Institut National de Recherche
en Informatique et en Automatique

TROISIÈME CONFÉRENCE EUROPÉENNE D'ERGONOMIE COGNITIVE

THIRD EUROPEAN CONFERENCE ON COGNITIVE ERGONOMICS

15 – 19 Septembre 1986
September 15 – 19, 1986

CITÉ UNIVERSITAIRE DE PARIS

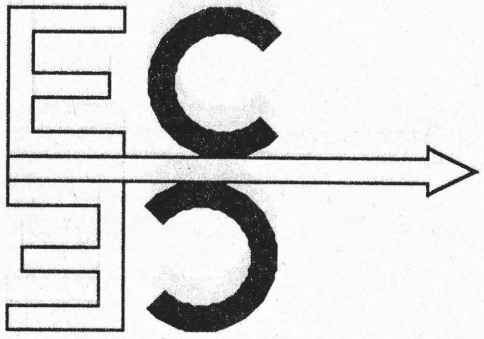
(France)

ÉDITION PROVISOIRE
PREPRINTS

ÉDITEUR:
Centre de Rocquencourt
Domaine de Voluceau-Rocquencourt-B.P. 105 78153 Le Chesnay Cedex

Centre de Rennes
IRISA-Campus Universitaire de Beaulieu-35042 Rennes Cedex

Centre de Sophia-Antipolis
Avenue Émile Hugues-06565 Valbonne Cedex



ECCE 3

Third European Conference on Cognitive Ergonomics

Troisième Conférence Européenne d'Ergonomie Cognitive

September 15-19, 1986
15-19 Septembre 1986

Cité Universitaire de Paris

PAPIER RECUPERE ET RECYCLE

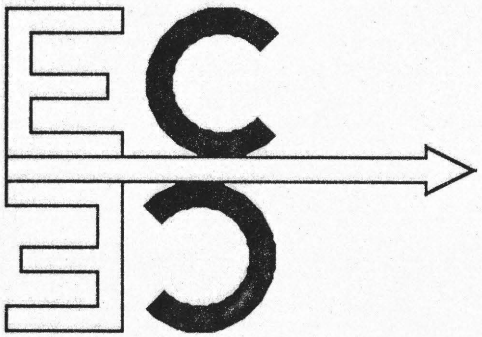
Comité d'organisation
Organization committee

P. Falzon (France)
T.R.G. Green (U.K.)
J.M. Hoc (France)
N. Streitz (F.R.G.)
G. van der Veer (Netherlands)

Comité de Programme
Program Committee

D. Ackermann (Switzerland)
P. Falzon (France)
T.R.G. Green (U.K.)
P. Hietala (Finland)
J.M. Hoc (France)
H. Pitariù (Romania)
N. Streitz (F.R.G.)
G. van der Veer (Netherlands)
Y. Waern (Sweden)

FONDATIONS PSYCHOLOGIQUES DE L'INTERACTION HOMME-MACHINE



PSYCHOLOGICAL FOUNDATIONS OF HUMAN-COMPUTER INTERACTION

CONTENTS

Session A1: Action Regulation

"Damn it, I've done it again": An investigation of action slips
R.K.E. Bellamy & T.R.G. Green (U.K.).....3

Action regulation and the mental operational mapping process in human-computer interaction: Does the interaction reflect dialog grammar?
D. Ackermann, J. Stelovsky & T. Greutmann (Switzerland).....11

Session A2: Computer-aided problem solving

Psychological aspects of the design process with computer assistance: CAD situations involve intensified self-planning
J.-C. Lebahar (France).....35

Computer aids for decision making in process control
L.P. Goodstein (Denmark).....43

Session A3: Programming

Cognitive strategies of novices and experts in computer program comprehension
A. Vihmalo (Finland).....63

Program comprehension and knowledge organization: the influence of acquired schemata
F. Détienne (France).....75

Understanding the cognitive difficulties of novice programmers: a didactical approach
R. Samurcay (France).....83

Representation of computer programs in memory: Comprehending and recalling computer programs of different structural and semantic complexity by experts and novices
D. Widowski & K. Eyferth (F.R.G.).....93

Session B1: Transfer and analogy

Analogical thinking in human-computer interaction
C.M. Allwood, M. Eliasson & T. Wikström (Sweden).....107

On using production systems for cognitive task analysis and prediction of transfer of cognitive skill
J.E. Ziegler, P.H. Vossen & H.U. Hoppe (F.R.G.).....115

The information-seeker's mental model of the content of a data base
L. Linde (Sweden).....129

Preknowledge and the efficiency of self-guided user's learning
R. Schindler & A. Schuster (G.D.R.).....139

Session B2: The encoding of semantics

On the role of memory in personal information management
M. Lansdale (U.K.).....157

Conceptual primitives and their representation in understanding complex operations (abstract only)
G. Rohr (F.R.G.).....165

Session B3: Humans as models for computers

Search strategies in internal and external memories (abstract only)
J.J. Beishuizen & F.M.T. Brazier (Netherlands).....169

Man-man communication: a model for computer-human communication?
P. Falzon (France).....173

Session C: The description problem

Human-computer interaction models and their use in computer system design
A. Whitefield (U.K.).....185

Development of mental models of office systems (abstract only)
G.C. van der Veer, M. Felt & B. van Muylwijk (Netherlands).....195

Assessing cognitive aspects of learning and using computer systems
M. Wilson & A. MacLean (U.K.).....197

Languages for users used both in familiarizing them with computing systems and in their communication with the systems: four cases
P. Tyllilä (Finland).....209

LA REGULATION DE L'ACTION

Session A1

ACTION REGULATION

"Damn it, I've done it again": An Investigation of Action Slips

R.K.E. Bellamy and T.R.G. Green

M.R.C. Applied Psychology Unit,
15 Chaucer Road,
Cambridge,
CB2 2EF,
UK

1. Introduction

Everyone knows the frustrations of repeatedly doing the wrong thing; everyone who uses interactive computer systems knows it especially well. Why do some systems seem to encourage mistakes? Can the problems be foreseen during design?

This paper discusses some theoretical frameworks of action control. Only by formulating a general statement can we avoid the trap of criticising particular systems. Such a framework must cover the whole of the action process from the formation of the plan through various levels of cognitive processing to the realisation of the plan in an action or a sequence of actions, and should ideally account for all types of natural actions. However, we shall concentrate on studies of slips – interesting phenomena in there own right, but also allowing insight into some of the mechanisms underlying the cognitive control of skilled actions.

It is important to differentiate slips from mistakes; mistakes are errors occurring during the formation of plans or intentions. Slips have been defined as "actions not as planned" (Reason, 1979) or "the error that occurs when a person does an action that was not intended." (Norman, 1981). Slips have usually been studied in the context of typing or speech. The studies have involved the collection of vast databanks of slips, followed by attempts to categorise the slips along various dimensions.

2. Theories of Action Slips

Theoretical frameworks primarily describe slips in various contexts, such as speech or typing, with implications about action control in general; others primarily describe action control in general, with little or no attention to slips.

Dell and Reich (1980) propose a spreading activation model of speech errors, in which signals flow through a network incorporating semantic, syntactic, and phonological connections. Activation spreads through the network from a node to all nodes directly connected to it, increasing as a node is due for activation and then decaying with time. At any point in time the most active node causes output. Random noise within the network causes occasional errors. This model produces many of the errors observed in naturalistic collections of speech slips, but some of

those observed in other domains appear not to be predicted – notably "capture" errors, in which a well learnt sequence of actions obtrudes on a similar but less familiar sequence.

Representations at the different levels of an action is a basic part of Norman's ATS (activation-trigger-schema), (Norman, 1981, 1983). This theory was developed as a general account of the action slips made by skilled typists. In ATS, an action sequence is represented by a parent schema and numerous child schema. Each schema has associated with it an activation value and a set of triggering conditions. An action is initiated with the activation of a parent schema, child schemas representing the component parts of the initial intention are then activated. These are selected when the conditions match those of the triggering conditions. Using this model Norman classified typing slips into three categories;

1) Mode errors, where the situation has been wrongly classified.

2) Capture errors, these occur "when there is an overlap in the sequence required for the performance of two different actions, especially when one is done considerably more frequently than the other" (Norman, 1981), e.g. consistently typing "cat" for "cp", perhaps because the "cat" command is used more frequently than the "cp" command. This becomes very irritating.

3) Faulty triggering of schemas: an example of this occurs when users perform the desired result, rather than the sequence of actions necessary to obtain the desired result, e.g. typing a "$" rather than the sequence of commands necessary to obtain the "$" prompt.

ATS is limited in its level of explanation, for instance it is difficult to see how ATS could move to the more complex world of text editing, where tasks are more diverse, the system state changes after commands, and a degree of problem solving ability is required (Green et al., 1984). A second problem with the ATS model concerns the use of activation to determine the sequence of events. Jordan(1985) has noted a problem in using activation to determine both the actions and the temporal ordering of the actions. When an action occurs more than once in a sequence, the additive effects of partially activated elements will cause the first occurrence of the action to move forward in time. Norman attempts to overcome this problem by postulating doubling and alternation mechanisms. This does not help in providing general mechanisms, and for repetitions across more than one element he needs to provide a parser to break up the sequence into separate pieces. Experimental data have shown that errors can spread across such sequences, thus indicating that the parser is not a satisfactory mechanism.

A particularly interesting developement has been the emergence of parallel connectionist models. Jordan (1985) proposes a theory of this type. Although it would take up too much space to describe the architecture of this class of model, the following central features may be noted. There is no distinct next action token, rather the actions are represented by distributed patterns of activation within a network. These representations are featural,and the representation of an action is often under-specified, so that all non-critical features are left as 'don't know'. These actions are fully specified at the time of output by reference to the features of nearby actions, which may proceed or follow them. This mechanism models the phenomenon of coarticulation in which the form of speech segment changes depending on the nature of the neighbouring segments.

E.g.

In the production of the word "freon", the vowels /i/ and /a/ tend to become nasalised due to the presence of the nasal consonant /n/ (Moll and Daniloff, 1971). These vowels are not usually nasalised in this context, as can be seen by comparison with the word "trio" (Jordan, 1985).

In a model of this type, slips of action would be expected to occur when an under-specified 'don't know' feature was wrongly assigned a value. The central contribution of this model to the study of slips are

1) Viewing actions as feature sets.

2) Predicting that non-critical features would 'diffuse' to nearby actions.

E.g.

Homology errors in typing (right finger, wrong hand) indicate that a single feature of an action is influenced by a nearby action (Fromkin, 1971; Grundin, 1983).

3. Previous Empirical Research

Laboratory studies attempting to demonstrate the effects postulated by theorists have been rare. Slips of the tongue, such as spoonerisms, have been demonstrated in the laboratory (Baars, 1980). The demonstration of such slips has lead to the suggestion that spreading activation models such as that described by Dell and Reich, described above, effectively account for the production of such slips.

The first attempt to demonstrate slips of action, other than slips of the tongue, appears to be a study conducted by Gillie (1984). He used a command structure isomorphic to the heavily-moded UCSD Pascal system, but he presented this as a scenario in which the user had to oversee the construction of a house. The aims of the experiment are not only to observe capture and mode errors in the laboratory but also to clarify the nature of the underlying cognitive processes.

Although the experimental task was clearly different from using the UCSD Pascal system, it was of similar complexity. We can therefore draw direct conclusions about the ease of use and how to improve many real life systems of similar complexity.

Gillie found that the directions of the effects were as predicted and there were nearly twice as many capture errors in the worst condition as in the best. However, these differences were not statistically significant; possibly his data was too scanty.

4. The Experiment

We now report a replication of Gillie's study, which is being conducted in collaboration with him. The principal idea behind this study is to train the subject on particular sequences and to see if these familiar sequences capture less familiar sequences. If so, will the frequency of capture errors vary with the set of possible sequences, the number of familiar sequences, etc ?

(a) procedure

The experiment is run online, using the same house-building scenario. Messages appear on the VDU containing instructions about who to contact and why, e.g. a typical message might read, "*You are with Smith the Estate Agent. You need an initial supply of bricks, contact Jones at the Material Suppliers*". The characters the subject needs to contact are arranged in a hierarchical structure, where : the higher level nodes represent functional specialisms (*professionals, services*, etc); intermediate nodes, where present, represent sub-specialisms (*tradesmen*, etc); terminal nodes name individuals (*Long, the Solicitor, Jones, Materials*, etc). 'Contact' is 'made' by typing sequences of letters, which denote the names of all the nodes between the subject's present position in the tree, and their destination, e.g. *P* (for *Professionals*), *H* (for *Home*), *B* (for *Builder*), *T* (for *Tradesmen*), *M* (for *Material Supplier*) and *J* (for *Jones*). Valid routes always involve vertical movement, horizontal shortcuts between nodes at the same level are not permitted.

The experiment has two stages. Stage 1 consists of a set of 36 training trials, for which the tree and the help of the experimenter are available. Stage 2 consists of 89 main trials during which the tree is not available. The general procedure for one trial starts with a message appearing at the top of the screen, telling the subject who to contact. The subject then has to type in the necessary letter strings. The subjects are asked to try to think of the whole sequence before starting to type, but if they get stuck of forget where they are at any point, help is available. When subjects type nothing for 5 seconds they are given the opportunity to get help. Help consists of being told where they are and where they can go from this point. It is strongly emphasised both in the experimental instructions and by the experimenter that the subject should try to type ahead of these opportunities to get help. After every 10 trials subjects are told how well they are doing. Subjects are given one chance to correct any mistakes, using a cancel key which can be pressed to cancel back one step, but not any further.

(b) design

Four groups are being compared. In each group the subjects are trained on a small number of routes in stage 1. During stage 2 these familiar routes are further trained, and at the same time less familiar routes are presented.

In Group 1 four routes are familiarised. In Group 2 only two routes are familiarised but receive twice as much use as in Group 1.

In Groups 3 and 4 the routes are the same as in Group 1 but the labels on the tree are changed. In Group 3, names of 'people' are chosen so that one letter (E) is

more frequent. More of the routes therefore end with the letter "E". Whereas for Group 4, the first letters of the names are the same at many different levels in the tree.

(c) measures

Analysis will concentrate on errors, in particular 'capture' errors (correct sequences partially replaced by a familiarised sequence with a shared segment) and 'over-run' errors (actual sequence is correct but with spurious characters past the correct termination). Speed, use of help facilities, post-experimental knowledge of the tree and subjects' reports will also be considered.

5. Predictions

The theoretical frameworks described above postulate various mechanisms by which capture errors are produced. Unfortunately many of the theories are incomplete and thus tight predictions are not possible.

Spreading activation (following the Dell and Reich model) appears not to predict capture errors at all; we have not yet explored in detail its predictions for other error types.

With regard to the remaining theories, an important point is that our subjects are not extensively practised. There is no question of 'automatic' behaviour in the Reason sense. This means that Reason's explanation, based on shifts between controlled and automatic processing, cannot account for any of the slips that we observe.

Norman's ATS model, as presented, is also primarily concerned with automised behaviour, but it would be possible to argue that it should predict similar phenomena with non-automised behaviour. In this case it would predict capture errors in all groups but it would not account for differences between Group 1 and Groups 3 and 4.

Jordan makes no attempt to account for slips and although it may be possible to incorporate such a mechanism into his theory, to do so would be a separate theoretical enterprise. We suggest that in the framework he offers slips occur at the point where features are 'filled in'. such errors could be labeled 'diffusion errors'.

6. Conclusions

What can these experimental results offer the designers of software systems? If the results are as expected, i.e. more or less like Gillie's, we conclude:

1) Contrary to some previous assertions, capture errors can occur at all levels of practise.

2) Slips appear to be increased by similarity between response sequences, although the precise nature and causal mechanisms are not yet established.

3) At least for novices, similarity is not restricted to sequences with large shared segments (capture errors). Configural features such as over-use of one symbol, reduce descriminability and increase slip frequency.

4) The typing and speech data suggest that slips can also occur when sequences are 'regularised' towards expected patterns. A notable example is the transposition of letters to obtain hand alternation in typing. These errors like doubling and homology errors suggest that individual actions should be regarded as feature bundles.

5) A first step for the conscientious designer in reducing slips might therefore be to represent each action as a feature set, to list the frequent action sequences in a system, and then to ensure that the other sequences are reasonably distinct both from the frequent sequences and from any regular patterns of alternation and doubling.

It would be gratifying to obtain considerably more data on the occurrence of slips. This would be an area where considerable design improvements might be possible.

References

Baars, B. (1980)
"The Competing Plan Hypothesis: A Heuristic Viewpoint of the Process of Errors in Speech", in H.W. Dechert and M. Raupach (eds.), *Temporal Variables in Speech,* The Hague: Mouton

Dell, G.S. & Reich, P.A. (1980)
"Toward a Unified Model of Slips of the Tongue", in V.A. Fromkin (ed.), *Errors in Linguistic performance,* Academic Press, New York

Fromkin, V.A. (1971)
"The Non-anomalous Nature of Anomalous Utterances", *Language*, 47, pp.27–52.

