*Int. J. Man–Machine Studies* (1991) **35**, 275–289

# Display-based action at the user interface

STEPHEN J. PAYNE†

*Departments of Psychology and Computing, University of Lancaster, Lancaster LA1 4YF, UK*

*(Received 7 September 1989 and accepted in revised form 2 January 1990)*

This paper examines the hypothesis that information flow, from device to user, is a vital part of skilled activity in human–computer interaction. Two studies are reported. The first study questions users of keyboard-driven word processors about the effects of cursor-movement, finding and word-deletion commands in various contexts. The second study questions users of the Apple Macintosh-based systems, MacWrite and Microsoft Word, about the behaviour of the menu-driven find command. In both studies it is discovered that users often do not know the precise effects of frequently-used actions, such as the final position of the cursor, even though these effects are vital for future planning. It is concluded that even experienced users must acquire the information they need from the device's display during interactions, and that they do not necessarily remember regular details that are available in this way. This conclusion conflicts with those current models of user psychology that assume routine skill relies on complete mental specifications of methods for performing tasks.

## 1. Introduction

### 1.1. PLANNING AND INTERACTING

In the beginning there was the Plan. Miller, Galanter and Pribram (1960) announced the creation of cognitive science with the idea that bodily actions are mentally organized in advance through hierarchies of control. In the simplest model of planning and action, a goal is decomposed into subgoals, which in turn are decomposed into further subgoals, until these can be accomplished by "operators", which specify actions. When faced with a new goal, the agent must either create a plan of actions from scratch ("problem solving"), or select some pre-stored plan ("routine skill").

This simplest planning model provides the framework for theories of cognitive skill and its acquisition that can boast impressive achievements. The successive decomposition of goals into subgoals and operators is evidenced in problem solvers' verbal protocols (Newell & Simon, 1972), and important aspects of skill acquisition have been modelled by the transition from problem solving search to plan-maintenance and selection (e.g. Anderson, 1983). In the domain of HCI, the GOMS model (Card, Moran & Newell, 1983) relies on a model of expert performance within the simplest planning framework: goals are achieved by *selecting* a pre-compiled *method* (sequence of *operators*). This model has had considerable empirical success in the prediction of operating times (e.g. Card *et al.*, 1983), and, with additional assumptions, of learning and transfer times (e.g. Polson, 1987).

† Address correspondence and offprint requests to: School of Psychology, University of Wales College of Cardiff, PO Box 901, Cardiff CF1 3TG, UK.



0020-7373/91/090275 + 15$03.00/0 

An obvious omission from the simplest planning model is the importance of feedback. Miller *et al.*'s (1960) TOTE (Test Operate Test Exit) units insisted on the testing of feedback from actions against success criteria within the lowest-level units of control. Norman (1986) has imported this priority into his stage model of action, which he has applied to the analysis of user interfaces. In both these approaches, the role of feedback is constrained to checks on the success of operators or plans, but some important observations demonstrate that such a limited view underestimates the reactive nature of human action: its responsiveness to and reliance upon patterns in the external environment. Some crucial phenomena undermine the simplest top-down model of planning, and suggest that human action is, to an important degree, "display-based".

*1.1.1. The interleaving of planning and action*

The simplest planning model has it that plan-creation is an exclusively mental activity: goals are decomposed into sequences of operations by search in a mental problem space. Only when the sequence of operators has been computed does the planner need to act. However, people do not always work out complete plans in advance. Instead they may begin acting on the world immediately the first, or the first few actions are decided on. Such a strategy offers savings in working memory load, as long stacks of subgoals and intermediate problem states no longer need be maintained.

Limited pre-planning of this kind has been observed in human–computer interaction by Robertson and Black (1986), who found that users of a word processor begin by forming very short partial plans, as evinced by the distribution of pauses during task performance. Related findings, in the domain of program coding, are reported by Green, Bellamy and Parker (1987).

Many models of problem solving are silent about this issue, failing to distinguish between the application of operators in a problem space, and real action on the real world. Young and Simon (1987) discuss how limited pre-planning might be computer-modelled by coupling a planning module with an executor. If the executor is passed partial plans, and carried out those pieces which are enabled, then the remaining plan can be fleshed out in the context of the new state. A similar idea has been implemented by Larkin (1989), in a production system model called DIBS, for display-based problem solver. In DIBS the key idea is that the state of the external world contains much information about the current problem state, which can be exploited by the problem solver to obviate the need to maintain goal-stacks.

*1.1.2. Display-based control*

In many domains, plans are triggered not so much in service of goals as in response to patterns in the environment. Limited pre-planning means that operations within plans may be similarly display-cued. This phenomenon contributed to the rise of the production system as a model of human performance (e.g. Newell & Simon, 1972). The conditions of productions may specify either goals or display-properties, allowing goal-driven and data-driven control to be freely intermingled. Perhaps the fullest model of display-cueing is the "opportunistic planning" model of Hayes-Roth and Hayes-Roth (1979), which uses a blackboard architecture, very similar to the classical production system design, to model the seizing of plan-control by autonomous data-driven demons.

