TELEINFORMATICS 79, Boutmy/Danthine (eds.)

INTERACTING WITH THE COMPUTER: A FRAMEWORK

J. Morton, P. Barnard, N. Hammond* and J.B. Long

M.R.C. Applied Psychology Unit, Cambridge, England
*also IBM Scientific Centre, Peterlee, England

Recent technological advances in the development of information processing systems will inevitably lead to a change in the nature of human-computer interaction. Direct interactions with systems will no longer be the sole province of the sophisticated data processing professional or the skilled terminal user. In consequence, assumptions underlying human-system communication will have to be re-evaluated for a broad range of applications and users. The central issue of the present paper concerns the way in which this re-evaluation should occur.

First of all, then, we will present a characterisation of the effective model which the computer industry has of the interactive process. The shortcoming of the model is that it fails to take proper account of the nature of the user and as such can not integrate, interpret, anticipate or palliate the kinds of errors which the new user will resent making. For remember that the new user will avoid error by adopting other means of gaining his ends, which can lead either to non-use or to monstrously inefficient use. We will document some user problems in support of this contention and indicate the kinds of alternative models which we are developing in an attempt to meet this need.

The Industry's Model (IM)

The problem we see with the industry's model of the human-computer interaction is that it is computer-centric. In some cases, as we shall see, it will have designer-centric aspects as well. To start off with, consider a system designed to operate in a particular domain of activity. In the archetypal I.M. the database is neutralised in much the same kind of way that a statistician will ritually neutralise the data on which he operates, stripping his manipulation of any meaning other than the purely numerical one his equations impose upon the underlying reality. This arises because the only version of the domain which exists at the interface is that one which is expressed in the computer. This version, perhaps created by an expert systems analyst on the best logical grounds and the most efficient, perhaps, for the computations which have to be performed, becomes the one to which the user must conform. This singular and logical version of the domain will, at best, be neutral from the point of view of the user. More often it will be an alien creature, isolating the user and mocking him with its image of the world and its resources to which he must haplessly conform.

Florid language? But listen to the user talking.

> "We come into contact with computer people, a great many of whom talk a very alien language, and you have constant difficulty in trying to sort out this kind of mid-Atlantic jargon."
>
> "We were slung towards what in my opinion is a pretty inadequate manuel and told to get on with it"
>
> "We found we were getting messages back through the terminal saying there's not sufficient space on the machine. Now how in Hell's name are we supposed to know whether there's sufficient space on the machine?"

In addition the industry's model does not really include the learning process; nor does it always take adequate note of individual's abilities and experience:

> "Documentation alone is not sufficient; there needs to be the personal touch as well."
>
> "Social work being much more of an art than a science then we are talking about people who are basically not very numerate beginning to use a machine which seems to be essentially numerate."

Even if training is included in the final package it is never in the design model. Is there anyone here, who, faced with a design choice asked the questions "Which option will be the easiest to describe to the naive user? Which option will be easiest to understand? Which option will be easiest to learn and remember?"

Let us note again the discrepancy between the I.M. view of error and ours. For us errors are an indication of something wrong with the system or an indication of the way in which training should proceed. In the I.M. errors are an integral part of the interaction. For the onlooker the most impressive part of a D.P. interaction is not that it is error free but that the error recovery procedures are so well practiced that it is difficult to recognise them for what they are.

We would not want it thought that we felt the industry was totally arbitrary. There are a number of natural guiding principles which most designers would adhere to. We do not anticipate meeting a system in which the command DESTROY has the effect of preserving the information currently displayed while PRESERVE had the effect of erasing the operating system.

However, the principles employed are intuitive and non-systematic. Above all they make the error of embodying the belief that just as there can only be one appropriate representation of the domain, so there is only one kind of human mind.

A nice example of a partial use of language constraints is provided by a statistical package called GENSTAT. This package permits users to have permanent userfiles and also temporary storage in a workfile. The set of commands associated with these facilities are:

PUT - copies from core to workfile
GET - copies from workfile to core
FILE - defines a userfile
SAVE - copies from workfile to userfile
FETCH - copies from userfile to workfile

The commands certainly have the merit that they have the expected directionality with respect to the user. However to what extent do, for example, FETCH and GET relate naturally to the functions they have been assigned? No doubt the designers have strong intuitions about these assignments. So do users and they do not concur. We asked 40 people here at the A.P.U. which way round they thought the assignments should go: nineteen of these agreed with the system designers, 21 went the other way. The confidence levels of rationalisations were very convincing on both sides!