Gillie, T. (1984)
Slips: an Applied Psychological Approach, Unpublished M.A. Thesis, Sheffield University

Green, T.R.G., Payne, S.J., Gilmore, D.J. & Mepham, M. (1984)
"Predicting Expert Slips", *Proceedings INTERACT 1984, IFIP Conference on Human Computer Interaction*, London

Grundin, J.T. (1983)
"Error Patterns in Skilled and Novice Transcription Typing", in W.C. Cooper, (ed.), *Cognitive Aspects of Skilled Typing*, Springer-Verlag, New York

Jordan, M.I. (1985)
The Learning of Representations for Sequential Performance ,Unpublished Ph.D. Thesis, University of California, San Diego

Moll, K.L. & Daniloff, R.G. (1971)
"Investigation of the Timing of Relative Movements During Speech", *Journal of the Acoustical Society of America*, 50, pp.678–684

Norman, D.A. (1981)
"Categorisation of Action Slips", *Psychological Review*, 88, pp.1–15

Norman, D.A. & Rumelhart, D.E. (1983)
"Studies of Typing from the LNR Research Group", in W.C. Cooper, (ed.), *Cognitive Aspects of Skilled Typing*, Springer-Verlag, New York

Reason, J.T. (1979)
"Actions not as Planned: the Price of Automatisation", in G. Underwood & R. Stevens (eds.), *Aspects of Consciousness,* Academic Press

Proceedings of the Third Europeen Conference on Cognitive Ergonomics, Paris Sept. 1986

Action regulation and the mental operational mapping process in human-computer interaction: Does the interaction reflect dialog grammar?

David Ackermann*, Jan Stelovsky and Thomas Greutmann***

* Work and Organizational Psychology Unit, Prof. Dr. E. Ulich
** Institut für Informatik (Institut of Computer Science)
ETH Zürich (Swiss Federal Institute of Technology), CH-8092 Zürich (Switzerland)

Abstract

A major research area of cognitive ergonomics focuses on the adaptability and individualization of interactive systems. In cooperation between psychology and computer science, we have conducted two experimental studies to investigate individual differencies in constructing mental models and their influence on action regulation: 1) in an game driven by command macros 2) in adaptable interactive system with numerous applications. Both studies involved experiments with varying dialog grammars and the analysis of the observed interaction. Our findings show that for a given dialog grammar there is a large variety of different user's action grammars. We conclude that there is no optimal dialog grammar and that the adaptability to the task-, dialog- and individual-specific aspects of the human-computer interaction is a necessary and feasible requirement for design of interactive systems.

1 Theoretical context

1.1 Psychological aspects

For an optimal development of the adult personality in working live it is indispensable that individual differences are taken into account as Ulich [1978] stated in his principles of differential and dynamic work design: There has to be the possibility to adapt and enlarge existing production systems sensu Ulich (or task-solving processes), to create new and switch in between different ones thus permitting individual working styles. Triebe [1980] and Zülch and Starringer [1984] have shown that such individualized work can increase efficiency and poductivity.

A theoretical explanation for the mentioned results is required. Based on the hypothesis of "semantic information transfer" [W. Krause 1982], Ackermann [1983] formulated the hypothesis that difficulties with the human-computer interaction during task-solving processes are caused by the discrepancy between individual mental representation and cognitive styles on the one hand and the given operations of the scope of action prescribed by the software on the other hand. Therefore, cognitive and action regulation processes are the most important aspects in our research.

1.1.1 Cognitive processes

The "mental representation" of task, tools and prescribed operation methods is involved in the ongoing task-solving process. External stimuli have to be transformed into an internal representation. Goals have to be derived and retransformed into actions. As a dynamic and task-specific process the mental operational mapping is defined by goals, scripts and plans [Schank & Abelson 1977] and depends on motivation, knowledge in the sense of "mental representation" and cognitive skills [Johnson-Laird 1983]. Oschanin [1976] called this mediating process "operatives Abbildsystem" (task related mental model) but did not formulate an explicit model. Ackermann and Stelovsky [1986] pointed out that the actual mental operational mapping process - as a cognitive process - is affected by individual differences.

1.1.2 Action-Regulation

Since Miller, Galanter and Pribram [1960], psychologists assume that action can be described best by "feedback-loops" combined to hierarchically organized structures [Hacker 1978]. In order to control the "action", only the highest level of the feedback-loop - mentally represented by a "chunk" [Cube 1968] - has to be activated. Therefore we have investigated the composition of complex action-units and the resulting scope of action [Ackermann 1985]. We could show that the mental operational mapping process and the resulting command programs are individually different and that problems in dialog are due to mismatch between the mental operational mapping processes and the prescribed methods.

1.1.3 Model of human-computer interaction

We summarize the psychological theoretical background outlined above in the model of human-computer interaction depicted in Fig. 1. During the task-solving process, data-processing is the relevant common aspect of both, the computer and the human. In the physical world, the computer offers tools to manipulate the physical data. The tools' user-interface can be viewed as a dialog grammar formed by a set of rules for accepted input and generated output. This dialog grammar defines the objective degrees of freedom in the physical scope of action. In this paper, we will concentrate on the input rules only. On the human side the dialog grammar is perceived and matched with the task. This matching is influenced by cognition, motivation, action knowledge and intentions. The last step in the action regulation - the mental-operational mapping process - generates an observable result: the action grammar. Thus, the action grammar allows us to infer the subjective mental scope of action.

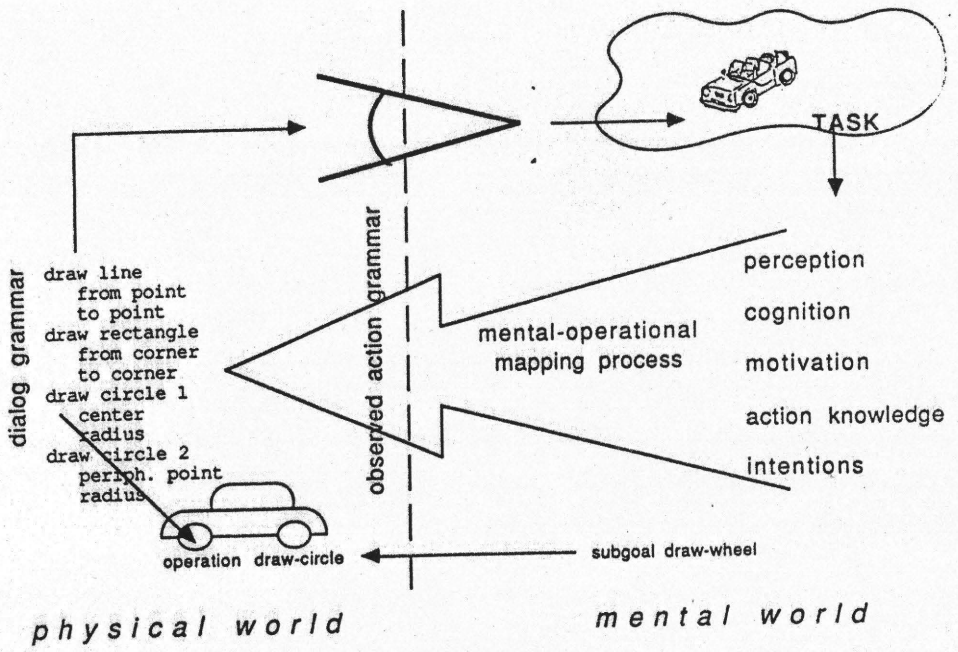


Fig. 1: The model of human-computer interaction

1.2 Computer science aspects

Human-computer-interaction can be regarded as a communication process. Moran [1981], Reisner [1981], for example have created grammatical systems to describe the dialog interface. For years, command languages have been the major dialog scheme for the communication between man and the computer. Some research efforts have focused on the formal specification of a command language and its use as a system specification and design language. Nowadays, representation of commands as menus and their selection by pointing (e. g. with a mouse) has captured the users acceptance of interactive systems. It also became fashionable to use icons and windows for representing data and command objects.

The difference between these approaches is the way how the objects are identified. In the command language approach, command names and parameters must be typed in to be identified. Because the command language is based on a grammar, it permits the specification of complex, structured commands. A complex grammar, however, imposes restrictions on possible user input. If the restrictions must be memorized, the cognitive complexity [Ackermann and Stelovsky 1986] and thus also the mental workload increases. In the menu approach, commands labeled by names or icons are displayed on the screen and used for input by pointing. Since the grammatical constraints are reduced to the minimum, defining complex command sequences is tedious. Menus can be intuitive for the

user, but their use tends to oversimplify structure and grammar which are inherent to highly interactive applications. Moreover, a flat command structure can restrict the subjective action-scope of the user [Spinas 1986]. In short, a command language is precise and expressive, but rigid if its grammar must be remembered.

Adaptive interactive systems allow us to offer varying dialog grammars and investigate their influence on the observed action grammars their structure and suitability for the task-solving process. Such experiments help us to develop and refine the requirements for the design of computer dialogs.

1.3 Goals of our research

- Psychological aspects

As a consequence of the principles of differential and dynamic work design, computer dialogs must be adaptable according to the user´s competence, intentions and needs [Ulich 1985, Ackermann 1986c]. Therefore we need (1) to know how users map the dialog schema offered by the computer to an action grammar and (2) to capture the individual differences.

- Computer science aspects

We need (1) to formulate requirements for the design of an adaptable interactive system and (2) to investigate the consequences of different user-interfaces (e.g. command structures) on the mental operational mapping process and hence on the outcome of the task-solving process.

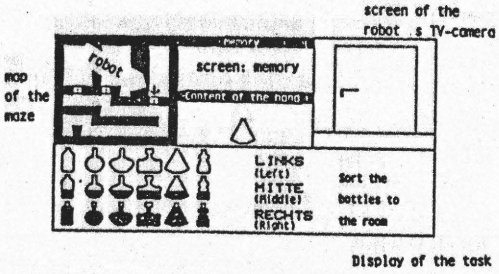
We suggest an iterative approach to cognitive ergonomics, where the psychological investigation with the help of an adaptable system results in design requirements for user-interfaces. These requirements in turn should serve as the input for further psychological testing.

2 Experimental study 1: Individual optimality of dialog grammars

2.1 The game PRIMP-1 (Nickname "Robi Otter")

To investigate mental aspects of task-solving processes, we have constructed the computer game PRIMP (Programmable Robot for the Investigation of Mental Processes). PRIMP allows the player to pilot a virtual robot in a maze displayed on the video screen (Fig. 2). The robot's view is also shown on the screen. The player's task is to sort bottles according to their contents. The robot´s actions can be directed with six

basic commands which can be combined into iterations and selections. Moreover, commands can be embedded into macros which we call command programs. The names of the command programs can be chosen freely; they usally reflect the intentions and the goals derived by the player and indicate its scope of action.



Display of the task

Basic commands

Go	The roboter moves one field forwards
Turn	The roboter turns 90 degrees to the left
Open	The roboter opens the door
Take	The roboter takes the bottle from the rack
Deposit	The roboter deposits the content of his hand to the empty rack
Remember	The bottle of the rack in sight will be represented in the memory on the screen
Compare	The roboter compares the bottle in the rack with that represented in memory

The commands can be combined with an editor. There are the following possibilities:

until	door, wall, rack, free, empty, equal
if	door, wall, rack, free, empty, equal
not	

Example
Right = turn; turn; turn;
GT = Go until not free; Right; if door open;

Fig. 2: A section of the manual for PRIMP showing the basic commands and the screen showing the robot's view

In a first test series we could show that different cognitive styles - depending on abilities as on goal setting - lead to different "command-structures" and "action grammars" [Ackermann 1983, 1986a)] In a second study 6 students developed command programs, tested the programs created by their colleagues and reported their results [Ackermann, 1986b].

The developed command programs show differences in the use of the possible scope of action (Fig. 3), thus reflecting individual strategies. Nevertheless, implementation of command programs comparable with respect to the offered objective scope of action can differ considerably (see command programs AROBBY) and HCOMMANDS) in Fig. 3). The general objective in design of the command programs was to prevent own errors and errors of possible users. The other tendency was to find an optimal strategy. Fig. 4 compares the player´s ranking when playing their own and the other command programs. In most of the cases, the efficiency is better with own command programs.

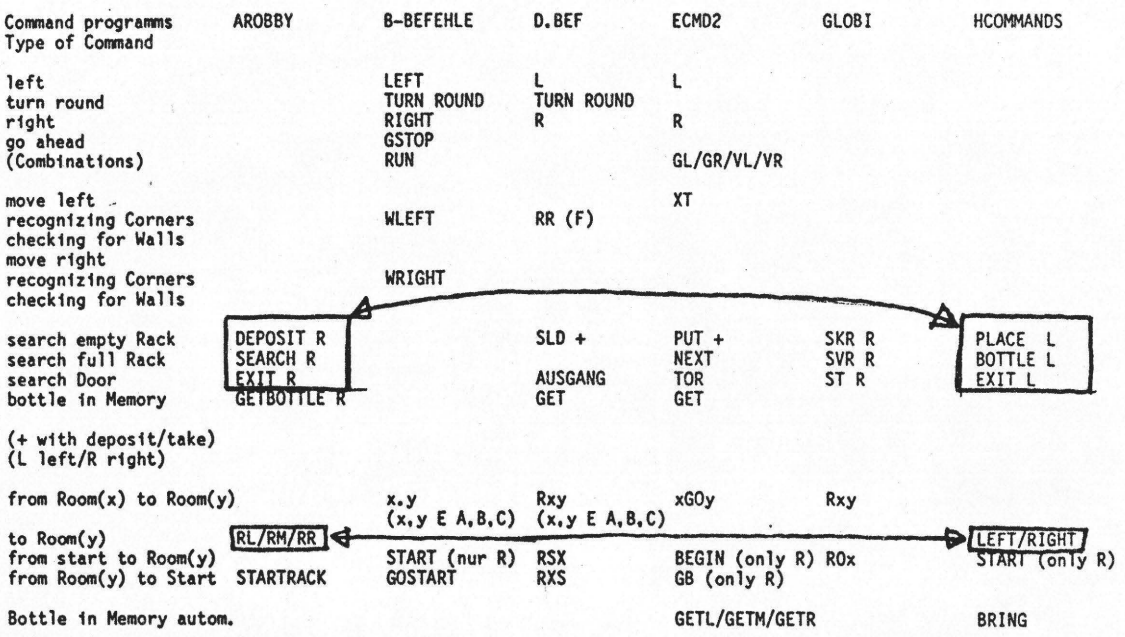


Fig. 3: Overview of the developed command programs. Comparable commands are marked.

Subject (Player)

Parameter	A	B	D	E	F	G
Time/ Bottle	1.	2.	4.	1.	2.	1.
Commands/ Bottle	1.	6.	3.	1.	2.	1.
Errors/ Bottle	4.	3.	1.	1.	1.	1.

Rank orders of the individual developed commandsets in 6 trials (individual commandset + the best trial of each other)

Fig. 4: Players' ranking

2.2 Analysis of the mental operational mapping process with prescribed command programms.

Analyzing the reported results we formulated the hypotesis that the developed command programs not only mirror the designer's action grammar but also represent which subjective action grammars he wanted to propose to the user, thus defining the degrees of freedom in the user's scope of action. Furthermore, we assume that the mismatch between the designer's intentions and user's mental model is the main source of troubles in the human-computer interaction.

To test this hypothesis, we gave these command sets to other students without experience in creating command programs to see how prescribed

command programs would be accepted and mentally represented and how this would influence their action grammar. The students were told to sketch their playing strategies on paper or to report them verbally. Fig. 5 shows how differently the prescribed command programs were represented. While the drawing on the left is spatially oriented and based on the geography of the maze (in the sense of a cognitive map - [Tolman 1948]), the algorithm on the right shows mental representation of the command flow. Another interesting result is documented in Fig. 6: The tendency to enlarge the scope of action prescibed by the command program according to individual needs (for example the need for control). The graph on the left is the designer's description of his command set and the drawing on the right is the user's interpretation of the offered scope of action. Curiously enough, this tendency can be observed even when the given command program does not allow to expand the scope of action or does not offer it explicitly. On the other hand, one player reduced the given command program which offers numerous possibilities to the commands he found necessary: Fig. 7 shows its interpretation (Fig. 3 includes the original command program ECMD2).