"Middle-period" production systems, like Anderson's (1983) ACT*, tended to eschew the data-driven side of production systems, preferring instead a control-regime based primarily around goal hierarchies. This is the scheme that has been imported into HCI modelling, by the GOMS model and Kieras and Polson's (1985) production system implementation. More recently, however, production systems like Larkin's (1989) DIBS and Anderson's own PUPS (Anderson, 1987), have reduced the role of goal-stacks, to allow reactive action back into the model.

These first two phenomena, the interleaving of planning and action, and display-based control, apply generally to human action in all kinds of context. Beyond them, using a device has some special features which introduce new roles for the display as a basis of action.

*1.1.3. Recognition of operators: display-based task-action mapping*

Two studies on device-users' memory for the user interface have exposed an important constraint on some device users' plans: the actions required to effect operators are apparently not remembered if instead they can be recognized on the user interface.

The first study by Morton (1967), was a large-scale survey of British telephone users' memory for the pattern of letters above the numbers that used to be a feature of British telephone dials. Morton discovered a "singular lack of incidental learning": none of more than 200 subjects, including 45 who frequently used the letters, could correctly recall the layout of letters on the dial. Recently a similar study by Mayes, Draper, McGregor and Oatley (1988) tested users' memory for the visual interface of the MacWrite word-processing system. Even everyday users of MacWrite showed poor recall for the names of menu-items, although they had no trouble selecting these items correctly in the service of tasks.

Howes and Payne (1990) discuss how findings like these can be modeled by extensions to the idea of a task-action grammar (Payne & Green, 1986). TAGs use semantic features of tasks to modify rewritings from tasks onto actions. In a D-TAG (display-based task-action grammar), actions can be specified as "the item on the display which best matches the current relevant task-features".

*1.1.4. Uncertainty about the effects of operators*

The three phenomena noted so far all point to the need to elaborate the simplest planning model, in various ways. None, however, justifies jettisoning the planning framework, and its impressive achievements. This conclusion differs from that of Suchman (1987), who advocates the total rejection of planning and cognitivism in favour of the descriptive approaches of ethnomethodology and conversation analysis.

To bolster her argument Suchman presents some observational work with novice users of a copier coupled to an intelligent plan-based help system. Suchman reports that the novices approach such a machine in a "try it and see" style, in which users are uncertain of the effects of the low-level actions themselves. Young and Simon (1987) also note the possibility of such an interactive style.

When the user is uncertain about the effects of operators, planning is clearly severely compromised. Pre-planning will be limited, not merely to save working memory resources, or as a convenience, but of logical necessity: interpretation of the operator effects will contribute to the setting of new subgoals and the choice of subsequent operators.

### 1.2. INTRODUCTION TO THE EMPIRICAL STUDIES

The studies reported here address the two device-specific phenomena of display-based action: recognition of operators, and uncertainty about operator effects. As yet, neither of these proposed "phenomena" has a great deal of empirical support.

Concerning the recognition of operators, there is a problem with the interpretation of both the Morton (1967) and the Mayes *et al.* (1988) studies. Failure to recall was noted in the absence of any task context: subjects were simply asked to recall the appearance of the device, cued, if at all, by visual, background features. Perhaps the names of menu-items would be cued by task-demands, so that operators could, after all, be recalled as well as recognized? The second study incorporates an imagined task-scenario to address this question.

The main goal of both studies is to explore the extent to which experienced users of computer systems remain uncertain about the effects of operators. If such uncertainty is limited to novice users of very complex devices (such as Suchman's subjects), then it has few implications for models of action, for we can readily consider that uninstructed novice users must pass through a phase of experimentation before the problem space in which planning takes place can even be constructed (cf. Shrager & Klahr, 1986). If, however, it is prevalent among experienced users then it suggests more radical revisions to our notions of planning and interaction.

## 2. Study 1

### 2.1. INTRODUCTION

Keyboard-driven screen editors rely on commands to move the cursor around the text. Most such systems have separate commands for cursor movements by various units, either forwards or backwards. If any of these commands are to be used within a planned sequence, then users will need to remember the exact cursor-behaviour that each command effects. The same is true of commands that are not specifically for cursor movement. For example, if a find command is to be followed by an insertion, and the sequence of actions is to be planned, the user must know exactly where, on or by the found string, the cursor finishes. Likewise, for deletions: if they are to be integrated into a planned sequence, the exact effects of commands must be known. This study examines the extent to which experienced users of screen editors possess such knowledge, and thus, by implication, the extent to which they are able to plan action sequences during use of the device.