The problem, then, is not just that systems tend to be designer-centric but that the designers have the wrong model either of the learning process or of the non-D.P. users' attitude toward error. A part-time user is going to be susceptible to memory failure and, in particular, to interference from outside the computer system. du Boulay and O'Shea [1] note that naive users can use THEN in the sense of 'next' rather than as 'implies'. This is inconceivable to the I.M. for THEN is almost certainly a full homonym for most D.P. and the appropriate meaning thoroughly context-determined.

An Alternative to the Industry Model

The central assumption for the system of the future will be 'systems match people' rather than 'people match systems'. Not entirely so, as we shall elaborate, for in principle, the capacity and perspectives of the user with respect to a task domain could well change through interaction with a computer system. But the capacity to change is more limited than the variety available in the system. Our task, then, is to characterise the mismatch between man and computer in such a way that permits us to direct the designer's effort. In doing this we are developing two kinds of tool, conceptual and empirical. These interrelate within an overall scheme for researching human-computer interaction as shown in Figure 1.

Relating Conceptual and Empirical Tools

The conceptual tools involve the development of a set of analytic frameworks appropriate to human computer interaction. The empirical tools involve the development of valid test procedures both for the introduction of new systems and the proving of the analytic tools. The two kinds of tool are viewed as fulfilling functions comparable to the role of analytic and empirical tools in the development of technology. They may be compared with the analytic role of physics, metallurgy and aerodynamics in the development of aircraft on the one hand and the empirical role of a wind tunnel in simulating flight on the other hand.

Empirical Tools

The first class of empirical tool we have employed is the observational field study, with which we aim to identify some of the variables underlying both the occasional user's perceptions of the problems he encounters in the use of a computer system, and the behaviour of the user at the terminal itself. The opinions cited above were obtained in a study of occasional users discussing the introduction and use of a system in a local government centre [2]. The discussions were collected using a technique which is particularly free from observer influence [3].

In a second field study we obtained performance protocols by monitoring users while they solved a predefined set of problems using a data base manipulation language [4]. We recorded both terminal performance and a running commentary which we asked the user to make, and wedded these to the state of the machine to give a total picture of the interaction. The protocols have proved to be a rich source of classes of user problem from which hypotheses concerning the causes of particular types of mismatch can be generated. There is thus a close interplay between these field studies, the generation of working hypotheses and the development of the conceptual frameworks. We give some extracts from this study in a later section.

A third type of empirical tool is used to test specific predictions of the working hypothesis. The tool is a multi-level interactive system which enables the experimenter to simulate a variety of user interfaces, and is capable of modelling and testing a wide range of variables [5]. It is based on a code-breaking task in which users perform a variety of string-manipulation and editing functions on coded messages.