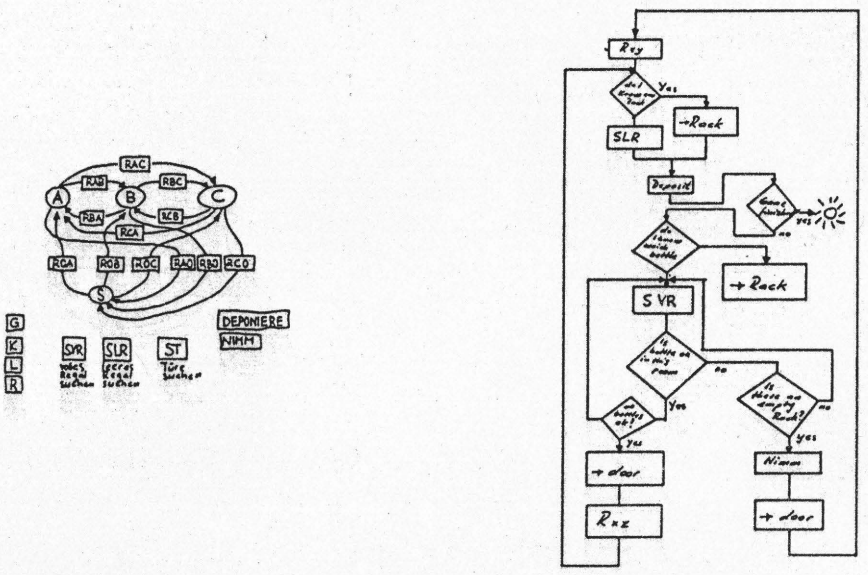


Fig. 5: Different representations of the command program GLOBI

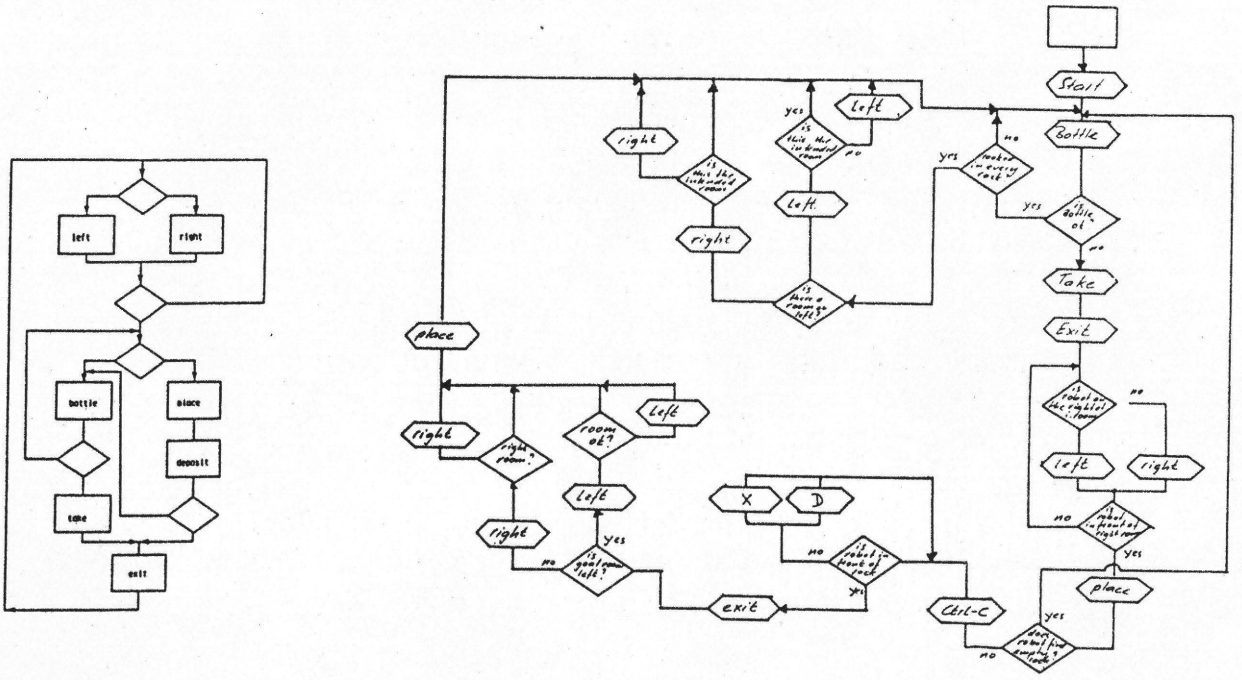


Fig. 6: Intended (on the left) and subjective (on the right) scope of action of command program HCOMMANDS

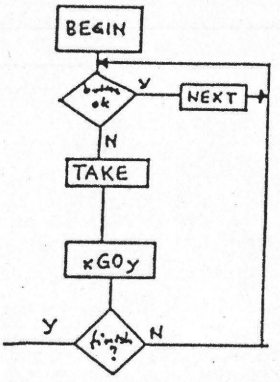


Fig. 7: Subjective scope of actions limited to the indispensable operations of command program ECMD

Beside the discrepancies in the use of the possible scope of action, the verbal reports and sketched representations document individual differences in other aspects. The degree of abstraction, for instance, varies considerably: One player pointed out that the goal setting was his main interest and elaborated seven levels of goal settings reflecting the state of the game. Although one player employed all given command programs without noticeable difference in efficiency, several test persons refused to try some command programs (in particular the command program AROBBY) or adapted or wrote their own versions that suited better their intentions.

2.3 Conclusions

The dialog grammars offered by an interactive system are a subset of all possible operations and can often be combined to more powerful commands, e.g. macros. The macros increase the level of abstraction and structure the dialog grammar to suit better the structure of the task. It depends on the user, however, how he can make use of the offered dialog grammar. The degrees of freedom of his scope of action are defined by the subjective action grammar. These action grammars are the observable means that mirror the underlying mental representation.

The presented results show wide range of different action grammars for equivalent command programs. We conclude that the task is represented in mental models which constitute different subjective scopes of action and vary individually in form and content. The analysis of the player's reports shows that the action grammars are influenced by cognitive styles, motivation, memory capacity and other individual skills.

3 Experimental study 2: The correspondence between dialog and action grammars

In the experimental study 1, we have shown that different cognitive strategies and command grammars can lead to improved efficiency. in the task-solving process within the limited context of a game. Can these results be generalized as a guideline for a complex system with numerous different and more sophisticated interactive application programs? How does the opportunity to adapt the dialog structure to the user's own preferences and abilities influence his interaction, efficiency and the quality of his work? Furthermore, can a designer offer such dialog grammars that would induce a mental model that is more suitable for the given task, thus promoting the necessary learning effect? To investigate these questions we have conducted a further series of experiments with the interactive system XS-2.

3.1 The XS-2 system

The eXperimental System 2 [Stelovsky et al. 1985] defines a hierarchically ordered command language grammar based on regular expressions. This interactive system incorporates both, the command language approach and menus [Stelovsky and Sugaya 1985]: The resulting command structures together with their syntax form command trees and are displayed as menus to the user (Fig. 8). The user can conveniently adapt XS-2 different hardware settings or change the grammars that determine the structuring of his data [Stelovsky 1984] Moreover, he can tailor the commands of an application according to his needs. A command can be renamed, the commands in a menu can be regrouped or subdivided into a menu hierarchy.

Several alternative command structures for the same command can be introduced and tested within one application.

All these structural changes are done with one standard tool, which is also used to define the commands for a new application. The corresponding actions are simulated by an automatically generated program. This approach reduces the effort of application design: the commands for a new application can easily be developped and the existing commands can be reorganized without changing the application's implementation.

3.2 A pilot study: classification of influence of dialog grammars

3.2.1 Experimental design

In order to analyze and classify the influence of dialog grammars on the action grammars, we made a pilot study with the XS-2 system. We selected three different tasks for a group of 6 students who were willing to paricipate in our study. The first task was to do some computations with a pocket calculator "TRPol" (Fig. 8). In the second task, they had to use the "graphic editor" to draw a treble clef with the necessary lines. Designing their own initials with the "font editor" was the third task. They received one hour of introduction and a manual with a tutorial before they started to work. Furthermore, they could use paper and pencil to document their thoughts and problems. The XS-2 system was enhanced by a logfile facility that enabled us to record all their actions.

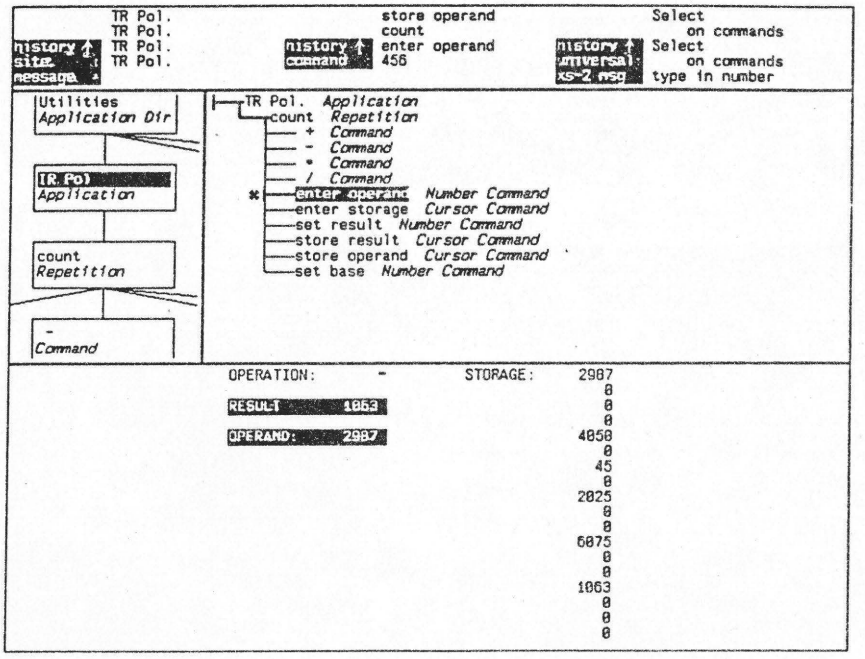


Fig. 8: Screen copy of "TRPol", windows of XS-2 system

The calculator's dialog grammar was aimed at simplifying continuous computations using the same operations (e.g. chain of additions). Fig. 9 shows the designer's intention: the emphasized path shows that such chain calculations require only typing in numbers. Although the calculator could be used in the normal way, i.e. with the infix notation, the command structure did not explicitly offer this working style and rather suggested the use of the "prefix polish notation". This suggested dialog schema was unfamiliar to our students who are used to commercial pocket calculators with either algebraic or the postfix polish notation.

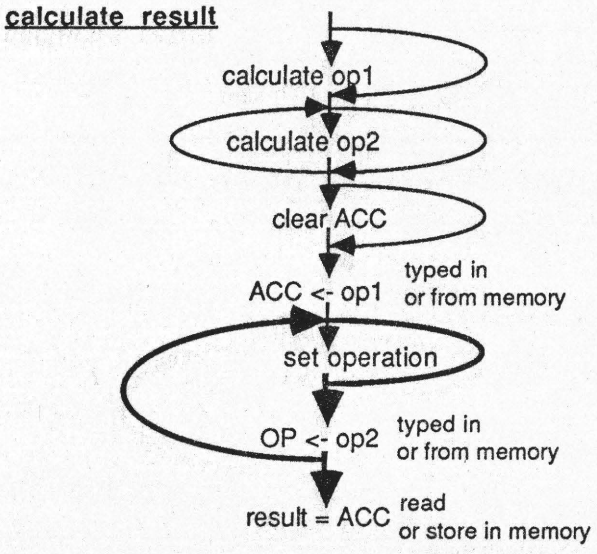


Fig. 9: Dialog grammar of "TRPol" according to the designer

The original command structure of the "graphic editor" is presented in Fig.10 The functionality of the editor had a limitation: once a sequence of lines and arcs was finished, it could not be modified. The drawing of such a sequence, however, could be aborted or the entire picture could be deleted. The command structure was intended to simplify a rather complex task: the drawing of a continuous chain of lines and arcs. This intention lead to an unfamiliar concept that cannot be found in a common graphic editor. Nevertheless, a single line, arc or circle could be drawn using the same command structure.

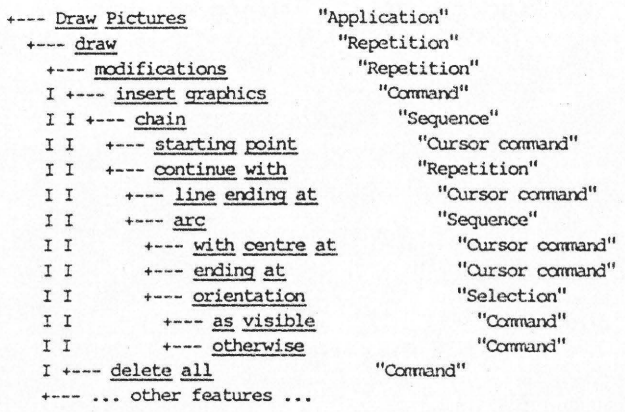


Fig. 10: Command structure of the "graphic editor"

These dialog grammars of "TRPol" and "graphic editor" are similar in their design philosophy: both command structures were intended to simplify the input of complex data objects (sequences of additions or lines). In contrast, it was not obvious how these concepts should be applied to define primitive objects (such as an operation on two numbers or drawing a circle). Indeed, both command structures were developed by the same person.

3.2.2 Results

In the calculator tool, the students tried to manage the offered dialog schema according to their previous experience and its mental representation. Restructuring the given task to the unfamiliar suggested prefix polish notation made more difficulties than the command activation itself. This lead to a predominantly trial-and-error behavior until the restrictions in the dialog were learned. Sometimes the calculator was operated using the not obvious infix notation as a normal pocket calculator.

In the graphic editor, the first step - to open the appropriate data field, was the most difficult aspect of the task. The correspondence of data to applicable commands caused some errors because the students tried to apply experiences or heuristics aquired with other software concepts. After they overcame this difficulty, no more dialog-specific problems were observed. Few students had hardware- and individual-specific problems with the mouse. Working with the font editor made no difficulties - again provided the students were able to open the appropriate data field. The action style changed from trial-and-error to goal-oriented command sequences.

The pilot study revealed that the influence of a prescribed dialog grammar can be classified into task-specific, dialog-specific and individual-specific aspects. Thus, it is important that, during the design of the user-interfaces, a dialog grammar can be tailored to suit a particular task, corrected to minimize the difficulties with the dialog

itself, and last but not least adapted with respect to the personal preferences.

3.3 Extended study: Consequences of different command grammars

In our further investigations with XS-2, we decided to focus on the task-specific, dialog-specific and individual-specific aspects of the human-computer interaction and concentrate on the question how different dialog grammars influence the user's actions. Therefore, a second pocket calculator "TRNorm", which offers "normal/algebraic" computations explicitly, was implemented. The tasks were redefined as well: Instead of drawing clefs, the participants were asked to draw a car. Since everyone knows how a car looks like, our subjects should not have any task-specific problems. Another task was introduced: a simple table calculation with a newly implemented spreadsheet application.

3.3.1 Experimental design and research objectives

29 students participated in this series of experiments. They started with the conventional calculator "TRNorm", changed to the drawing program and afterwards had to do a comparable calculation with the original calculator "TRPol". With the two calculators we offered different working environments with different command grammars to the user. They had to solve the same task - the sum of 3 easy computations - with both calculators. Our objective was to investigate how the action grammar varied depending on different working environments, whether and to which degree people could adapt to prescribed dialog grammars. The cars had first to be drawn with a single prescribed command structure.

Then, the participant's problems, expectations and needs were evaluated and the "graphic editor" augmented by additional command structures adapted to their suggestions. In a follow-up experiment, 4 students were asked to draw two cars. One car should be drawn using a new, sophisticated command structure proposed by an experienced XS-2 user. In contrast, they were free to choose among any of the commands in the enlarged set to draw the other car. Beside the interdependence between dialog and action grammars, we could test how the variety of dialog grammars met the requirements of differential design of interactive systems.

3.3.2 Results of the calculator study

As the entire sequence of user's actions was recorded on a log-file, we were able to reconstruct the motion on the command trees and derivate the underlying action grammars in the form of a set of rules. Fig. 11 contrasts the command structure of "TRNorm" on the left with that of

"TRPol" on the right. In Fig. 12, the variety of different action grammars is generalized to five classes of working strategies. The classes are ordered according to their optimality with respect to the number of rules to be applied. For instance, applying strategy 1) needs less steps than 3) and 4). The results of "TRNorm" and "TRPol" are compared in Fig. 12: 15 persons adhered to their computational strategy but 12 persons learned that they could improve their efficiency when they assimilated another strategy. To give an example, some participants took advantage of the fact that they can reuse the previous result in "TRPol". In contrast to the pilot study, where numerous participants had dialog-specific problems with the unfamiliar command structure of "TRPol", only participant V) had such problems. On the other hand, the participant L) restructured the task into chains of computations to make the optimal use of the stored operation facility in "TRPol" in the sense of the tool's designer.

