVAN, N., 1981, Is there an optimum speed for presenting text on a VDU? *International Journal Man–Machine Studies*, **14**, 59.

CARROLL, J. M., 1980, Learning, using, and designing command paradigms. IBM Research paper RC 8141, Yorktown Heights, New York.

DUNSMORE, H., 1980, Designing an interactive facility for non-programmers. *Proceedings of the ACM National Conference*, pp. 475–483.

EHRENREICH, S., and PORCU, T. A., 1982, Abbreviations for Automated Systems: Teaching Operators the Rules. In *Directions in Human-Computer Interaction*, edited by A. Badre and B. Shneiderman (Norwood: Ablex, New Jersey).

FEYCOCK, S., 1977, Transition diagram-based CAI/HELP systems. *International Journal of Man–Machine Studies*, **9**, 399.

FOLEY, J. D., and WALLACE, V. L., 1974, The art of graphic man–machine conversation. *Proceedings of the I.E.E.E.*, **62**, 4.

GOODMAN, T. J., and SPENCE, R., 1978, The effect of computer system response time on interactive computer aided problem solving. *ACM SIGGRAPH 1978 Conference Proceedings*, pp. 100–104; 1981, The effect of computer system response time variability on interactive graphical problem solving. *I.E.E.E. Transactions on Systems, Man and Cybernetics*, **11**.

HATFIELD, D. 1981, Lecture at Conference on Easier and More Productive Use of Computer Systems, Ann Arbor, Michigan.

HEFFLER, M. J., 1981, A human-computer interface that provides access to a diverse user community. *Proceedings of the 14th Hawaii International Conference on System Sciences*, Vol. 2, pp. 601–610.

HELFER, D. P., HOWELL, D. R., OWINGS, J. O., SZCZUR, M. R., and DALTON, J. T., 1981, A transportable executive for interactive applications. NASA/Goddard Space Flight Center, Greenbelt, Maryland.

HEROT, C. F., 1980, Spatial management of data, *ACM Transactions on Database Systems*, **5**, 493.

HILTZ, S. R., 1982, *On Line Scientific Communities: A Case Study of the Office of the Future* (Norwood, New Jersey: Ablex).

IVES, B. and OLSON, M. H., 1981, User involvement and MIS success: a review of research. Unpublished.

KEEN, P. G. W., 1981, Information systems and organizational change. *Communications of the ACM*, **24**, 24.

LEDGARD, H., WHITESIDE, J. A., SINGER, A., and SEYMOUR, W., 1980. The natural language of interactive systems. *Communications of the ACM*, **23**.

LYONS, M., 1980, Measuring user satisfaction: the semantic differential technique. *Proceedings of the 17th Annual Computer Personnel Research Conference*, ACM SIGCPR, pp. 79–87.

MILLER, D. P., 1981, The depth/breadth tradeoff in hierarchical computer menus. *Proceedings of the Human Factors Society 25th Annual Meeting* (12–16 October 1981) (Rochester, New York), pp. 296–300.

MILLER, L. H., 1977, A study in man–machine interaction. *Proceedings of the National Computer Conference*, **46** (Montvale, New Jersey: AFIPS Press), pp. 409–421.

MILLER, R. B., 1968, Response time in man–computer conversational transactions. *Proceedings Spring Joint Computer Conference 1968*, 33 (Montvale, New Jersey: AFIPS Press), pp. 267–277.

NELSON, E., 1980, Interactive systems and design of virtuality, *Creative Computing*, **6**.

OLSON, M. H., and IVES, B., 1981, User involvement in system design: an empirical test of alternative approaches. *Information and Management*, **4**, 183.

REISNER, P., 1981, Formal grammar and human factors design of an interactive graphics system, *Institute of Electrical and Electronic Engineers Transactions on Software Engineering*, **7**, 229.

RELLES, N., 1979, The design and implementation of user-oriented systems, Ph.D. Dissertation, University of Wisconsin Computer Sciences Technical Report No. 357.

ROBERTS, T. L., 1979, Evaluation of computer text editors. Report SSL-79-9, Xerox Palo Alto Research Center, Palo Alto, California.

SCHILD, W., POWER, L. R., and KARNAUGH, M., 1980, PICTUREWORLD: A concept for future office systems. IBM Research Report RC 8384, Yorktown Heights, New York.

SCHNEIDER, M. L., 1982, Models for the design of static software user assistance, In *Directions in Human–Computer Interaction*, edited by A. Badre and B. Schneiderman, Norwood, New Jersey: Ablex.

SHNEIDERMAN, B., 1980, *Software psychology: Human Factors in Computer and Information Systems* (Boston: Little, Brown and Co.); 1981, a note on human factors issues of natural language interaction with database systems. *Information Systems*, **6**, 125; 1982 a, Multiparty grammars and related features for defining interactive systems, *I.E.E.E. Systems, Man, and Cybernetics*, **12**; 1982 b, System message design: Guidelines and experimental results. In *Directions in Human-Computer Interaction* (edited by A. Badre and B. Shneiderman) (Norwood, New Jersey: Ablex).

SHNEIDERMAN, B., and MAYER, R., 1979, Syntactic/semantic interactions in programmer behavior: a model and experimental results. *International Journal of Computer and Information Sciences*, **7**, 219.

SMITH, C., IRBY, C., KIMBALL, R., VERPLANK, B., and HARSLEM, E., 1982, Designing the Star user interface. *BYTE*, **7**, 242.

WASSERMAN, A. I., and STINSON, S. K., 1929, Specification method for interactive information systems. *Proceedings of the Conference on Specifications of Reliable Software*, I.E.E.E. Catalogue No. CH 1401-9c.

ZLOOF, M. M., 1975, Query-by-example. *Proceedings of the National Computer Conference* (Montvale, New Jersey: AFIPS Press).