### 2.2. METHOD

#### *2.2.1. Subjects*

The 15 volunteers were all staff members or research students in the Department of Psychology or in the School of Independent Studies at the University of Lancaster. The experience of the subjects with their target system ranged from 50 hours (one subject) to 1500 hours (two subjects), with a mean of over 700 hours, according to the subjects' own estimates.

#### *2.2.2. Word-processing systems*

Ten of the 15 subjects used the WordPerfect system on an IBM PC compatible. Three subjects used LocoScript on an Amstrad WPC. One subject used WordStar,

and one subject used SuperWriter. The behaviour of these four systems with regard to the questions asked (Figure 1) differs only slightly. Obviously, subjects' answers were marked against the correct behaviour of the appropriate system. Where the systems' differing behaviour is of interest, note has been made in the results/discussion section.

*2.2.3. Questions*

Each subject completed a simple questionnaire about the behaviour of the word processor with which they were most familiar. After naming the system, and estimating the time they had spent using the system, in weeks and hours per week, subjects were asked if they used the word-forward and word-backward cursor move commands, the find command and the word-delete command. For each command they did use, subjects were asked to mark the exact effects of the command being issued in various situations. (Additionally, some subjects guessed answers for those commands which they reported never using.) Subjects indicated confidence in each answer, on a five point scale (1 = very unsure, 5 = certain). The command-effect questions are shown in Figure 1.

*2.2.4. Procedure*

On volunteering, subjects were handed a questionnaire, which they completed in their own time. They were asked to complete it at least 1 hour after they had last used their word processor.

## 2.3. RESULTS AND DISCUSSION

Table 1 summarizes the main data from the study. It shows, for each subject, the system used, the subject's estimate of time spent using the system and performance on the command-effect questions. Empty cells in the table indicate subjects' reports of never using a particular command.

*2.3.1. Experience and command usage*

The first findings to consider are experience with the target system, and the extent to which the four basic facilities covered by the questionnaire were used by the subjects (according to their reports). Table 1 shows that only four of the 15 subjects report using all four commands, and that two subjects, each with around 1000 hours experience of the system report never having used any of the four commands.

These findings have some precedent in the HCI literature. Draper (1984) has reported that "expert" Unix users typically only regularly use a small subset of the command language, but Unix is a large multi-purpose system, that few people would be expected to master totally. Similarly, in a field observation of word processor users, Rosson (1984) showed that experienced users typically failed to use all available commands.

However, unlike these previous studies, the current study was not designed primarily to investigate the usage of facilities, and all the four commands in question were chosen to be important, frequent, core commands, essential to the usability of keyboard-driven word processors. It is all the more surprising then, that the earlier reports of non-use are replicated. It is clear that there exist experienced users who only harness a tiny fraction of the full power of their word processors, and who labour with grossly inefficient methods to perform frequent, straightforward tasks.

1. Where will the cursor finish up if the forward-a-word command is issued in the following situations? The arrow represents the current position of the cursor.

1.1 ^the long room

1.2 the^ long room

1.3 the^ long room

1.4 the^ long room

1.5 the^long-room by the piano

1.6 the long ro^om. By the piano

1.7 Just before supper every evening, grandfather would go into the long^ room. By the piano

2. Where will the cursor finish up if the back-a-word command is issued in the following situations?

2.1 the l^ong room

2.2 the long^ room

2.3 the ^ long room

2.4 the l^ong room

2.5 the long-ro^om by the piano

2.6 the long room. ^By the piano

2.7 Just before supper every evening, grandfather would go into the long ro^om. By the piano

3. Where will the cursor finish up if "find long" is issued in the following situations?

3.1 ^the long room

3.2 it^ won't be long. But while

3.3 another^ of those long-shots I suppose

4. If the delete-word command is issued in the situations below, what will be the result? Please write out the final phrase, and mark the final cursor position.

4.1 the ^long room

4.2 the^ long room

4.3 the^ long room

4.4 the^ long room

4.5 the ^long-room by

4.6 the long ro^om. By

4.7 Just before supper every evening, grandfather would go into the l^ong room. By the piano

4.8 Just before supper every evening, grandfather would go into the long room. By the piano^

FIGURE 1. Study 1. Questions about the effects of commands. In the study each question was accompanied with a confidence rating scale.