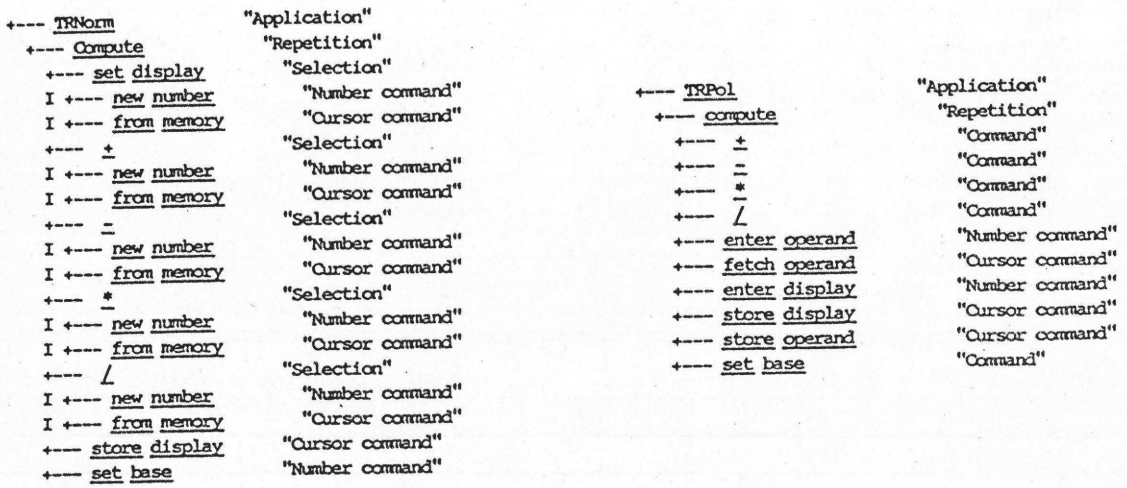


Fig. 11: Command structure of "TRNorm" and "TRPol"

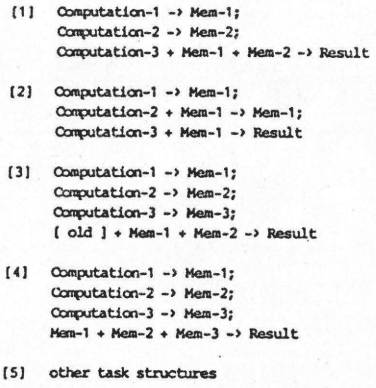


Fig. 12: Five basic strategies to solve the calculator task (the sum of three easy computations)

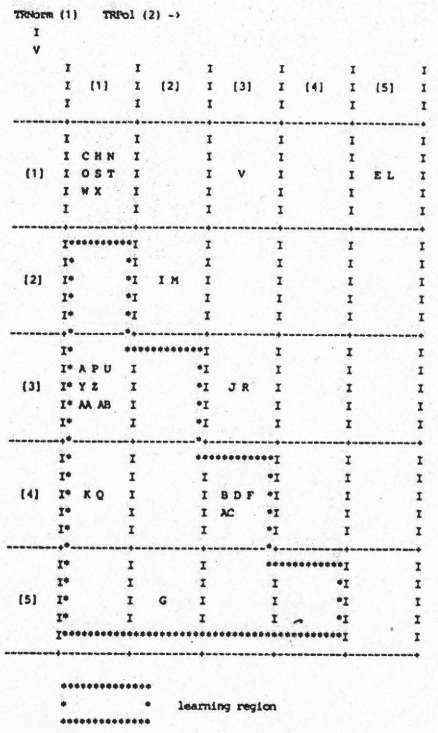


Fig. 13: Comparison of the basic strategies in "TRNorm" and "TRPol"

3.3.3 Results of car drawing

The resulting cars were evaluated by a jury and the three with the best and the three with the worst rating are depicted in Fig. 14 to demonstrate their aesthetic qualities. Certain cars look like professional sketches (AC), others rather resemble cans. Several participants tried to make the best out of lines and arcs that did not fit their intentions but some failed... The common explanation was a misleading expectation about what the commands can do. Analyzing the process of drawing car E), we can see how the participant managed to make the best of unintentionally drawn lines... "Dialog-driven" is an apt word to characterize this drawing. The participant F) erased the whole picture after the first steps and tried to redraw the car. But the dialog-specific problems appeared again.

The comparison of the participants F) and AC) in the calculator and car drawing studies brings an interesting result. They employed the same strategies in both calculators requiring unnecessary operations, no progress in learning how functionality of the dialog could be better employed for the given task (Fig. 13). But in between the two computations, they had to draw the car, a task that was according to the jury's ranking solved best by AC) and very inadequately by F). To interpret this result, we interviewed AC) and found out that he was very fond of drawing cars in the childhood. Thus, his task-specific skills permitted him to assimilate easily an unfamiliar dialog grammar.

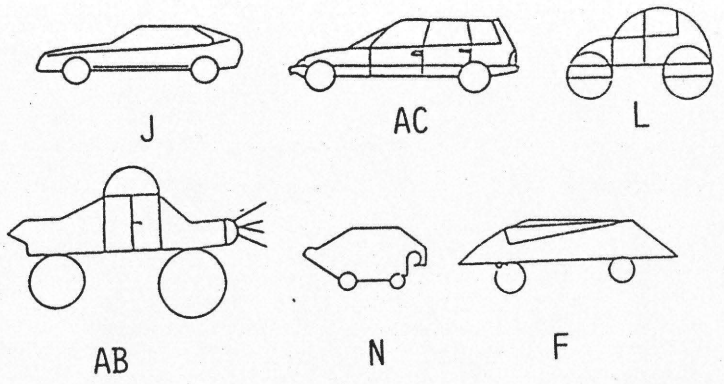


Fig. 14: Three of the "best" and three of the "worst" cars

In the follow-up study, the graphic editor was enlarged according to the users' suggestions by the command structures shown in Fig. 15. On the left we see the main menu offering a variety of commands tailored for different subgoals (e.g. a single line or circle) as well as different strategies to define them. The differences in these strategies are mirrored in the parameter sequences (i.e. circle defined by center first versus circle defined by the peripheral points first).

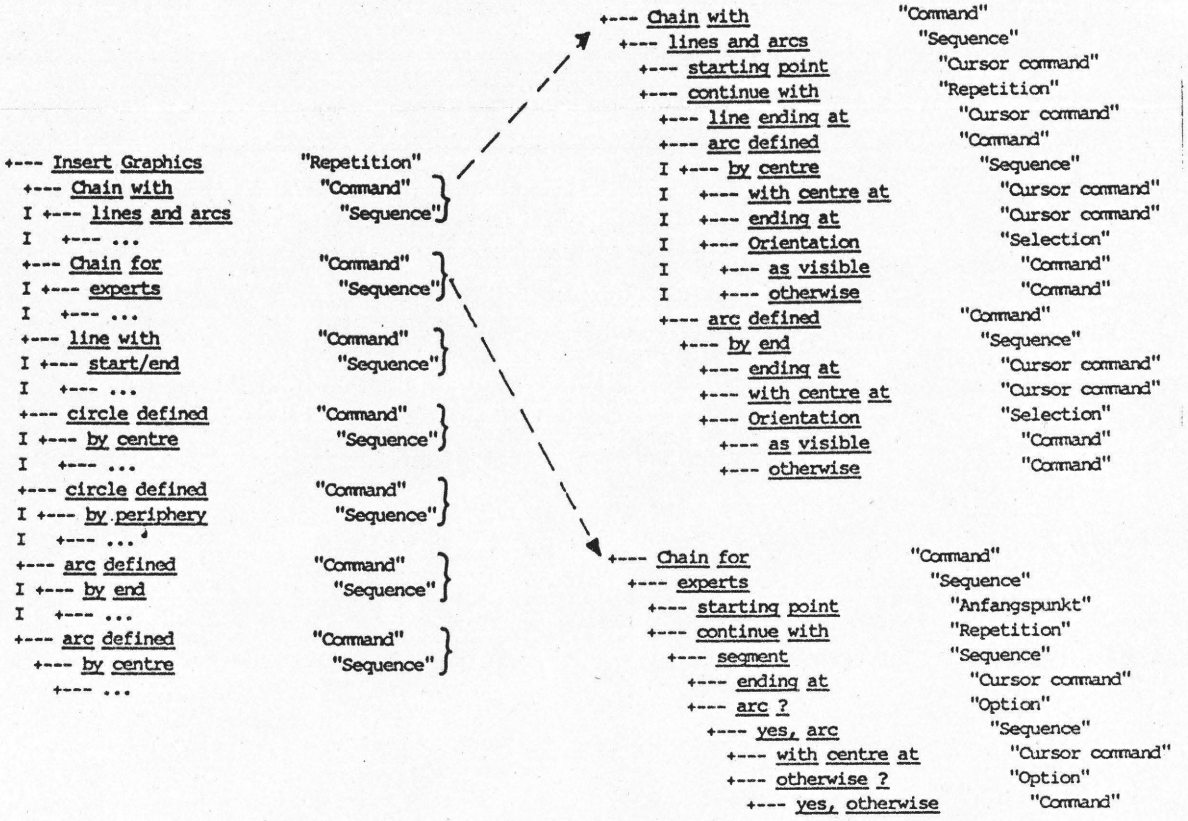


Fig. 15: Parts of the command structure of the enlarged graphic editor

Dialog-specific problems occured when participants had to use the prescribed command structure, (e.g. person E) in Fig. 16. When they were free to choose among commands, all of the participants were using varying command structures in correspondance to the subgoal and individual needs and skills. To give an example, wheels were drawn using one of the commands for circles. Interestingly enough, all participants were using one of the commands for drawing chains of lines and arcs when they found it most appropriate to accomplish a subgoal e.g. when drawing the outline of the car's body. On the other hand, both such commands were employed according to individual preferences. The participant AE), for instance, assimilated the sophisticated command structure and preferred it when drawing the second car - except for the wheels. Hence, we conclude that it is desirable to provide a larger set of alternative commands tailored to the task-specific requirements. Furthermore, an enlarged command set accomodates better individual working style and reduces dialog-specific problems.

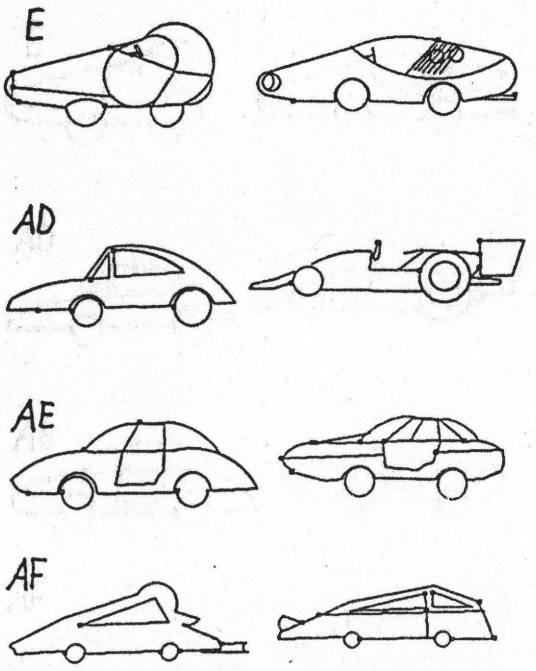


Fig. 16: Cars drawn with the enlarged graphic editor

4. Conclusions

During our experiments, we offered different dialog grammars which organize the scope of action into different degrees of freedom. According to our results, it is impossible to induce only one action grammar even when the command grammar is very limited. The intended possibilities are redefined by the user. As the experiments with PRIMP-1 show, participants construct their own scope of action very individually to achieve the personally optimal action grammar, thus improving their efficiency. The resulting mental models of the task and the interaction are individually different in style, degree of abstraction and kind of representation.

These results are further confirmed by the calculator and the car-drawing experiments. They demonstrate that the same task-solving process can be applied although the dialog grammars differ. Conversely, one dialog grammar will induce various action grammars, except for the trivial case where the dialog grammar does not offer any degrees of freedom. The resulting action grammars depends on the individual user's mental processes. We try to outline the relations in Fig. 17.

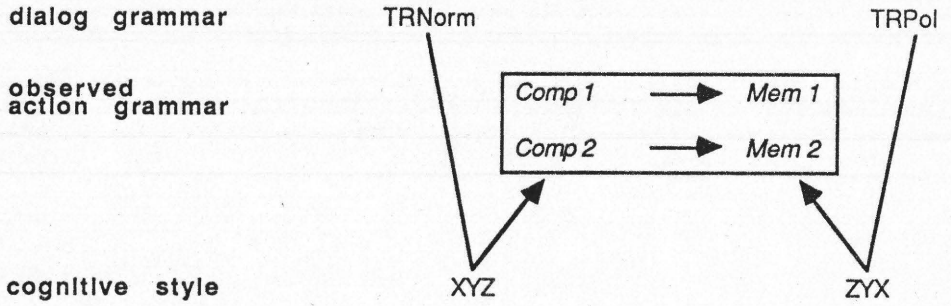


Fig. 17: The relations between dialog grammar, action grammar and cognitive style

In the car drawing experiment, it is interesting to see how certain participants manage to make the best out of unintentionally drawn lines. This shows the flexibility of the ongoing mental operational mapping process and adaptability to the restrictions imposed by the dialog grammar. But the results depend on individual skills and despite of the user's efforts certain participants fail to reach their goal. Thus, the fact that user's cognitive flexibility allows him to assimilate the prescribed dialog, is no prove of good dialog design.

The classification and distinction of task-, dialog- and individual-specific aspects of dialog design proved useful not only in the analysis of the experiments' results but also as a general tool for dialog design. We can confirm Young's results [Young 1981]: different dialog grammars are better suited for task-specific requirements. On the other hand, the comparison

between the participants F) and AC) in the calculator and car drawing studies indicates that task-specific skills can also compensate for the dialog-specific problems.

We can confirm our hypothesis, that problems in human-computer interaction are due to incompatibilities between mental operational mapping process and the offered dialog grammar. On the other hand, different dialog grammars can induce better strategies if they are more cognitivly suitable with respect to the users individual mental operational mapping process thus confirming the principles of differential and dynamic design of production systems sensu [Ulich 1978]. Individual dialog schemas adapted to the user's needs improve efficiency and prevent "dialog-driven" results. In Fig. 18, we depict the interdependencies among the three aspects of dialog design.

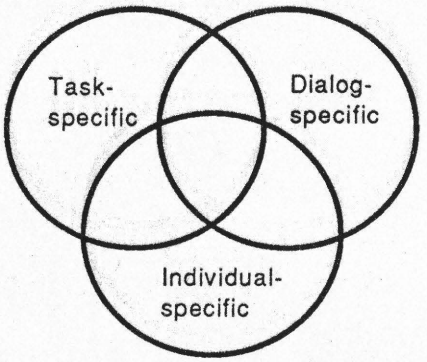


Fig. 18: The three aspects of dialog design

One of the most important results of our experiments is the influence of dialog grammars on the learning process. As we could demonstrate in our calculator study and later confirm in the car drawing study, offering first familiar and simple dialog grammars and then a large set of dialog grammars with different degrees of abstraction and adherence to the task enlarges the subjective scope of action. The participants are motivated to explore the offered choices, learn the suggested strategies and then apply those that fit best their intentions. Such a step-wise learning process can even overcome dialog-specific problems with very specialized dialog grammars.

Hence, we conclude that an interactive tool should offer a variety of dialog grammars to optimize the efficiency of a variety of users, tasks and tools. Therefore, we stress the importance of the adaptability in interactive systems. The example of XS-2 demonstrates the feasibility to construct such systems. Dialog design is an iterative process, it should be dynamic with respect to the individual user and to his skills, preferences and needs. Such a design will support his learning thus improving his qualifications.

References

Ackermann, D.: Robi Otter oder die Suche nach dem operativen Abbildsystem. Interner Bericht der Studienarbeiten im SS 1983.

Ackermann, D.: Untersuchungen zum Prozess der Handlungsregulation am Beispiel der Mensch-Computer-Interaktion. Interner Bericht der Studienarbeiten im SS 1984.

Ackermann, D.: Untersuchungen zum individualisierten Computerdialog: Einfluss des Operativen Abbildsystems auf Handlungs- und Gestaltungsspielraum und die Arbeitseffizienz. In: Dirlich, G., Feksa, C., Schwatlo, U. & Wimmer, K. (Hrsg): Kognitive Aspekte der Mensch-Computer-Interaktion. Ergebnisse eines Workshops vom 12./13. April 1984 in München. Berlin: Springer, 1986a.

Ackermann, D.: A pilot study on the effects of individualization in man-computer-interaction. 2nd IFAC/IFIP/IFORS/IEA Conference on Analysis, Design and Evaluation of man-machine-studies. Varese, September 1985. London: Pergamon Press, 1986b.

Ackermann, D.: Mentale Repräsentation und Handlungsregulation im Handlungsspielraum. Untersuchungen zum Prinzip der differentiellen und dynamischen Arbeitsgestaltung. Unpublished Dissertation, University of Bern: 1986c, in Prep.

Cube, v. F.: Kybernetische Grundlagen des Lernens und des Lehrens. Stuttgart: Klett, 1968.

Hacker, W.: Allgemeine Arbeits- und Ingenieurpsychologie. Bern: Huber, 1978

Johnson-Laird, P.N.: Mental Models. Cambridge University Press, 1983.

Krause, B.: Semantic information processing in cognitive processes. Zeitschrift für Psychologie, 1982, 190, 37-45.

Krause, W.: Problemlösen - Stand und Perspektiven. Zeitschrift für Psychologie, 1982, 190, 18-36 und 141-169.

Miller, G.A., Galanter, E. & Pribram, K.H.: Strategien des Handelns. Stuttgart: Klett, 1973.