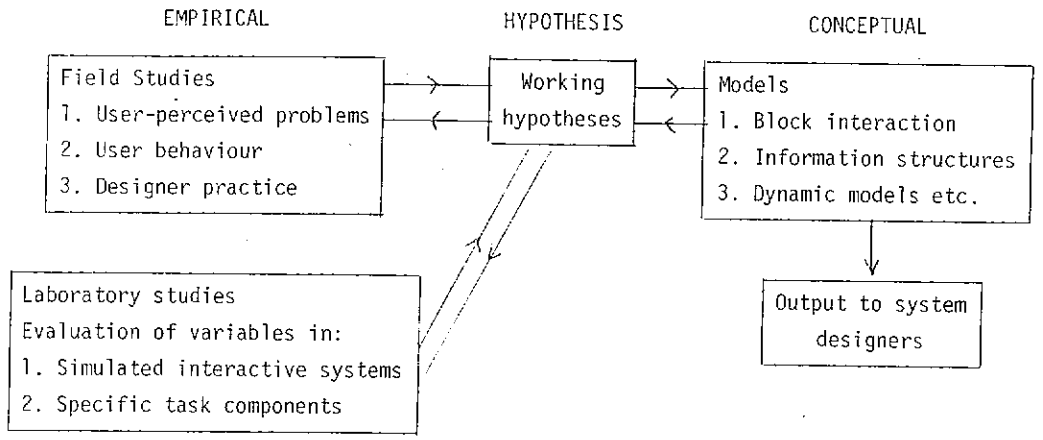


FIGURE 1: RELATIONSHIP BETWEEN EMPIRICAL AND CONCEPTUAL TOOLS

It allows the systematic evaluation of notational, semantic and syntactic variables. Among the results to be extensively reported elsewhere is that if there is a common argument in a set of commands, each of which takes two arguments, then the common argument must come first for greatest ease of use. Consistency of argument order is not enough: when the common argument consistently comes second no advantage is obtained relative to inconsistent ordering of arguments [6].

Conceptual Tools

Since we conceive the problem as a cognitive one, the tools are from the cognitive sciences. Also we define the problem as one with those users who would be considered intellectually and motivationally qualified by any normal standards. Thus we do not admit as a potential solution that of finding "better" personnel, or simply paying them more, even if such a solution were practicable. The cognitive incompatibility we describe is qualitative not quantitative and the mismatch we are looking for is one between the user's concept of the system structure and the real structure: between the way the data base is organised in the machine and the way it is organised in the head of the user: the way in which system details are usually encountered by the user and his preferred mode of learning.

The interaction of human and computer in a problem-solving environment is a complex matter and we cannot find sufficient theory in the psychological literature to support our intuitive needs. We have found it necessary to produce our own theories, drawing mainly on the spirit rather than the substance of established work. Further than this, it is apparent that the problem is too complex for us to be able to use a single theoretical representation.

The model should not only be appropriate for design, it should also give a means of characterising errors - so as to understand their origins and enable corrective measures to be taken. Take the following protocol.

> The user is asked to find the average age of entries in the block called PEOPLE.
> I'll have a go and see what happens"
> types: *T <-AVG(AGE,PEOPLE)
> machine response: AGE - UNSET BLOCK
> "Yes, wrong, we have an unset block. So it's reading AGE as a block, so if we try AGE and PEOPLE the other way round maybe that'll work."

This is very easy to diagnose and correct. The natural language way of talking about the target of the operation is mapped straight into the argument order. The cure would be to reverse the argument order for the function AVG to make it compatible.

The next protocol is more obscure. The task is the same as in the preceding one.

> "We can ask it (the computer) to bring to the terminal the average value of this attribute."
> types: *T <-AVG(AGE)
> machine response: AVG(AGE) - ILLEGAL NAME
> "And it's still illegal... (...) ... I've got to specify the block as well as the attribute name."

Well of course you have to specify the block. How else is the machine going to know what you're talking about? A very natural I.M. response. How can we be responsible for feeble memories like this.

However, a more careful diagnosis reveals that the block PEOPLE is the topic of the 'conversation' in any case. The block has just been used and the natural language conventions are quite clear on the point.

We have similar evidence for the importance of human-machine discourse structures from the experiment using the code-breaking task described above. Command strings seem to be more 'cognitively compatible' when the subject of discourse (the common argument) is placed before the variable argument. This is perhaps analogous to the predisposition in sentence expression for stating information which is known or assumed before information which is new [7]. We are currently investigating this influence of natural language on command string compatibility in more detail.

The Block Interaction Model

Systematic evidence from empirical studies, together with experience of our own, has led us to develop a conceptual analysis of the information in the head of the user (see figure 2). Our aim with one form of analysis is to identify as many separable kinds of knowledge as possible and chart their actual or potential interactions with one another. Our convention here is to use a block diagram with arrows indicating potential forms of interference. This diagram enables us to classify and thus group examples of interference so that they could be counteracted in a coordinated fashion rather than piecemeal. It also enables us to establish a framework within which to recognise the origin of problems which we haven't seen before. Figure 2 is a simplified form of this model. The blocks with double boundaries, connected by double lines, indicate the blocks of information used by the ideal user. The other lines indicate prime classes of interference.

The terminology we have used is fairly straightforward:

Domain - the range of the specific application of a system. This could be a hospital, a city's buildings, a set of knowledge such as jobs in an employment agency.