TABLE 1

*Study 1: Estimated experience and performance on command-effects questions. Empty cells signify a report of never using a command*

| System | Subject | Experience (h) | Forward-word (max 7) | Backward-word (max 7) | Find (max 3) | Delete-word (max 8) |
|---|---|---|---|---|---|---|
| WordPerfect 4.2 | 1 | 550 | 7 | 6 | — | 4 |
| | 2 | 300 | 6 | 4 | 3 | — |
| | 3 | 300 | — | — | 3 | — |
| | 4 | 1000 | — | — | — | — |
| | 5 | 600 | — | — | — | 3 |
| | 6 | 50 | 7 | 7 | 0 | 1 |
| | 7 | 300 | 7 | 3 | 3 | 3 |
| | 8 | 600 | 7 | 7 | 0 | 3 |
| | 9 | 1000 | — | — | — | 5 |
| | 10 | 1000 | — | — | — | — |
| LocoScript II | 11 | 1000 | 6 | 3 | 3 | — |
| | 12 | 800 | 0 | 0 | 3 | — |
| | 13 | 500 | — | — | 3 | — |
| SuperWriter 1.09 | 14 | 1500 | 5 | 3 | 3 | 2 |
| WordStar | 15 | 1500 | — | — | 3 | — |
| Mean | | 713 | 5·6 | 4·1 | 2·4 | 3 |

*2.3.2. Performance on the command-effect questions*

Knowledge of the effects of commands varies substantially between individuals and between commands. We will begin by considering word-forward cursor movement, for which four out of the eight users made no errors. However, close inspection of the errors that are made, in relation to the behaviour of the systems, suggests that this performance may be a gross overestimate of what subjects know.

With the exception of S12, who consistently reported that the cursor finished on the space character before the next word, subjects' responses to all the questions were the same: the cursor finishes on the first character of the next word, regardless of spacing, or of new-lines. This is indeed the way the WordPerfect command behaves, leading to a perfect performance, except for S2, who failed to answer question 1.4. However, LocoScript treats hyphens as word breaks, leading to a different result on question 1.5, and an error for S11. Likewise, SuperWriter's word-forward command does not move the cursor onto new lines, leading to an error for S13 on question 1.7. The other error by S13 results from another peculiarity of SuperWriter: it appears to treat the space before a word as part of that word, leading to different results for questions 1.2 and 1.4.

The pattern in these results seems clear. Users have common expectations which allow them to predict the effects of the word-forward command. When their particular word processor departs from these predictions, they fail to acquire a different model. There is little evidence that subjects have learnt how the command behaves by using the system. The good performance is an accident; the behaviour of the WordPerfect system happens to conform to the most common guesses.

This conclusion is given strong additional support if we consider the performance of users who admit to never using the command, but who guessed at the answers. For this command, six of the seven non-users guessed answers. In each case their guesses were identical to those of the "perfect-performance" WordPerfect users.

Turning to the find command, we again find very good performance in general. Apart from two subjects, who both guess that the cursor finishes after a searched-for string, all subjects correctly respond that the cursor finishes over the first character of the string. In this case there are no variations between systems, so we simply cannot tell whether subjects' knowledge is suspect in any way, but once again the subjects who guessed, having no experience of the command, performed perfectly. The find questions evidently make a very poor instrument for inspecting users' knowledge.

It is worth pausing momentarily to consider the implications of the apparent guessability of the effects of the WordPerfect word-forward command, and find commands in general. In many analyses such guessability may be seen as an indicator of good design. However, the thesis developed in this paper is that users readily pick up information from the display during routine task performance. If this thesis is correct, then guessability of visible operator effects may hardly matter.

Performance on the word-backward and word-deletion commands is substantially more interesting. The word-backward command, on all the systems under consideration, moves the cursor to the left until it reaches a character immediately preceded by a space. In general terms, it moves the cursor to the beginning of the current word, or, if it is already at the beginning, or in a space between words, to the beginning of the previous word. This seems as straightforward as the behaviour of the word-forward command, and it seems likely that the commands are used equally often. However, performance on the word-backward questions was much worse. Only two subjects achieved perfect performance. Four subjects (S2, S7, S11, S14) guessed that the cursor moves to the beginning of the previous word, regardless of its current position. This is perhaps a generalization from the move forward algorithm, but it leads to incorrect answers on four of the seven questions (S2 achieved a better score because he mixed the incorrect with the correct algorithm). As with the word-forward command, S12 mistakenly guessed that the cursor finished on the space before the first character of a word.

Performance on the delete-word questions is even worse, and much more irregular. The behaviour of the two systems for which responses were offered varies quite radically. In WordPerfect, the delete-word command deletes a whole word (i.e. a string of adjacent characters between spaces), regardless of the position of the cursor. If the cursor is on a space character between words, the previous word is deleted. The cursor finishes on the first character of the word after the one deleted. SuperWriter deletes the letter on which the cursor is positioned, and the remainder of the current word, or, if the cursor is on a space, it deletes the next word. If the cursor was on a letter, no space is left between the previous letter and the first letter of the next word, on which the cursor finishes. If the cursor was on a space character, a single space remains between the previous word and next remaining word; the cursor finishes on that space.