Moran, T.P.: The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems. International Journal of Man-Machine Staudies, 1981,15,

Oschanin, D. A.: Dynamisches operatives Abbildsystem und konzeptionelles Modell. Probleme und Ergebnisse der Psychologie, 1976,59,37-48.

Reisner, P.: Formal Grammar and Human Factors Design of an Interactive Graphics System. IEEE Transactions on Software Engineering, 1981,SE-7.

Schank, R. & Abelson, R.: Scripts, Plans, Goals and Understanding. Hillsdale, N.Y.: L. Erlbaum, 1977.

Stelovsky, J.: XS-2: The User Interface of an Interactive System. Disseration No 7425 ETH Zurich,1984.

Stelovsky, J.: User - Tailored Dialog - Just a Slogan? Proceedings 8th International Computing Symposium, Florence March 1985, 345-352. Amsterdam: North Holland, 1985.

Stelovsky, J., Nievergelt, J., Sugaya, H, & Biagione, B.: Can Operationg System Support Consistent User Dialogs? Experience with the Prototype XS-2. Proceedings ACM Annual Conference. October 1985, Denver, p. 476-483.

Stelovsky, J. & Sugaya, H.: Command Languages vs Menue or Both? Software Ergonomie 85. Stuttgart: Teubner, 1985, 129-141.

Spinas, Ph.: VDU-Work and user-friendly man-computer-interaction: Analysis of Dialogue Structures. In: Proceedings of the 2nd Symposium on Work Psychology, Espoo, 1986 (in Press).

Tolman, E.C.: Cognitive maps in rat and man. Psychological Review, 1948,55, 189-208.

Triebe, J. K.: Aspekte beruflichen Handelns und Lernens. Eine Feld- und Längsschnittuntersuchung zu ausgewählten Merkmalen der Struktur und Genese von Handlungsstrategien bei einer Montagetätigkeit. Unpublished Disseration, University of Bern, 1980.

Ulich, E.: Ueber das Prinzip der differentiellen Arbeitsgestaltung. Industrielle Organisation. 1978, 47, 281-286.

Ulich, E.: Arbeitspsychologische Konzepte für Computerunterstützte Büroarbeit. Spektrum 14, 1985.

Young, : The Machine inside the Machine: User's Models of Pocket Calculators. International Journal of Man-Machine Studies, 1981, 15.

Zülch, G., & Starringer, M.: Differentielle Arbeitsgestaltung in der Fertigung für elektronische Flachbaugruppen. Zeitschrift für Arbeitswissenschaft, 1984,38,211-216.

L'ASSISTANCE A LA RESOLUTION DE PROBLEME

Session A2

COMPUTER-AIDED PROBLEM SOLVING

PSYCHOLOGICAL ASPECTS OF THE DESIGN PROCESS WITH COMPUTER ASSISTANCE: C.A.D SITUATIONS INVOLVE INTENSIFIED SELF-PLANNING.

Jean-Charles Lebahar
G.A.M.S.A.U
Ecole d'architecture de Marseille

This work analysis is based on two fields of study: C.A.D and design. Design can be understood as human thinking during the genesis of operative knowledge .This knowledge describes a virtual object, a car or a building.

Design activity already existed before the advent of CAD , as a specific topic for the psychological approach of intellectual work, with its own cognitive riddles.

CAD has modified more or less deeply design strategies.The difficulty of the analysis methodology aiming to explain design work with CAD lies in the technological cognitivism of the computer and more exactly in the software: this new tool - CAD - thinks and draws.

Architects and designers are working with a thinking tool when they build models!

The building of an operative drawing (sketches, draft or model) of an object is a different problem than the creation of the object itself.

Drawing consists in the processing information concerning a material object with assistance of graphic and geometrical problem solving.(JC Lebahar, 1983).

Design is connected with technological, financial, functional, aesthetic references and decisional strategies. Drawing is connected with management of semiotical informations: design and drawing are deeply distinct , but producing a drawing is a design activity also.

When a system has a specific design (and/or) drawing ability, the user must adjust his design problem space to this ability, in addition to the accomodation of his own human and technical ability.

For example the control of a design step process by a cross section is one of the possible operative choices in solving the problem of positioning an architectural element in a general unity of a building . There are several other possible cross-sections as well as other means.

Nevertheless, it's possible for cognitive analysis to set reference marks in this complexity:

-a semiotical design medium

-inherent reasonings in design problem solving
-a strategic management of the fields of possibilities;

THE SEMIOTICAL DESIGN MEDIUM

is an operative scaled model more or less in analogical relation with the virtual object, as well as in a semantical relation with it. It's a combination of the geometrical invariants of the object and of classes of human thinking. It is an result of operational equilibrium between constraints of matter and of designers' competence, between space and reasoning.

This double status of the semiotical medium allows operational reasoning to test by anticipation hypotheses and choices affecting the behaviour or the feasability of future object.It's a "simulation medium".

THE REASONING IMPLIED BY DESIGN PROBLEM SOLVING stems from knowledge, operational experience and skill in co-ordinating heterogeneous contents and requirements (utility action, aesthetics, ergonomics, cost , technical validity).

THE STRATEGIC MANAGEMENT OF FIELDS OF POSSIBILITIES reduces information processing. This economy consists in indentifying and outlining operative fields of possible solutions, reduced by goals and methods of the designer.This is the synchronical aspect of information processing . The diachronical aspect is principally a problem of self-planning, or in other words, of programming activity (conscious or unconscious).

This point has been evinced in the work of J M Hoc (1981) who has inspired my approach of auto-regulation in the design process since 1981.

More over self-planning of tasks and activities, is the central problem when in order to understand CAD practice, we are brought to compare it to unaided design activities(JC Lebahar, 1985, 1986).

1 DESIGNING AN OBJECT can be viewed as transforming a body of informations concerning this object -approximate in its precision, scope and technological conformity - into a set of precise and complete data, the communication of which will enable their more or less immediate translation into a material object: a building or a car.This process reveals a gradual and total reduction of the fields of possibilities inherent to the preliminary representations of the problems generated by the cognition of the object.

At stake in the formation of these representations is a possible check of the process, excess cost of retroactive errors or inaccurate decisions due to choices unwillingly left to builders or other users of the draft.

The fields of spatial action embodied in graphical practice and modelling: an operative instrument for managing uncertainty.

Our examples derive from architecture , industry and industrial design. These areas differe mainly on account of the specific influences of their technical and economical environments, which in turn condition the process of design itself: in architecture the variety of forms, the computation of materials and the specification of construction sites; in industry, the methods, the extend of series and the variety of technological approaches.In addition organisational conditions set industry radically apart in implying the existence of precise constraints integration procedures within the project.

In this case design problems (analysis of requirements, sketches, dimensionig of elements, constructive representation) are solved by a collective and organised agent . For each step and each task, there is a specialized sub-agent (marketting comitee, design office, ergonomics, CAD service).

Consequently the cognitivism of individual designers using CAD must be distinguished from users of CAD acting in a collective situation: the latter are specialised in a task, which is a sub-task in the global activity of individual designers. Timing and requirements of task are different in this context.

2 REDUCING UNCERTAINTY AND SELF- PLANNING

However, whichever the case , the designer is brought to gradually reduce his uncertainty by emitting hypotheses through projective sketches or topological diagrams guided by partially measured forms which, in metrical and euclidian terms, thus assuring the preservation of validated decisions. This process was named by me "simulation"; "cognitive simulation" is expressed by drawing simulation (JC Lebahar, 1983).

The more the hypothesis is uncertain, the more indistinct is the drawing; conversely, the more it is validated, the more the dimensioning tends to secure an exclusive decision. Each of these types of decisions contributes to the final representation (working drawing in coded and dimensional projection) of the object, their coordination allowing for their compatibility.

This compatibility is one of the foremost results sought for by the designer's anticipatory behaviour as evidenced in the self-planification of this conception process (for example, in avoiding to prematurely fix a surface or a volume to as not to limit his manoeuvrability in the event of an ulterior decision depending on this first one).

As his conception work progresses, the designer is led to complement his representation which in turn expresses its own constraints and in which areas not having been anticipated generate sub-zones of learning and finding.

In brief, all begins with probabilistic constructions contextualised by certainties (e.g in all- defined room containing specified furniture; an iron supposed to include a particular steam generator) and all ends with the completion of a deterministic puzzle.

Simulation occurs at different stages in the process . In architecture it takes place during the intermediate phases of elaboration.

In industry and design, due to the mechanisation of construction, simulation can also occur in the final stage when prototypes and models can actually threaten the whole process which moreover entails dynamic functioning (on the opposite, there is not engine in a house even though there may be a tiger).

In this context (without CAD), planning is pre-adjusted by graphic methods (from fuzzy to precise, from possible to certain, from shape to dimensioning, from cross-section to graphic or solid models).

It results from work habits and is often implicit. Several steps in the design process are explorational because problem solving accomodates to different kinds of requirements and task-environnments, whether known or unknown.

Sketches and approximative solid models express hypotheses of cognitive simulation. They appear through a relative desorder among a relative planning.The plannification modifies itself along the process, pressed by the action of unpredictable events and masqued requirements.

3 The increase of auto-planification due to CAD situations

This increase ,through simulation, implies a differenciation of the auto-regulative status of the process, explains the different roles played by CAD in design work.

The CAD systems lack power for architectural work but, mainly and most important , they are weak in graphic performance concerning intermediate simulation.

In manual work, this simulation implies focusing on the internal reasoning of the designer whose graphic signals, expressing this reasoning, are informatively very reduced, susceptible to be quickly thrown out on behalf of another sketch because at the same time the spatial reasoning changed. Drawing and thinking are connected in an analogical functioning.

These signals must therefore be measurable, superposable, reversible and this, in a manner such that their process of production be brought to the level of an subconcious reflex; all consciousness of graphic activity being a hindrance of internal reasoning.

In industry, final simulation of an object, of its physical behaviour and of its computed spatial composition is made possible through CAD .Simulation process speeds render all former programing constraints obsolete.

Consequently, the situation is different from that of architecture where, in addition to the static nature of the objects, the mode of social production is not the same.

What stands out, as regards design problems, is that in all cases, comparative analysis shows that the best CAD operators are also the most proficient in all the fields of geometry (descriptive and trigonometry).This mental pre-adjustment of data acquisition behaviour enables them (architects, telephone set and motor car designers..)to avoid falling into the majority of errors.

In other words, CAD users know the main spatial invariants of their design project before the construction of executive model with CAD aid. They organise their own activity to avoid difficulty , errors and time loss in operational research.But mainly they cannot simulate with CAD because rythm and centration on constructing the simulation medium with this tool draws mental work away from imagining the object's transformation.This transformation takes place in the world of visual images as internalized actions and mental schemes ans sketches(JC Lebahar, 1985).

The mind cannot exist in two worlds in the same time: on the one hand , in graphic and logical procedures to produce CAD informatons, on the other, to design attributes and shape of an unknown object.

A cognitive simulation medium must be productive without mental cost, as a reflex, in order to materialize concepts with the rythm and speed of a mental flash.

CAD action requires measuring, cleanness, exactness and planning.

This planning modifies the contents of simulation work in anticipation: in other terms , when designer builds problem space, requirements and sketches, he forecasts the CAD intervention, that's to say that the geometrical procedures and design style of this tool are always particular and directive.

The planning resulting from CAD entails retro-active and pro-active influences.(JC Lebahar , 1985).

CONCLUSION

Consequently we hold that CAD generates methods and order in design works, through the tendencial exclusion of zones of fuzzy

arrangements.Now, designers in CAD situations avoid problems not solvable by CAD, as soon as they begin to build general sketches that is to say as soon as they develop internalized action (JC Lebahar, 1983).

The same problem is solved in other circumstances if the designer is a collective worker: design work is strictly separated from constructive modelling within two distinct teams of workers.The competence of CAD systems is revealed in standards libraries (specific elements to be assembled: joineries, walls, symbols, geometrical elements, pre-adjusted drawings) ; in procedures; in co-ordinate systems and labelling of axes; in operational interdicts or strict requirements (the impossibility of using concave volumes in some systems).In particular the structure of these libraries strongly pre-adjust the designer's fields of operational possibilities.

The most operative constraint is the preliminary dimensioning of shapes and qualities, concerning the design goal. This approach of the object to be designed promotes an analytical and a posteriori mental process consisting in assembling pre-adjusted elements.

At last, users must give exact and definite spatial informations to the CAD system.Therefore it's impossible for them to improvise spatial models with CAD. They must know and design geometrical sketches and dimensioned lines and figures, to control the graphic result visualised on the screen and stocked in memory.They need accurate skill in geometry, because for each plan and cross section, the control of their action requires constructive procedures and identification of geometrical results as well as the system means of geometrical transformation.

Their drawing activity becomes a geometrical activity . Observation shows that good operators are virtuosoes in main geometrical constructions. An other kind of cognitive planning! CAD practice replaces spontaneous skills with rationalized constructions. Method supersedes analogical skill.

REFERENCES CONNECTED WITH THE THEME

HOC JM 1981 LA DIMENSION ORGANISATRICE DE LA PLANIFICATION DANS LA RESOLUTION DE PROBLEME. DOC.LABO DE PSYCHO. DU TRAVAIL. EPHE. 41 RUE GAY LUSSAC 75005 PARIS.

HOC JM 1981 LES HABILETES MENTALES DANS LE TRAVAIL. LE TRAVAIL HUMAIN. T.44 N° 2/1981.

LEBAHAR JC 1983 LE DESSIN D'ARCHITECTE :SIMULATION GRAPHIQUE ET REDUCTION D'INCERTITUDE MARSEILLE ED.PARENTHESES. 134p

LEBAHAR JC et BERTHELOT; BERNARD; MICHELUCCI ET GANGNET; KOMORN; MANAGO. 1985 ETUDES EN CFAO : ARCHITECTURE ET BATIMENT.ED. HERMES PARIS.

LEBAHAR JC ET COLLAB 1986 ARCHITECTURE,DESIGN ET CONCEPTION DE PRODUIT. ED; PARENTHESES . MARSEILLE (sous presse).

LEBAHAR JC 1985 LA CAO EN ARCHITECTURE ACTES DE LA QUATRIEME CONFERENCE EUROPEENNE SUR LA CFAO ET L'INFOGRAPHIE PP.89-105.MICADO-HERMES.PARIS.

LEBAHAR JC 1985 LE TRAVAIL DE CONCEPTION EN ARCHITECTURE: CONTRAINTES ET PERSPECTIVES APPORTEES PAR LA CAO. LE TRAVAIL HUMAIN.TOME 49. N°1/1986.PUF PARIS.

LEBAHAR JC 1985 LES DIMENSIONS COGNITIVES DU TRAVAIL DE CONCEPTION EN ARCHITECTURE AVEC UTILISATION DE LA CAO .IN ACTES DU COLLOQUE"L'ERGONOMIE EN INFORMATIQUE " UNIVERSITE LIBRE DE BRUXELLES.NIVELLES NOV.85.

LEBAHAR JC 1986 LE DESSIN TECHNIQUE . IN TECHNOLOGIES IDEOLOGIES ET PRATIQUES.VOL.5/4/6:1. UNIVERSITE DE PROVENCE. AIX EN PROVENCE.

LEBAHAR JC 1986 XAO: QUELLES COMPETENCES POUR LES UTILISATEURS? ACTES DE L'AFCET. 3-5 JUIN 86 PARIS.

COMPUTER AIDS FOR DECISION MAKING IN PROCESS CONTROL

L.P. Goodstein
Risø National Laboratory
DK 4000 Roskilde, Denmark

INTRODUCTION

The introduction of computers in process control has greatly influenced the possibilities for the treatment of information in the control rooms of industrial plants - plants which themselves are becoming more centralized and complex with greater potential for serious consequences in the event of (human) errors. In addition, computers are being introduced in the design phase of these plants with the result that data bases are being established which also will be of great value in connection with operational planning and management.