Objects - the elements in the particular data base. They could be a relational table, patients' records.

Operations - the computer routines which manipulates the objects.

Labels - the letter sequences which activate operators which, together with arguments and syntax, constitute the commands.

Work base - in general, people using computer systems for problem solving have had experience of working in a non-computerised work environment either preceding the computerisation or at least in parallel with the computer system. The representation of this experience we call the work-base version. There will be overlap between this and the users representation of the computer's version of the domain; but there will be differences as well, and these differences we would count as potential sources of interference. There may be differences in the underlying structure of the data in the two cases, for example, and will certainly be differences in the objects used. Thus a user found to be indulging in complex checking procedures after using the command FILE turned out to be perplexed that the material filed was still present on the screen. With pieces of paper, things which are filed actually go there rather than being copied.

Here are some examples of interference from one of our empirical studies [4]:

Interference on the syntax from other languages. Subject inserts necessary blanks to keep the strings a fixed length.

> "Now that's Matthewson, that's 4, 7, 10 letters, so I want 4 blanks"

```
types:  A←<:S:NAME = 'MATTHEWSON    ':>PEOPLE
```

Generalised interference

> "Having learned how reasonably well to manipulate one system, I was presented with a totally different thing which takes months to learn again."

Interference of other machine characteristics on machine view

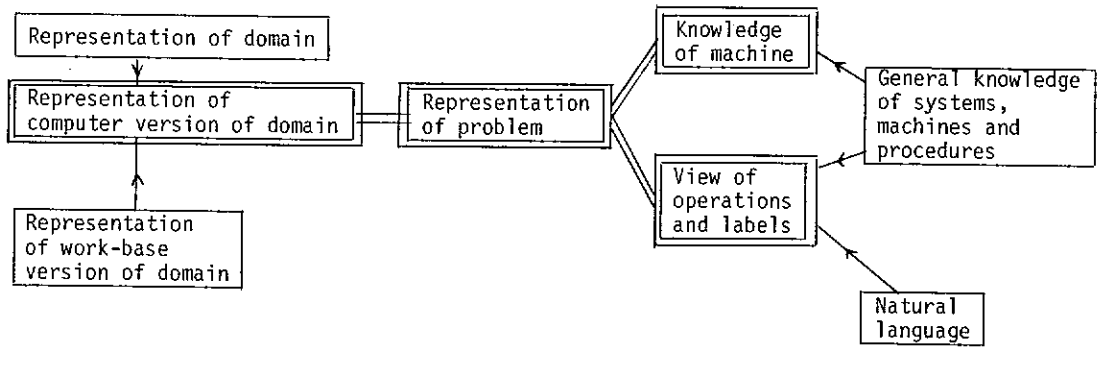


FIGURE 2: BLOCK INTERACTION MODEL

> "I'm thinking that the bottom line is the line I'm actually going to input. So I couldn't understand why it wasn't lit up at the bottom there, because when you're doing it on (another system) it's always the bottom line."

The B.I.M. can be used in two ways. We have illustrated its utility in pinpointing the kinds of interference which can occur from inappropriate kinds of information. We could look at the interactions in just the opposite way and seek ways of maximising the benefits of overlap. This is, of course, the essence of 'cognitive compatibility' which we have already mentioned. Trivially, the closer the computer version of the domain maps onto the user's own version of the domain the better. What is less obvious is that any deviations should be systematic where possible.

In the same way, it is pointless to design half the commands so that they are compatible with the natural language equivalents and use this as a training point if the other half, for no clear reason, deviate from the principle. If there are deviations then they should form a natural sub-class or the compatibility of the other commands will be wasted.

Information Structures

In the block interaction model we leave the blocks ill-defined as far as their content is concerned. Note that we have used individual examples for user protocols as well as general principles in justifying and expanding upon the distinctions we find necessary. What we fail to do in the B.I.M. is to characterise the sum of knowledge which an individual user carries around with him or brings to bear upon the interaction. We have a clear idea of cognitive compatibility at the level of an individual. If this idea is to pay then these structures must be more detailed.

There is no single way of talking about information structures. At one extreme there is the picture of the user's knowledge as it apparently reveals itself in the interaction; the view, as it were, that the terminal has of its interlocutor. From this point of view the motivation for any key press is irrelevant. This is clearly a gross oversimplification.