None of the WordPerfect users knew that delete-word would delete the previous word if the cursor started on a space; they also made several errors with the final

cursor position. The SuperWriter user believed that delete-word deleted entire words, rather than only the letters ahead of the cursor.

Finally, let us consider subjects' reported confidence in their answers. Whatever the individual variations underlying confidence ratings, if subjects possess explicit, but partial knowledge of the effects of commands we would expect confidence on correct answers to be greater than confidence on incorrect answers. Ten subjects made some correct and some incorrect answers. For each of these subjects a mean confidence rating was computed for correct and incorrect answers. The difference between these mean confidence ratings is not significant ($t_9 = 1{\cdot}43$, $p > 0{\cdot}1$).

### 2.4. CONCLUSIONS

This first study has suggested that users of word processing systems do not typically learn the precise effects of commands, even commands that they use regularly. This negative finding constrains plausible models of action at the user interface. Users cannot make precise plans to control actions beyond those whose affect they do not know. The approximate knowledge that users do possess about the effects of commands like delete-word is not sufficient to determine subsequent actions, such as the insertion of a new word. Instead, users must consult the display before continuing, even if their top-level goals remain unaltered.

## 3. Study 2

### 3.1. INTRODUCTION

It is possible that the results of Study 1 expose a design-flaw with keyboard-driven screen editors, more than a general tendency for display-based action at the user interface. Perhaps it is the complex nature of the algorithms effecting cursor-movements and word-deletions, and the fact that they can map only approximately onto the users' "goal space" concepts of words, sentences and the like (Payne, 1987), that depresses learning of and memory for command effects, and forces reliance on the display? Perhaps more recent designs, which, after all, have removed the need for cursor-control commands, will not suffer from this "problem"?

The second study tests the knowledge of users of menu-driven word-processing systems (specifically, MacWrite and Microsoft Word, running on an Apple Macintosh) for the find dialogue—the actions needed to search for strings of text under various circumstances, and their effects. As well as testings user's knowledge of the precise effects of operators, this study tests recall of the details of actions, such as the names of menu-items. It thus offers a partial replication of the Mayes *et al.* (1988) study, using different test conditions.

### 3.2. METHOD

#### *3.2.1. Subjects, systems, questions and procedure*

Fifteen staff members and postgraduate students of the Departments of Psychology and Computing at Lancaster University volunteered to take part.

Five subjects were regular MacWrite 1 users, the remaining 10 were regular users of MicroSoft Word 3.

*Please consider the system with which you are more familiar. Delete the other name.*

1. Your are editing a paper using MacWrite/Word. Your cursor is at the top of the file. You wish to find the next occurrence of the word "target". Please specify the sequence of actions you would need to execute:

2. Assume "target" is found.

2.1. Where will the cursor now be?

2.2. What will happen if you type "new"?

2.3. Specify the sequence of actions you need to execute to enter "new" in front of "target" (assume you have done 1, but not done 2.2).

3. Assume the start-point and actions of question 1. Assume "target" is **not** in the current file (so the scenario in question 2 did not occur).

3.1. What will happen? (Specify the screen states as exactly as possible).

3.2. Specify the sequence of actions you now need to execute to find the first "new" in the file (i.e. starting from the current device state).

4. How long have you been using the system? (Weeks, hours per week.)

FIGURE 2. MacWrite/Word string finding questionnaire.

The complete questionnaire is shown in Figure 2. Subjects were asked to describe sequences of actions to achieve find tasks, and to describe device states resulting from such actions. Subjects were handed the questionnaire by the author in their own offices, and completed them immediately. Completed questionnaires were returned in person to the author, who talked with subjects about any difficulties.

### 3.3. RESULTS AND DISCUSSION

Table 2 summarizes the experience of each subject, and the main errors they made on each question. It can readily be seen that no subject, of whatever experience, demonstrated perfect knowledge of the find dialogue. The nature of the errors is discussed further below.

The experience of subjects ranged from 12 to 2500 hours with a mean of over 814 hours (subjects' own estimates). One subject (with 60 hours experience) reported never using the find command, and is excluded from all analyses.

#### *3.3.1. Question 1: knowledge of primary action sequence*

A full answer to question 1 would specify that Find must be selected from the Search menu; "target" must be typed (it will appear automatically in the find window); the whole-word box should be checked (Word only—in MacWrite it is selected as default); finally the Start Search button (Word) or the Find Next button (MacWrite) must be clicked. Alternatively, for the last step, Word users could simply press the return key.