This advanced technology has also had the effect that industrial plants are becoming more automated and do not rely on human intervention for the control of normally planned and predicted situations. However the operating crew still has to cope with all the tasks which are badly structured as well as the unforeseen events and disturbances which can arise. In this way, the role of the operator has shifted from being a manual controller to that of a systems manager, problem solver and decision maker; hence the term **supervisory controller.**

However, in reality, the operating crew is part of a decision-making **team** and, in accordance with the functional allocations of the designers, plays certain assigned roles in dealing with the process. Use of the word **team** reflects the fact that the supervisory control of such complex systems has actually to be a cooperative effort within a group consisting of the designers,

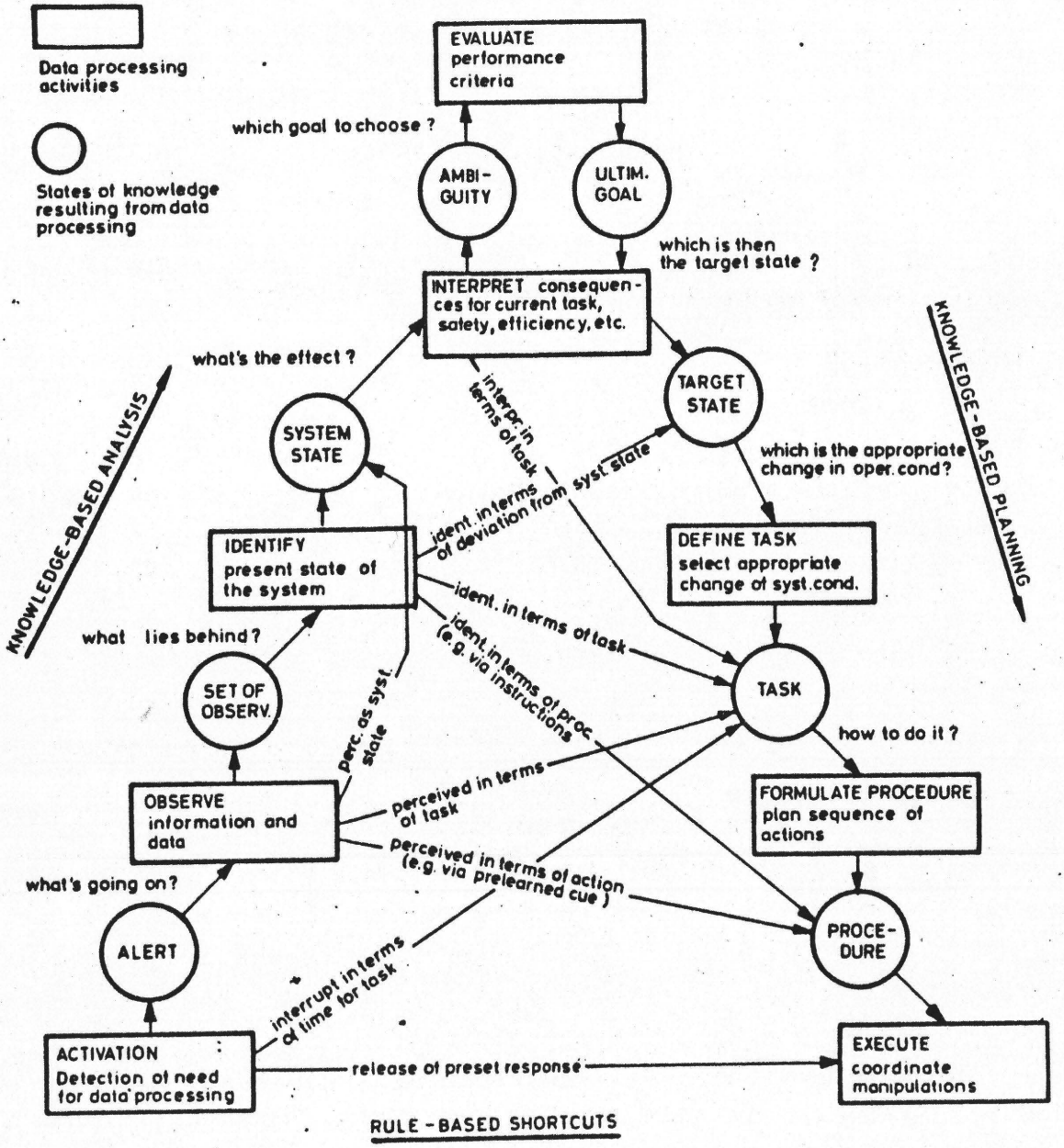


Figure 1. Schematic map of the sequences of information processes involved in a control decision. Rational, causal reasoning connects the "states of knowledge" in the basic sequence. Stereotyped processes can by-pass intermediate stages. Adopted from Rasmussen, 1976.

the automatic (computer-based) control system and the operating staff. This three-way arrangement arises from the fact that the decisions of the designers are embedded in the automatic system as well as the training of the operators. Thus the prerequisite for a successful cooperation is that the designer ensures that the computer and the operators can work together in a positive way by taking advantage of their different and complementary information processing abilities and their different knowledge about the system, the environment, the goals, etc. This probably means that the framework in which this cooperation takes place should involve a dynamic allocation of decision functions between the "partners" with appropriate feedback and communication facilities between them.

DECISION MAKING FRAMEWORK

Fig. 1 - taken from Rasmussen (1976) - describes a framework which encompasses the various types of information processing sequences that characterize a decision maker's (dm = human or computer) activities in dealing with a problem. These include the completely rational approach where the dm "climbs up the ladder" on the left hand side while **observing**, making an **identification** of state, **interpreting** the implications and **prioritizing** goals. Thereafter the dm "climbs down the ladder" on the right hand side in connection with **planning** and **carrying out** the appropriate set of actions in order to achieve or reach the chosen **target** state.

The diagram indicates as well alternate paths from the initiation of a decision making activity to its conclusion. For example, in situations which seem to be familiar to the dm, shortcut paths can exist in the form of a large number of association rules - e.g. IF xx, DO yy. It is these which form the basis for the veteran or experienced dm's behaviour in dealing with the object system and often comprise the "shallow expert knowledge" which is acquired for building current expert systems.

LEVELS OF ABSTRACTION

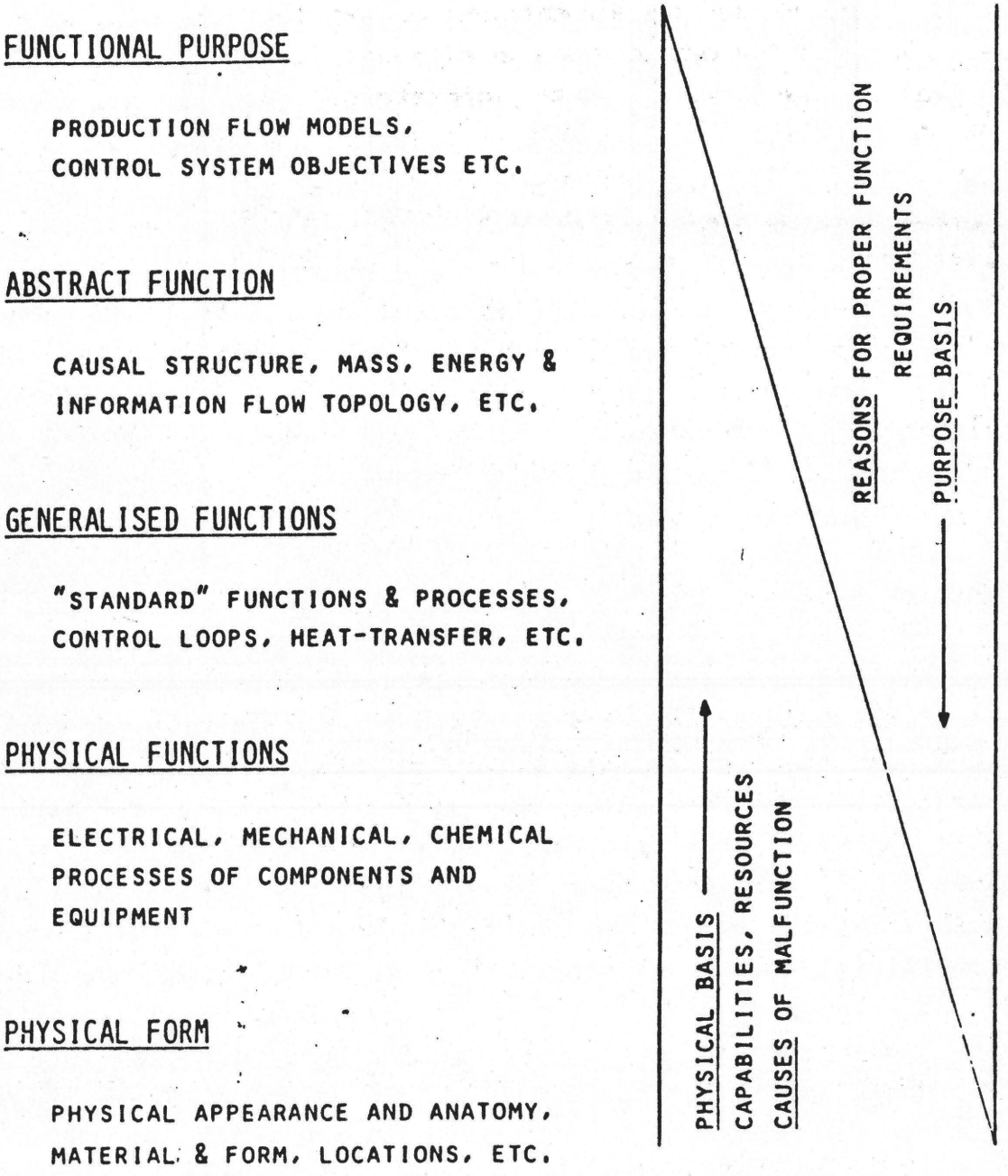


Figure 2. The system properties considered in man-machine interaction can be described at various levels of abstraction, representing the physical implementation and functional purpose in varying degrees.

Thus, while in the real world the rational type of decision making behaviour is probably mostly restricted to novices feeling their way, much can be said for trying to force experienced users to resort more to this type of response through a suitably designed display and controls interface. The reason for this is of course that the repertoire of quick and effective responses is not all-inclusive for every possible situation. Thus, in their interaction with the system, operators **first** have to be made to realize the inherent risk in a hasty response and **thereafter** be supported in more of a knowledge-based decision making sequence in order to solve their current problem.

HIERARCHICAL DECISION SPACE

Another important consideration in decision support is the **decision context**, i.e., the representation of the **problem space**. Rasmussen (1984a) has dealt with this in detail in his description of the different levels of abstraction and decomposition which a human may use to cope with the complexity of a technical system, depending upon the situation and the phase of a decision task.

In the abstraction, or means-end, hierarchy (see Fig. 2), the low levels of abstraction are related to the available set of physical equipment which can be used to serve several different purposes. Models at higher levels of abstraction are closely related to specific purposes, each of which can be served by different physical arrangements. This hierarchy is therefore useful for a systematic representation of the many-to-many mappings in the purpose / function / equipment relationships which represent the context of - and are a necessary precondition for - supervisory decision making.

When considering a control task at any level of the hierarchy, information about the proper function, target states and answers to the question **WHY** are obtained from the level above, while

information about present limitations and available resources, i.e. answers to the question **HOW**, can be gotten from the level below.

In the present context, we are in particular interested in those human functions in man-machine systems which are related to corrections of the effects of faults and other disturbances. States can only be defined as disturbed or faulty with reference to the planned or intended functions and purposes. Causes of improper functions depend on changes in the physical world. Thus they are propagating - and are explainable - **bottom-up** in the hierarchy. In contrast, reasons for proper functions are derived **top-down** in the hierarchy from the functional purpose. During plant operation, the task of the supervisory controller - man and/or machine - will be to ensure by means of proper actions on the system that the actual state of the system matches the target state specified for the intended mode of operation.

This task can be formulated at any level in the means-ends hierarchy. During plant start-up, for instance, the task moves bottom-up through the hierarchy. In order to have an orderly synthesis of the overall plant function during start-up, it is necessary to establish a number of autonomous functional units at one level before they can be connected to one function at the next higher level. This definition of functional units at several levels is likewise important for establishing orderly separation of functional units for shut-down and for reconfiguration during periods of malfunction.

During emergencies and major disturbances, an important supervisory control decision is the selection of the level in means-ends hierarchy at which to consider the control task. In general, highest priority will be related to the highest level of abstraction: First consider overall consequences for plant production and safety in order to judge whether the plant mode of operation should be switched to a safer state (e.g. stand-by or emergency shut-down). Next, consider whether the situation

can be counteracted by reconfiguration of functions and physical resources. This is a judgement at a lower level representing functions and equipment. Finally, the root causes of the disturbance are sought to determine how the failure(s) should be corrected. This involves the level of physical functioning of parts and components. Generally, this search for the physical disturbance is of lowest priority (not considering the role which knowledge about the physical cause may have for the understanding of the situation).

Thus, when a disturbance has been identified and the control task located at a certain level of abstraction, depending upon the perceived situation, the supervisory control task includes the determination of the appropriate target state derived top-down from the operation mode chosen together with an identification of the available functional resources and limits of capabilities, established bottom-up in the hierarchy.

SHARED DECISION MAKING

As stated earlier, decision making in supervisory control is a shared enterprise comprising the designer, the automatic computerized system and the operating crew. Since the designer may not be sure about his ability to foresee the necessary control responses for all possible disturbances (or because management and/or the regulators insist), there is need for a representative on site - the operator - who has to be able to take over in a competent way. The operator's supervisory control task is indeed in many respects a completion of the system design for the particular, perhaps infrequent, situation being dealt with. As a consequence, the operator will probably need information about the problem space underlying the design and the designer will have to communicate this kind of information to the operator. This can take place through the system itself; i.e., by means of the information gathered, processed and stored in the computer-based instrumentation and/or directly through training, manuals and instructions.

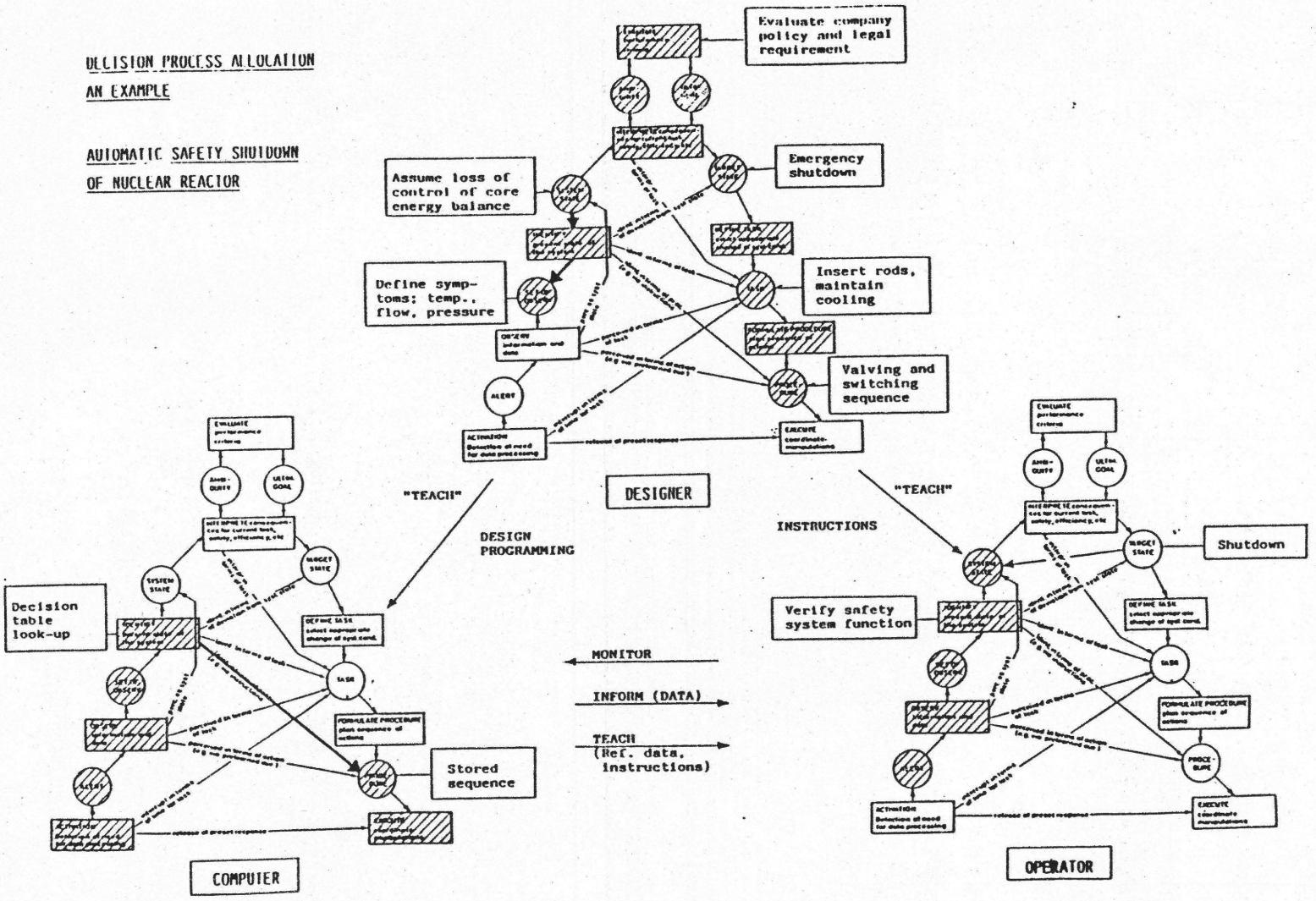


Figure 3A. The designer, the operator, and the process computer are all parts of a control decision. The figure illustrates their roles (shown hatched) in an automatic safety shut-down, in which they act "in parallel"; each with a diagnostic task based on different strategies.

Sharing decision making among operator, designer and computer will result in a complex communication depending on the extent to which the designer wants his representatives on site to take an active part.