The next stage can be achieved by means of a protocol. In it we would wish to separate out those actions which spring from the users concept of the machine and those actions which were a result of him being forced to do something to keep the interaction going. This we call heuristic behaviour. This can take the form of guessing that the piece of information which is missing will be consistent with some other system or machine. "If in doubt, assume that it is Fortran" would be a good example of this. The user can also attempt to generalise from aspects of the current system he knows about. One example from our study was where the machine apparently failed to provide what the user expected. In fact it had but the information was not what he had expected. The system was ready for another command but the user thought it was in some kind of a pending state, waiting with the information he wanted. In certain other stages - in particular where a command has produced a result which fills up the screen - he had to press the ENTER key - in this case to clear the screen. The user then over-generalised from this to the new situation and pressed the ENTER key again, remarking

> "Try pressing ENTER again and see what happens."

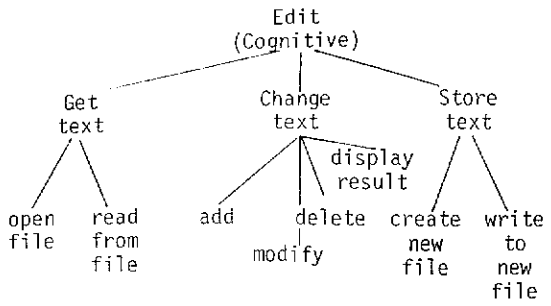
We would not want to count the user's behaviour in this sequence as representing his knowledge of the system - either correct knowledge or incorrect knowledge. He had to do something and couldn't think of anything else.

When the heuristic behaviour is eliminated we are left with a set of information relevant to the interaction. With respect to the full, ideal set of such information, this will be deficient with respect to the points at which the user had to trust to heuristic behaviour. Note that it will also contain incorrect information as well as correct information; all of it would be categorised by the user as what he knew, if not all with complete confidence, certainly with more confidence than his heuristic behaviour.

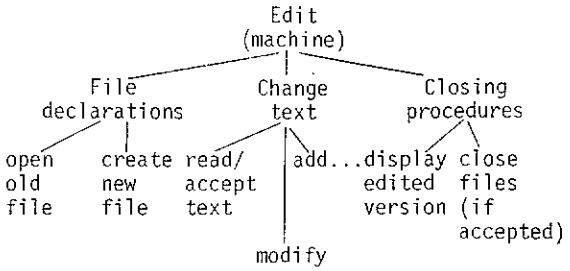
The thing which is missing from B.I.M. and I.S. is any notion of the dynamics of the interaction. We find we need three additional notations at the moment to do this. One of these describes the planning activity of the user, one charts the changes in state of user and machine and one looks at the general cognitive processes which are mobilised.

Goal Structure Model

The user does some preparatory work before he presses a key. He must formulate some kind of plan, however rudimentary. This plan can be represented, at least partially, as a hierarchical organisation. At the top might be goals such as "Solve problem P" and at the bottom "Get the computer to display Table T". The Goal Structure model will show the relationships among the goals. This can be compared with the way of structuring the task imposed by the computer. For example, a user's concept of editing might lead to the goal structure:



The particular editing system may have a structure, defined by modes of operation.



Two problems would arise here. Firstly the new file has to be opened at an 'unnatural' place. Secondly the acceptance of the edited text changes from being a part of the editing process to being a part of the filing process.

The goal structure model, then, gives us a way of describing such structural aspects of the user's performance and the machines requirements. Note that such goals might be created in advance or at the time a node is evaluated. Thus the relationship of the GSM to real time is not simple.

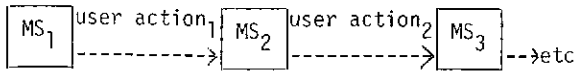
The technique for determining the goal structure may be as simple as asking the user "What are you trying to do right now and why?" This may be sufficient to reveal procedures which are inappropriate for the program being used.

State Transition Model

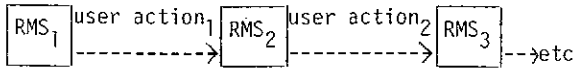
In the course of an interaction with a system a number of changes take place in the state of the machine. At the same time the user's perception of the machine state is changing. It will happen that the user misjudges the effect of one command and thereafter enters others which from an outside point of view seem almost random. Our point is, as before, that the interaction can only be understood from the point of view of the user.