One subject (S9) reported using the accelerator key (clover-F) instead of the

TABLE 2

*Study 2: Summary of major errors made by each subject on each question. For question 1, names in quotes denote names of menu-items, or dialogue-buttons, reported incorrectly. For question 2, "window" denotes failure to remember that the find window remains active. For question 3, "OK" denotes failure to report selecting "OK" in the not-found dialogue box; "Window" denotes failure to remember and exploit the fact that the find window is still visible*

| System | Subject | Experience (h) | Find User? | Question 1 | Question 2 | Question 3 |
|---|---|---|---|---|---|---|
| Word | 1 | 60 | No | N/A | N/A | N/A |
| | 2 | 100 | Yes | Start search | Window | OK |
| | 3 | 70 | Yes | Search | Window | Window |
| | 4 | 200 | Yes | Search | Window | OK<br>Window |
| | 5 | 12 | Yes | Search<br>Find<br>Start search | Window | No answer |
| | 6 | 200 | Yes | — | Window | OK<br>Window |
| | 7 | 1500 | Yes | Find<br>Start search | — | — |
| | 8 | 1500 | Yes | Search | Window<br>Delete<br>targe | OK |
| | 9 | 1500 | Yes | — | Window | OK |
| | 10 | 24 | Yes | Search<br>Find<br>Start search | Window | Window |
| MacWrite | 11 | 700 | Yes | Search | Window | OK<br>Window |
| | 12 | 700 | Yes | Search | Window | OK<br>Window |
| | 13 | 2500 | Yes | Find next | Window | OK<br>Window |
| | 14 | 2500 | Yes | Find next | — | Window |
| | 15 | 650 | Yes | Search | Window | OK<br>Window |

Search–Find menu selection. All the others knew that an option had to be selected from one of the top-level menus, but fared less well at remembering the names. Of the remaining 13 subjects, only five remembered the name of the Search menu. Ten subjects correctly remembered the name of the Find menu-option, but this was cued by the text of the question.

Only one subject (S14) described the whole-word check, and she was a MacWrite user, for which whole-word is the default. The Word users either forgot about this step, or deliberately neglected the possibility of finding "target" as a substring of, say, "targetted".

In reporting the final step of the method, four Word subjects correctly noted that "return" could be pressed, five noted a button to be clicked but none of these correctly reported the name of this button ("Start Search"). The five MacWrite

users all correctly reported a button to be clicked, but only two correctly named the button ("Find Next").

*3.3.2. Question 2: knowledge of command effects*

Question 2 tested knowledge about the state of the device after a find command has successfully been executed. The critical information here is that, although the found string in the text will be highlighted, the find window remains active. This information has a vital effect on any subsequent actions. Question 2.1 is ambiguous, as the word "cursor" has two possible referents in this situation (i.e. the cursor in the text-window, and the cursor in the find window), but question 2.2 unambiguously taps the subjects' knowledge. If "new" is typed, it will appear after "target" in the find window. The main text file will remain unaltered.

Only two of the 14 subjects answered question 2.2 correctly. Among the remaining 12 subjects, eight reported that "new" would replace "target" in the text file, two subjects thought "new" would appear before "target", one thought "new" would appear after "target", one thought "new" would appear at the top of the text file (i.e. where the cursor was before the find operation).

Question 2.3 essentially offers subjects a chance to confirm their knowledge about the device-state given in response to question 2.2. Before inserting text in front of the found word, users must return control from the find window to the main text-window. This can be done in two ways, either by closing the find window, or by clicking on the text-window.

Unlike question 2.1 and 2.2, question 2.3 taps the critical knowledge of the device state by asking subjects to remember actions for a commonplace operation. However, answers to question 2.3 merely revealed the same flawed memory as did those for question 2.2. Of the two subjects who remembered, in their answers to question 2.2, that the find window remained active, one reported a correct method for returning control to the text-window, while the other reported a method for entering "new" in front of target in the find window. None of the other subjects noted this potential ambiguity in the question. They all reported methods for inserting "new" in the text file, but neglected to mention any step for transferring control to the text window, and so reported inadequate methods.

In summary, question 2 reveals very poor memory for the effects of the find operation on the device state. Despite three separate questions alluding to the same issue, only two of 14 subjects remembered a feature of the device state that is vital for any subsequent action, and only one correctly reported the action necessary to deal with this feature.

*3.3.3. Question 3: knowledge of effects of failure to find*

When search begins from the top of the document, both MacWrite and Word respond in essentially the same way to not-found strings. A new dialogue box appears, saying "End of document reached" (Word) or "Target not found" (MacWrite), and containing an "OK" button. The OK button must be selected (or in Word carriage-return can be pressed) before the user can proceed.

In scoring responses to question 3.1 the wording of the "not found" box was ignored, as this plays no part in the user's actions. It is more important to note whether subjects remembered the OK button, which does determine a necessary action. Only 4 subjects mentioned the OK button (S2, S3, S4, S9).