Criteria for the role allocation may be related to reliability requirements, to the resources and conceptual model available to support the processing by the different agents, or to the actual system state data accessible to them. Thus the designer can choose to automate certain protective functions and thereby take over one of the supervisory sub-goals; he can choose to preanalyse certain phases of the decision sequence through use of his analytical models and issue instructions to the operators; or, finally, a lack of data can lead him to transfer his conceptual tools to the operators in the form of facilities for interactive decision making. These modes of role allocation will be considered in more detail in order to formulate the communication requirements.

Thus Figs. 3a and b indicate how the basic framework of Fig. 1 can be replicated to reflect how a designer might intend two typical situations to be allocated among the partners.

Fig. 3a illustrates the situation where the designer will ensure a safe **automatic** shut-down of a plant in situations which are critical because of time constraints and/or potential consequences. (The various assigned roles are shown hatched.)

The designer postulates sets of undesirable states (e.g., loss of control of core energy balance) and for each he specifies:

- a set of process attributes/symptoms which are definitive for the particular undesired state

- desired target state (e.g., shutdown)

- the tasks to be performed to achieve the target state

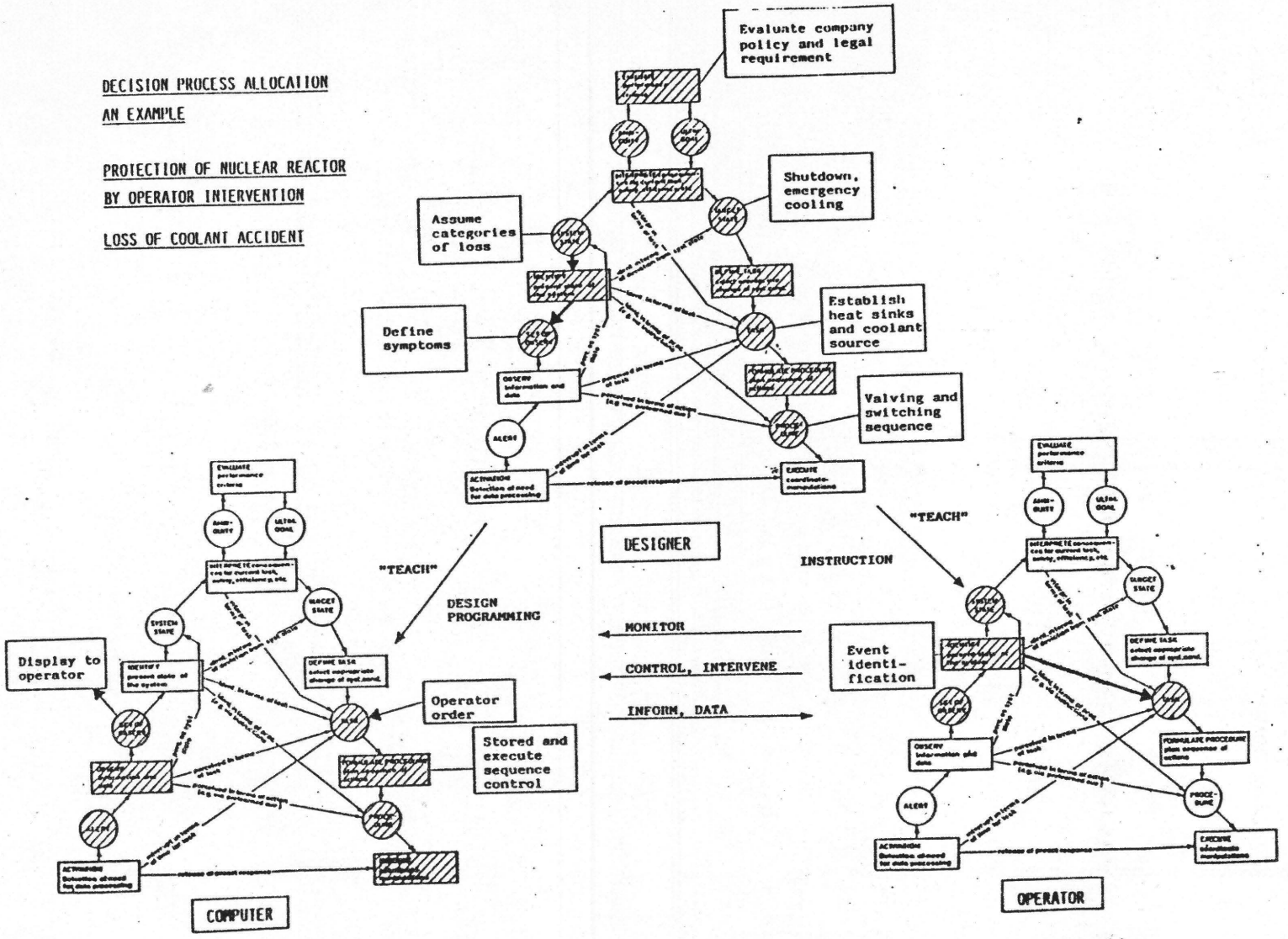


Figure 3B. The figure illustrates the roles of the designer, the operator, and a computer in a control decision which has not been fully automated, i.e. the operator and computer act "in series". The designer has automated a repertoire of protective sequences, but left the diagnosis to an operator. In addition to the decision functions, the designer, operator and computer support each other in different inform/teach/learn functions, Sheridan, [1982].

- the detailed switching sequences required.

He assigns to the computer the tasks of identifying the system state through the specified attribute set and, via a stored decision table which links symptoms to actions, of carrying out the automatic sequence.

After being informed via an early warning that automatic actions may occur, the operators' normal role is to continue with their supervisory task within the re-defined operational envelope.
In principle then, the goals of the computer and the operators are different for this situation. In practise, however, the designer will not be completely confident about the automatic system and the operators will be instructed to monitor that the automatic safety system functions properly AND TO INTERVENE IF IT DOESN'T. This completely alters the basis for the role allocation and, among other things, the information required by the operators regarding automatic shutdown as well as the control resources available to them.

Fig. 3b is an example for situations which cannot be fully automated by the designer. Again the undesirable states and the associated symptom sets are identified. In addition, various redundant and/or diverse resources for coping with the postulated conditions are provided and procedures for the appropriate sequencing and control of these resources are generated - often labelled by the event which is to be dealt with. However, the actual diagnosis of the event is left to the operators - perhaps due to uncertainty or variability in the symptom patterns. Therefore, in this case, the designer/computer and the operators are sharing a task and will, in turn, take care of different of the subroutines in the decision process. The switching back and forth will take place at the standard nodes of the decision ladder and will involve the exchange of intermediate results. Operators are in charge of diagnosis (state identification) with help from the computer-based displays while goal priority and planning for action should come

from a data base supplied by the designers. Subsequent requests for appropriate control actions will be carried out by the computer in response to orders by the operator.

There are again in this case some unclear elements in the planned allocation. If the operators are asked to treat the event-based procedures as irrevocable orders, difficulties will arise. Events are much more varied that the sterotyped categories which the designer can consider; in addition, the resources available at the moment will depend on maintenance schedules, errors during repair, etc.

On the other hand, if the instructions are intended to reflect recommended practise with room for variations in order to match specific occurrences, the operators will need further information about the designers' original analyses - especially the top-down specifications of intentions and goals and any other underlying assumptions and preconditions - in order to have the necessary references for judging the adequacy of the basic procedures as well as the acceptability of their own ideas for modifications to suit the current situation.

INTERACTIVE DECISION MAKING

The two examples have hopefully served to demonstrate the presence of a basic fork in the design philosophy road. Shall/can designers' analyses, heuristics and hypothetical foresight result in computer-stored decisions and guidance for all possible situations. Is this compatible with the policy of having well-trained, knowledgeable and experienced operators serving as final backup.

OR should one instead utilize modern information technology to place the designers' conceptual models and processing resources at the disposal of the operators who then can combine this design base with their up-to-data operational experience and knowledge in an integrated fashion and thus truly serve as the

designers' extended arm in the situations and disturbances which arise.

The argument in this paper is in favor of the latter. In this case, the basic role allocation is that the operator and designer interact closely during the different phases of the decision sequence. Instead of communicating results of analyses based on hypothetical data, the designer brings his conceptual tools into operation "on site" through the computer. The computer has the capacity and accuracy necessary to test consistency of sensor data, to check correspondence with component characteristics and basic physical relationships (e.g. "steam tables"), and to take account of mass and energy balances. In addition, analyses of the data collected during routine operations can be used for defining "normal states" of the various functional levels for the different operating regimes which thereafter can serve as the references for locating disturbances in the functional topography. In this way, the computer will be able to interrelate actual states and target states up through the means-ends hierarchy. In fact, it thereby performs a diagnosis in terms of location of disturbances in the functional topologies without being dependent upon the designer's foresight. The results can be displayed as neutral statements of actual relationships up through the hierarchy without recommendations of actions or priorities.

It will be important to present results from the key nodes in the computer analysis and to select content and form of displays which will allow operators (a) to make crosschecks with their own judgement which could be based on their empirical symptomatic data and (b) to check hypotheses from such sources. The choice of priorities in goals will be left to operators and will depend on their perception of policies and available resources. In such a system, the mental load from having to integrate data will be diminished but at the price of the extra task of retrieving the proper displays or asking the proper questions to the data base. Signals calling attention to the

functional domain where changes have been identified by the computer (functional alarming, Goodstein (1985)) may be an efficient alternative to traditional alarm systems whose main intention is to give advice for action.

Judgement of resources is included in the activities in the downward planning leg of the decision sequence. Again, the communication from the designer should be based on a consistent engineering analysis of intended functions and on tools for on-line analyses of the available physical resources rather than hypothetical analyses of abnormal states. Frequently, several physically possible solutions may be available, and the choice will be dependent upon economic factors or maintenance status. For support of such prioritising, an "expert system" structure may be useful for the operating staff for recording overall plant experience. (Present "expert system" technology probably will be more reliable in a planning task than for ranking choices in process plant diagnosis)

This means that the computer and operators will share the planning function. A basic prerequisite for planning will be information on the operational state and availability of equipment and functions. Thus computer support will depend on adequate access to actual configuration data -e.g., the actual state of the valving and switching. If this is available, the designer will be able to arrange displays of the possible configurations of equipment for various functions with indications of availability considering the maintenance states - the so-called "success-paths" (Corcoran, 1981; Long, 1984). Procedural support for establishing higher level functions on the basis of a choice of a proper success path can be available from a stored library of instructions which will be labelled neutrally by functions, rather than by events. For many such functions, automatic sequence control can be incorporated if necessary with input of actual conditioning information based on maintenance states from operators. This leaves the operator free to express the selected target states at different levels of integration, depending on conditions. Again, information from

the computer is in neutral terms representing engineering analysis of defined technical functions and not hypothetical situations.

CONCLUSION

All of this has serious implications for the design itself as well as the determination of the knowledge required by the operators. In practice, system design based on proven technology is largely an updating of previous designs with little specific attention being paid to a thorough task analysis and/or other special operator needs. However, a formal design and/or one based on new technology such as advanced computer-based techniques, requires a continuous iteration between considerations of purposes, functions and equipment in the means-end hierarchy (Rasmussen and Lind, 1981) - also as the basis for defining the information to be made available to the operators to support their allocated supervisory roles. Indeed, as discussed in Rasmussen (1985a,b), there is the need for a systematic design and evaluation strategy which enables the designer to take into account the capabilities and limitations of the human as well as the computer so as to ensure that a well-balanced and error tolerant design for meeting system requirements can be realized. In our minds, this amounts to a carrying out a multi-phase COGNITIVE TASK ANALYSIS which essentially is a mapping which must bridge the gap between, at the one end, engineering analyses of the target system (power plant, etc.) and, at the other, psychological descriptions of human performance. To be operational, the concepts used in the mapping also have to be consistent and compatible with as well as expressible in terms used in the design of computer and control systems.

The approach to computer support of supervisory decision making advocated in this paper is to consider operators as being capable of taking on the authority and responsibility for the decisions required. Rather than continue a development where

designers attempt to preanalyse all abnormal situations and to store their advice in computers, they should instead try to make available to operators their conceptual tools and use the capacity of computers to perform on-line analyses of the available measured data. Supervisory control should be based as far as possible on consistent engineering analysis. Heuristics and hypothetical foresight should be used to supplement such analyses and to guide the priority of choice and not to replace on-site engineering analyses. However, real life is not black or white; thus coming systems will probably have to combine all the approaches.

When an information system contains a mixture of factual information, heuristics and advice based on other people's foresight, credibility may be a problem. For the design of decision support systems, design criteria are necessary for enhancing users' acceptance of advice and establishing the preconditions for user understanding of explanations and justifications. A basis is needed for establishing and maintaining cooperative attitudes towards a computer intermediary.

REFERENCES

Corcoran, W.R. et al. (1981). The Critical Safety Functions and Plant Operation. Nuclear Technology v. 55 no. 3, pp. 690-712.

Goodstein, L.P. (1985). Functional Alarming and Information Retrieval; Risø-M-2511.

Lind, M. (1982). Multilevel Flow Modelling of Process Plants for Diagnosis and Control. International Meeting on Thermal Nuclear Reactor Safety August 1982, Chicago, Ill.

Long, A. (1984). Computerized Operator Decision Aids. Nuclear Safety. Vol. 25 No. 4. pp.512-524

Rasmussen, J. (1976). Outlines of a Hybrid Model of the Process Plant Operator. in Monitoring Behaviour and Supervisory Control. T. Sheridan and G. Johannsen (eds). Plenum.

Rasmussen, J. and Lind, M. (1981). Coping with Complexity. Risø-M-2293.

Rasmussen, J. (1984a). Strategies for State Identification and Diagnosis. In W.B. Rouse (Ed.): Advances in Man-Machine Systems Research, Vol.1, J.A.I. Press, Greenwich Co.

Rasmussen, J. (1984b). Human Error Data. Facts or Fiction? Proceedings of 4th. Nordic Accident Seminar. VVT. Symposium 55, ESPOO 1985, Finland. Also Risø-M-2499.

Rasmussen, J. (1985a). The Role of Hierarchical Knowledge Representation in Decision Making and System Management. IEEE Transactions on Systems, Man, and Cybernetics. vol. SMC-15, No. 2.

Rasmussen, J (1985b) Risk and Information Processing; Risø-M-2518

Sheridan, T.B. (1982) Supervisory Control: Problems, Theory and Experiment for Application to Human-Computer Interaction in Undersea Remote Systems. Dept. Mech. Eng. M.I.T.Tech Rep., March.

LA PROGRAMMATION

Session A3

PROGRAMMING

Vihmalo Anneli

COGNITIVE STRATEGIES OF NOVICES AND EXPERTS IN COMPUTER PROGRAM COMPREHENSION

1. Introduction

The reading and comprehending of a computer program is a subtask in writing and debugging programs. Maintenance work, which consumes a major proportion of the resources of data processing organizations, consists largely of reading and comprehending the programs written by someone else. Understanding the program is a prerequisite for making the necessary corrections and modifications for programs. Program reading and understanding are also considered one stage of learning to program (see Järvinen et al., 1982), and recently it has been emphasized that different strategies of program reading should be taught (Deimel, 1985).

To date we lack a theory of the cognitive processes and essential components of skills involved in program comprehension. Models of program comprehension, e.g., the syntactic-semantic model of the long-term memory structure (Shneiderman, 1977), may give a general framework to a detailed analysis of cognitive processes, but they are not sufficient alone.

In cognitive psychology comprehension has often been described in terms of schema theories (Neisser, 1976). A schema, then, is a data structure for representing the generic concepts stored in memory. Schemata are developed as a result of a person's earlier experiences, and they act as a kind of orientation basis in a new situation. The central function of schemata is in the interpretation of an object in the process of comprehension. Schema theories describe comprehension in terms of finding appropriate schemata (Rumelhart & Ortony, 1977). Schema theories seem to describe at a general level, why different schemata lead to the use of different comprehension strategies. They do, however, not provide an adequate explanation for the problem of knowledge activation.

Some theories of work psychology and models concerning complex problem solving this relation between knowledge and action has been much deliberated. Questions concerning the activation of knowledge have in work psychology often been associated with the scrutiny of action programs. The development of competency has been described in terms of the construction of higher-level action programs. These so-called metaprograms or metastrategies have been considered relatively general and task-independent action programs, which are manifested e.g., in the ability to choose those strategies or working methods which are adaptive from the point of view of the choosen goal (e.g., Dörner, 1978; Ulich and Frei, 1980). In this study we assumed that metastrategies were manifested in the thinking aloud protocols 1) as statements of self-instructions which concern the goal of reading and comprehending the program, and the conscious choosing of strategies which are adaptive to this goal, 2) as specific self-instructions concerning the order and methods of proceeding in the program text and the instructions to use e.g. external auxiliary means, and 3) as evaluative statements of one's own approaches and strategies provided that they have consequences to the following working strategies.

In the present study, an attempt is made to analyse the qualitative differences in strategies used in program comprehension by subjects with different competencies and work experience. The aim is not to find the one best strategy, but to describe strategy differences in relation to the subjects' competencies. Program understanding and the mental load of subjects are studied as the consequences of the use of different strategies (see Figure 1).