This brings us to the third of the dynamic aspects of the interaction: the progress of the user as he learns about the system.

Let us explore some ways of representing such changes. Take first of all the state of the computer. This change is a result of user actions and can thus be represented as a sequence of Machine States (M.S.) consequent on user action.

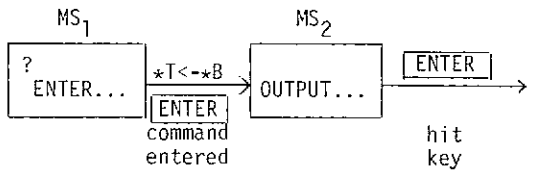


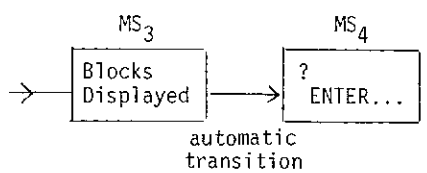
We have noted that the user's view of the machine state may be incorrect and his actions will depend upon his internalised Representation of the Machine State (RMS) rather than the actual state.



If the interaction is error free, changes in the representations would follow changes in the machine states in a homologous manner. Errors will occur if the actual machine state does not match its representation.

We will now look at errors made by a user of an interactive data enquiry system. We will see errors which reveal both the inadequate knowledge of the particular machine state or inadequate knowledge of the actions governing transitions between states. The relevant components of the machine are the information on the terminal display and the state of a flag shown at the bottom right hand corner of the display which informs the user of some aspects of the machine state (ENTER... or OUTPUT...). In addition there is a prompt, "?", which indicates that the keyboard is free to be used, there is a key labelled ENTER. In the particular example the user wishes to list the blocks of data he has in his workspace. The required sequence of machine states and actions is:





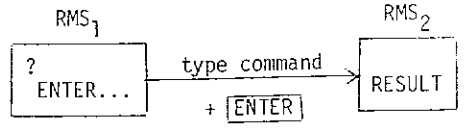
The protocol is as follows:

User types: *T ← *B ENTER

The machine echoes the command and waits with OUTPUT flag showing.

User: "Nothing happening. We've got an OUTPUT there in the corner I don't know what that means."

The user had no knowledge of MS_2; we can hypothesise his representation of the transition to be:



This is the result of an overgeneralisation. Commands are obeyed immediately if the result is short, unless the result is block data of any size. The point of this is that the data may otherwise wipe everything from the screen. With block data the controlling program has no look-ahead to check the size and must itself simply demand the block, putting itself in the hands of some other controlling program. We see here then a case where the user needs to have some fairly detailed and otherwise irrelevant information about the workings of the system in order to make sense of (as opposed to learn by rote) a particular restriction.

The user was told how to proceed, types ENTER, and the list of blocks is displayed together with the next prompt. However, further difficulties arise because the list of blocks includes only one name and the user was expecting a longer listing. Consequently he misconstrues the state of the machine.

(continuing from previous example)

User types ENTER
Machine replies with block list and prompt.
Flag set to ENTER...
"Ah, good, so we must have got it right then. A question mark! (the prompt). It doesn't give me a listing. Try pressing ENTER again and see what happens."
User types ENTER
"No? Ah, I see. Is that one absolute block, is that the only blocks there are in the workspace?"

This interaction indicates that the user has derived a general rule for the interaction:

"If in doubt press ENTER"

After this the user realises that there was only one name in the list. Unfortunately his second press of the ENTER key has put the machine into Edit mode and the user thinks he is in command mode. As would be expected the results are strange.

At this stage we can show the machine state transitions and the user's representation together in a single diagram, figure 3.

This might not be elegant but it captures a lot of features of the interaction which might otherwise be missed.

Information Processing Model

The final model we use calls upon models currently available in cognitive psychology which deal with the dynamics of word recognition and production, language analysis and information storage and retrieval. The use of this model is too complex for us to attempt a summary here.