The method sought in question 3.2 should begin with selection of OK. Only three of the four subjects who reported the button specified this step, but one more subject specified the closing down of a not found box, while not remembering the name "OK". After this, a new search can exploit the still-active find window: "target" must be replaced by "new", but the search-find initiation of a find-dialogue is unnecessary. However, all but 4 subjects made the error of reinitiating a find dialogue, presumably forgetting that the find window would still be visible.

### 3.4. CONCLUSIONS

This study has shown that users of Macintosh-based word processors demonstrate only patchy recall of the find dialogue. Many users cannot remember the name of menu-items or dialogue buttons necessary to initiate and continue find dialogues: operators must be recognized in the context of use. Furthermore, most users do not remember blatant, visible properties of device-states that result from various find commands: they remain uncertain about the effects of operators. Such properties of device-states determine the user's future actions, so we can conclude that many experienced users cannot fully plan a sequence of actions to find and then edit a string of text.

## 4. General discussion

Perhaps the most striking aspect of the current results is that *even very experienced users* rely on the display for doing routine tasks. When a user with 1500 hours experience using SuperWriter wants to follow a word-deletion with a pre-meditated word-insertion, he cannot plan his actions beyond the word-deletion, as he does not know where the cursor will finish (Study 1, S 14). When a user with 2500 hours experience of MacWrite wants to find a word in the text and insert another word before or after it, she cannot plan accurately beyond the find command, for she does not remember that the find window will need to be de-activated before the text can be edited (Study 2, S 13). Most users, whatever their experience level, appear to suffer from similar limitations.

For such tasks, at least, the simplest planning model is wrong, nor will the current elaborations of it readily address the most important aspects of cognition in action. Learning to use the device successfully is not a matter of internalizing goals and plans. Rather, it is, in part, a matter of interpreting the state of the device, and working out the implications of that state for future actions.

This is not the first time that such a perspective has been offered. Draper (1986) has argued for a perspective on human–computer interaction that he refers to under the rubric "Information Flow". The key idea is that skilled activity at the user interface, is, in a very strong sense, interactive, in that information must flow from device to user, as well as from user to device. Draper's theoretical work was a major influence on the Mayes *et al.* (1988) study reported above, and on the experiments reported here. These studies offer some firm empirical evidence for what has previously been a powerful intuition.

What are the implications of display-based action for models of the user, and, more generally, for the applied science of HCI? The first implication to point out is a positive one. Existing models of user competence, such as task-action grammars (Payne & Green, 1986) describe the user's knowledge to perform simple tasks such

as "move the cursor forward a word" that are approximate descriptions of the character-based operations that the computer really performs. Study 1 shows that such approximate descriptions are psychologically real: users know how to move the cursor forward by a word, but not exactly where the cursor will finish.

The second implication is less consoling and more demanding. To understand action at the user interface, it seems models will be needed of the way that actions can be performed in the absence of perfect knowledge of operator-effects. Such models will need to stress the way information from the display is interpreted during and in support of, action sequences. A strong analogy here is with human conversation. A speaker in a conversation may be acting on the world (cf. Austin, 1962) but he or she cannot know in advance the ("perlocutionary") effects of this speech act. Rather, contributions to the ongoing discourse must be collaborated on, with both speakers working to ground their utterances on mutual understanding (Clark & Schaefer, 1987). Many HCI workers have drawn on the conversation analogy, for a variety of purposes. The suggestion here is that it provides the basis for a cognitive model of action at the user interface (for the first steps toward such a model see Payne, 1990).

Finally, what are the implications of display-based action for the design of user interfaces? First, it is important to deny a simple evaluation of display-based action as a Bad Thing, to be designed around wherever possible. Although it has been convenient in this paper to describe users as "unable to recall" and thus "reliant on the display" and "unable to plan ahead", there is no intended implication that such reliance exposes a weakness, of the user, or the used system. Indeed, the opposite may be closer to the truth; perhaps systems should be designed so that exploitation of the display during action is maximized, and memory load consequently reduced (a similar argument is made by Larkin, 1989).

A problem with this guideline, of course, is that it is hard to follow. To generate more practical design implications, research on display-based action will need to extend the current findings. One route for extension is to develop display-based models of performance, as suggested above, that allow cognitive descriptions of user interfaces: models that do for interactivity what task-action grammars (Payne & Green, 1986) try to do for consistency. An alternative, complementary route is to undertake detailed analysis of existing software artifacts to uncover and understand the range of techniques that have already been developed for supporting display-based action (cf. Carroll & Kellogg, 1989).

Within either approach, the current findings suggest a need to shift attention away from static aspects of the interface such as command language design or screen layout. It is time to start work on the dynamic properties of interactive systems—the properties that make them "interactive".

This work was supported by Alvey/SERC under grant GR/D 60355, a collaborative project with British Telecommunications plc. I am grateful to Andrew Howes for many discussions about the ideas presented here.

## References

ANDERSON, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, **89**, 369–406.

ANDERSON, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

ANDERSON, J. R. (1987). Causal analysis and inductive learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, Irving, CA.

AUSTIN, J. (1962). *How To Do Things With Words*. Oxford, UK: Clarendon Press.

CARD, S. K., MORAN, T. P. & NEWELL, A. (1983). *The Psychology of Human–Computer Interaction*. Hillsdale, NJ: Erlbaum.

CARROLL, J. M. & KELLOGG, W. A. (1989). Artifact as theory nexus: hermeneutics meets theory-based design. In K. BICE & C. LEWIS, Eds. *Proceedings of the CHI 89 Conference on Human Factors in Computing Systems*, pp. 7–14. New York: ACM.

CLARK, H. H. & SCHAEFER, E. F. (1987). Collaborating on contributions to conversations. *Language and Cognitive Processes*, **2**, 19–41.

DRAPER, S. W. (1984). The nature of expertise in Unix. In B. SHACKEL, Ed. *Human–Computer Interaction—INTERACT 84*. Amsterdam: North Holland.

DRAPER, S. W. (1986). Inter-referential I/O. In D. A. NORMAN & S. W. DRAPER, Eds. *User-Centred System Design*. Hillsdale, NJ: Erlbaum.

GREEN, T. R. G., BELLAMY, R. K. E. & PARKER, J. M. (1987). Parsing and gnisrap: a model of device use. In G. M. OLSON, S. SHEPPARD & E. SOLOWAY, Eds. *Empirical Studies of Programmers: Second Workshop*. Norwood, NJ: Ablex.

HAYES-ROTH, B. & HAYES-ROTH, F. (1979). A cognitive model of planning. *Cognitive Science*, **3**, 275–310.

HOWES, A. & PAYNE, S. J. (1990). Display-based competence: towards user models for menu driven systems. *International Journal of Man–Machine Studies*, **33**, 637–655.

KIERAS, D. E. & POLSON, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man–Machine Studies*, **22**, 365–394.

LARKIN, J. (1989). Display-based problem solving. In D. KLAHR & K. KOTOVSKY, Eds. *Complex Information Processing: The Impact of Herbert A. Simon*. Hillsdale, NJ: Erlbaum.

MAYES, J. T., DRAPER, S. W., MCGREGOR, A. M. & OATLEY, K. (1988). Information flow in a user interface: the effect of experience and context on the recall of MacWrite screens. In D. M. JONES & R. WINDER, Eds. *People and Computers IV*. Cambridge, UK: Cambridge University Press.

MILLER, G. A., GALANTER, E. & PRIBRAM, K. (1960). *Plans and the Structure of Behaviour*. New York: Holt, Rinehart and Wilson.

MORTON, J. (1967). A singular lack of incidental learning. *Nature*, **215** (no. 5097), 203–204.

NEWELL, A. & SIMON, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.

Norman, D. A. (1986). Cognitive engineering. In D. A. NORMAN & S. W. DRAPER, Eds. *User-Centred System Design*. Hillsdale, NJ: Erlbaum.

PAYNE, S. J. (1987). Complex problem spaces: modelling the knowledge needed to use interactive systems. In H. BULLINGER & B. SHACKEL, Eds. *Human–Computer Interaction–INTERACT 87*. Amsterdam: North Holland.

PAYNE, S. J. (1990). Looking HCI in the I. In D. DIAPER et al., Eds. *Human–Computer Interaction—INTERACT 90*. Amsterdam: North Holland.

PAYNE, S. J. & GREEN, T. R. G. (1986). Task-action grammar: a model of the mental representation of task languages. *Human–Computer Interaction*, **2**, 93–133.

POLSON, P. G. (1987). A quantitative theory of human-computer interaction. In J. M. CARROLL, Ed. *Interfacing Thought*. Cambridge MA: MIT Press.

ROBERTSON, S. P. & BLACK, J. B. (1986). Structure and development of plans in text-editing. *Human–computer Interaction*, **2**, 201–226.

ROSSON, M-B. (1984). The role of experience in text-editing. In B. SHACKEL, Ed. *Human–Computer Interaction—INTERACT 84*. Amsterdam: North Holland.

SHRAGER, J. & KLAHR, D. (1986). Instructionless learning about a complex device: the paradigm and observations. *International Journal of Man–Machine Studies*, **25**, 153–189.

SUCHMAN, L. A. (1987). *Plans and Situated Actions: The Problem of Human–Machine Communication*. Cambridge, UK: Cambridge University Press.

YOUNG, R. M. & SIMON, T. (1987). Planning in the context of human-computer interaction. In D. DIAPER & R. WINDER, Eds. *People and Computers III*. Cambridge, UK: Cambridge University Press.