Conclusion

We have stressed the shortcomings of what we have called the Industrial Model and have indicated that the new user will deviate considerably from this model. In its place we have suggested an alternative approach involving both empirical evaluations of system use and the systematic development of conceptual analyses appropriate to the domain of person-system interaction. There are, of course, aspects of the I.M. which we have no reason to disagree with, for example, the idea that the computer can beneficially transform the users view of the problems with which he is occupied. However, we would appreciate it if someone would take the trouble to support this point with clear documentation. So far as we can see it is simply asserted.

Finally we would like to stress that nothing we have said is meant to be a solution - other than the methods. We do not take sides for example, on the debate as to whether or not interactions should be in natural language - for we think the question itself is a gross oversimplification. What we do know is that natural language interferes with the interaction and that we need to understand the nature of this interference and to discover principled ways of avoiding it. And what we know above all is that the new user is most emphatically not made in the image of the designer.

References

[1] du Boulay, B. and O'Shea, T. Seeing the works: a strategy of teaching interactive programming. Paper presented at Workshop on 'Computing Skills and Adaptive Systems', Liverpool, March 1978.

[2] Hammond, N.V., Long, J.B. and Clark, I.A. Introducing the interactive computer at work: the users' views. Paper presented at Workshop on 'Computing Skills and Adaptive Systems', Liverpool, March 1978.

[3] Wilson, T. Choosing social factors which should determine telecommunications hardware design and implementation. Paper presented at Eighth International Symposium on Human Factors in Telecommunications, Cambridge, September 1977.

[4] Documenting man-computer mismatch with the occasional interactive user. APU/IBM project report no. 3, MRC Applied Psychology Unit, Cambridge, September 1978.

[5] Hammond, N.V. and Barnard, P.J. An interactive test vehicle for the investigation of man-computer interaction. Paper presented at BPS Mathematical and Statistical Section Meeting on 'Laboratory Work Achievable only by Using a Computer', London, September 1978.

[6] An interactive test vehicle for the study of man-computer interaction. APU/IBM project report no. 1, MRC Applied Psychology Unit, Cambridge, September 1978.

[7] Halliday, M.A.K. Notes on transitivity and theme in English. Part 1. Journal of Linguistics, 1967, 3, 199-244.

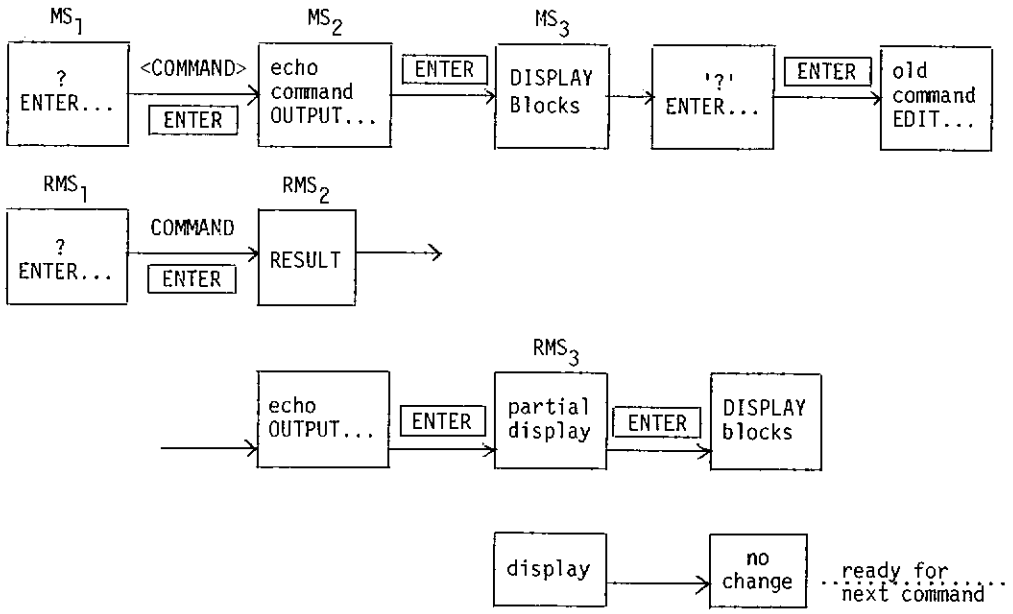


FIGURE 3: STATE TRANSITION EXAMPLE