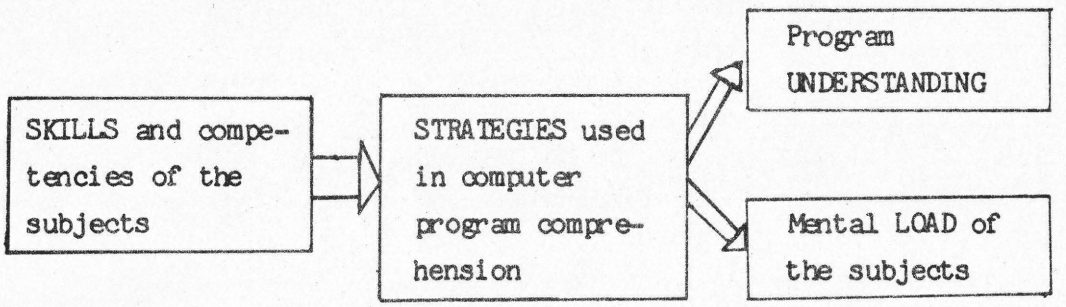


Figure 1. Main constructs of the present study

2. Methods

Program comprehension was studied with three groups of programmers at three different competence levels. The real work situation was simulated as far as possible in a laboratory setting, where two Cobol-programs were presented (1) to the computer science students who had taken the Cobol-programming course (= novices), (n = 12), (2) to the ADP-professionals who had used some programming language other than Cobol, primarily Fortran and FAS for at least two years (= non-Cobol-experts), (n = 12), and (3) to the ADP-professionals who had used Cobol for at least two years (Cobol-experts), (n = 12).

Two Cobol-programs which represented the most common programming problems in administrative data processing, updating and checking input data, were presented to each subject. Both programs were written according to the principles of structured programming.

Each subject was instructed to study what the program was doing and how it functioned. During the investigation he was asked to think aloud. In addition, we tried to get information on the distribution of the subjects' focus of attention during the investigation by video-taping the way the subject skimmed through the program text.

After the working period, which lasted for 22 minutes the subject was asked to answer five recall questions measuring different types of program comprehension at different levels. The subjects were then interviewed concerning the strategies and work methods which they had used during the inspection. The subjects' fatigue and motivation, and changes in these variables during the experimental session, were measured at the beginning of the experiment and again at the end with a 15-item self-rating method developed by Plath and Richter (in Hacker & Richter, 1980).

3. Results

The results concerning the strategies used in the comprehension process indicate that novices and experts who know the syntax of Cobol try to read the program in line with the execution logic of the program. The main indicator for this tendency (i.e., following the

execution logic of the program) was the number of times attention was directed to the data division and to the subparagraphs of the procedure division by reading the main paragraphs of the programs. The more diversions subjects made while reading the program, the more closely they followed the program's execution logic (see Figure 2). The Cobol-experts differed from the non-Cobol-experts in checking-program ($p < .05$) and in updating-program ($p < .10$).

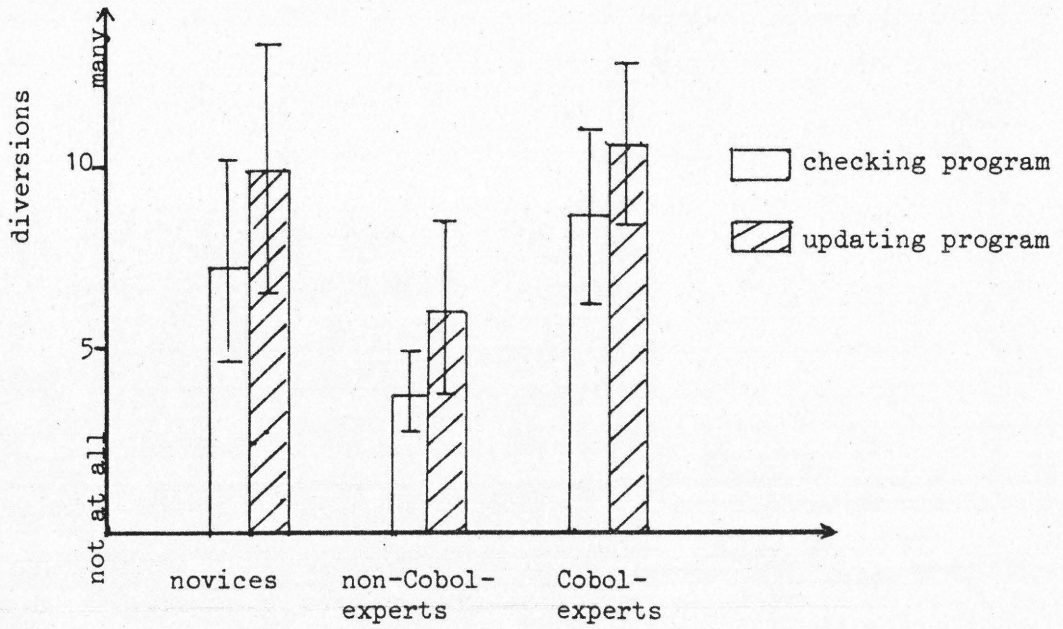


Figure 2. Diversions to data division and subparagraphs of the procedure divisions in different skill groups; means and standard deviations

The variance of diversion in each skill group was largely explained by the subjects' motivation at the beginning of the working period. Motivation was related to the strategy so that, when motivation was low, subjects made no attempt to follow the program's execution logic, but read the program according to the presentation order: page by page and line by line. The effect of low motivation on the reading strategy used was especially noticeable in those extreme cases with the lowest motivation score in their groups. These subjects also had the fewest diversions in their groups.

Novices and experts who knew the syntax of Cobol thus mainly tried to read the program following the program's execution logic. These groups, however, significantly differed in their strategies so

that experts easily perceived large pieces of information in the procedure division of the program while most novices had great difficulties with them. Experts could read continuously 4-5 hierarchically subordinated paragraphs and safely return back to their starting point while most novices could read continuously only two or three paragraphs and returning was always very difficult.

The mastery of the hierarchically subordinated paragraphs and the levels composed of these was evaluated by a three-level scale: good, satisfactory and poor mastery of the paragraph levels. Cobol-experts were evaluated as having without exception a good mastery of the paragraph levels, but only 50 per cent of novices reached this level (see Figure 3).

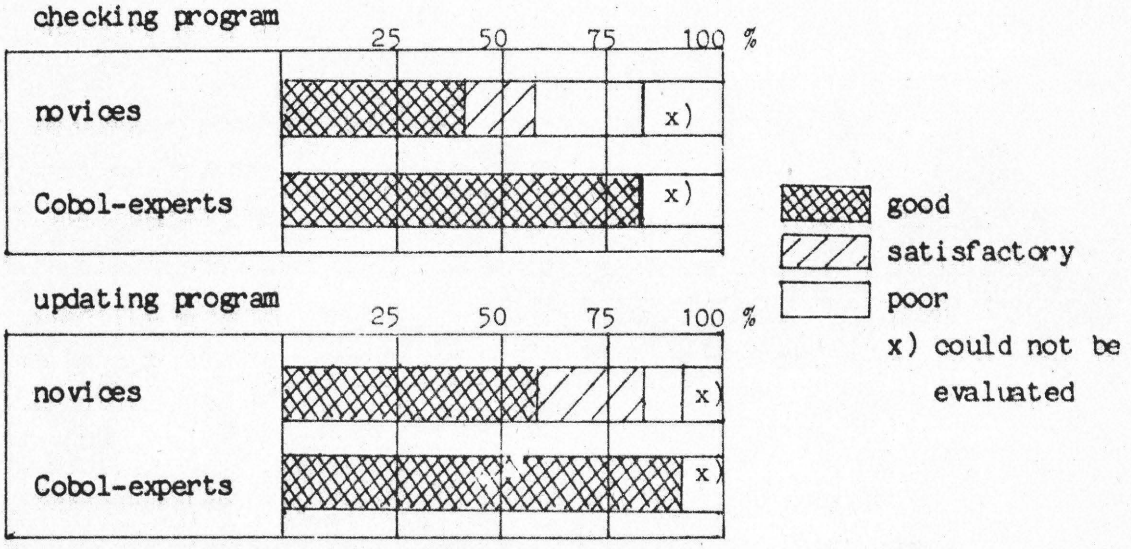


Figure 3. Mastery of hierarchically subordinated paragraphs among novices and among Cobol-experts; percentages

There is no sense in comparing non-Cobol-experts with the two other groups in the mastery of paragraph levels, because non-Cobol-experts did not normally read the program according to the program's execution logic. Non-Cobol-experts, who had programmed updating and checking tasks similar to those used in this study, but who did not know the syntax of Cobol, utilized considerably more than the other groups the natural language information included in the programs. (The variables, files and paragraphs were named according to their function in both programs, and natural language description about the purpose

and the application area of the program was presented in one program.)

Like the other groups, the non-Cobol experts tried to find the essential general points of the programs before going into the details, but the non-Cobol-experts had to use different methods in doing this. For instance, they read through all the pages of the program before going into the details, while the other groups got the same information by reading the name of the program and glancing at the files of the program.

The thinking aloud protocols and the interview statements indicate that the Cobol-experts paid significantly more attention than others to the programming style: 66.7 % of Cobol-experts and only 8.3 %-16.7 % of non-Cobol-experts and novices made comments on the programming style. It was also noted that the Cobol-experts' comments were quite detailed and critical, and included specific alternatives to the writing style of the programs.

There were features in the strategies of the experts which probably made them more economical than the strategies used by the novices. During the comprehension process about 50 per cent of the experts and only 10 per cent of the novices were making notes and sketches, which represents, e.g., the input and output files and their functions in the program. These notes and sketches are called external auxiliary means because of their function as aids to the subject's "internal" information processing and to the short-term memory load.

The use of the external auxiliary means was probably not, however, the only strategy dimension contributing to the fact that the experts were consuming significantly less of their mental resources than the novices during the comprehension process (Figure 4). Fatigue among the novices grew significantly ($p < .05$) more than among the experts. Also the decrease in motivation during the working period was highly significantly ($p < .001$) greater among the novices than among the non-Cobol- and the Cobol-experts, who were surprisingly similar to each other in fatigue.

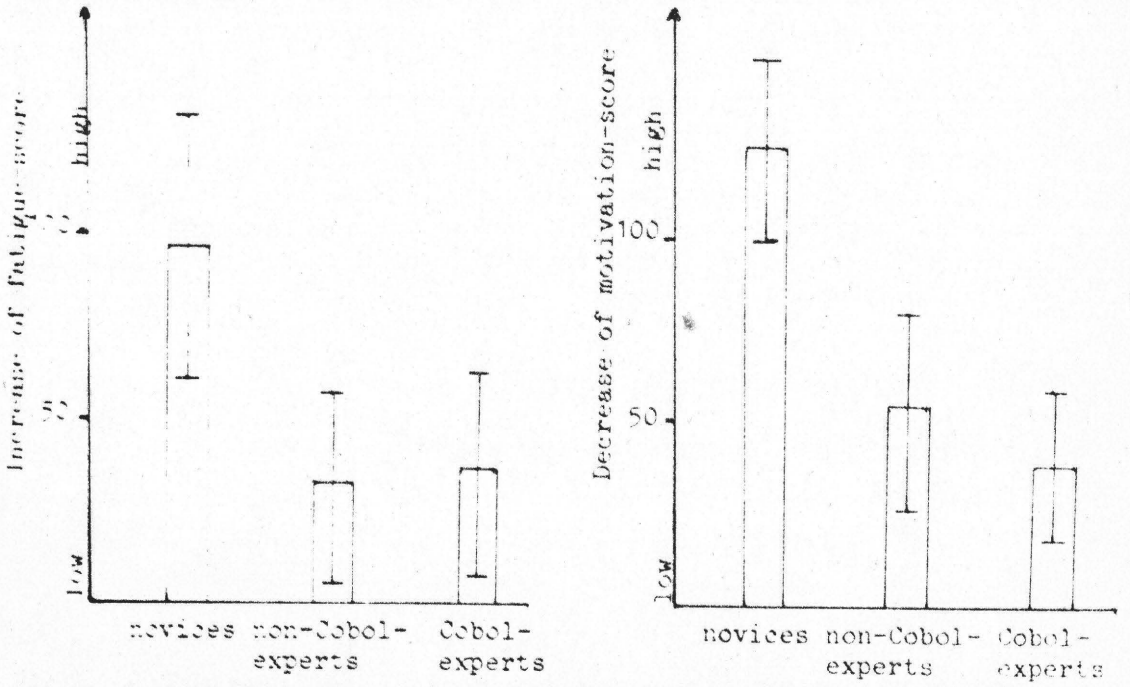


Figure 4. Increase of fatigue and decrease of motivation during the working period in different skill groups; means and standard deviations

Program understanding was mostly explained by those strategy dimensions which clearly differentiated the three skill groups: the tendency to follow the execution logic of the computer and the mastery of the paragraph levels.

An interesting result concerns the role of the so-called meta-strategy for program understanding. Significant differences between the three skill groups could not be found in the use of self-instructive statements. But within each skill group, we discovered extreme groups: about 25 per cent of subjects in each group did not express any statements referring to metastrategies and another 25 per cent expressed comprehensive statements about the goals and strategies of working. When comparing these extreme groups in program understanding (see Figure 5), we can see that in both experts' groups, program understanding was significantly better in those groups which expressed comprehensive statements referring to the use of the metastrategy. In the novice group, this kind of relation could not be found.

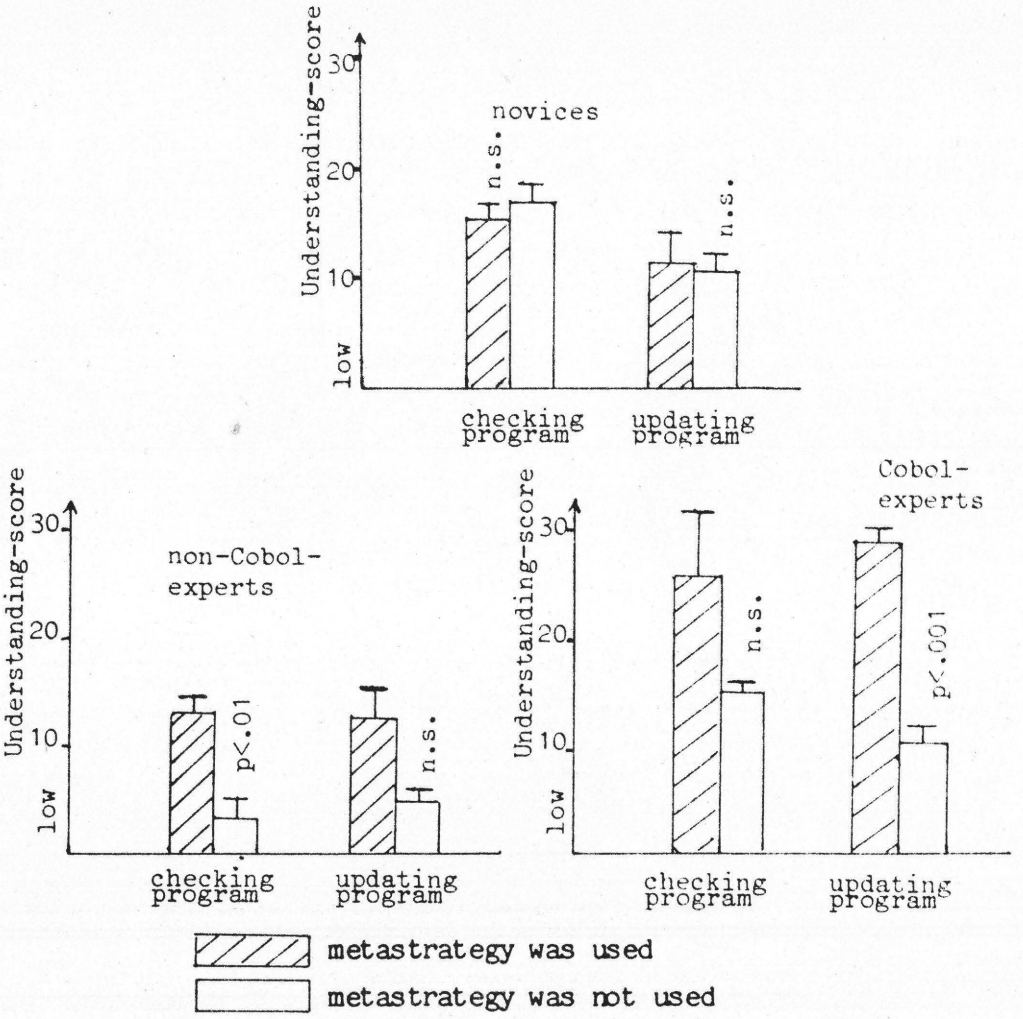


Figure 5. Program understanding and use of metastrategy in different skill groups: means and standard deviations of the extreme groups.

4. Discussion

The main results of this study addressed the qualitative differences in program comprehension strategies of three skill groups. Program understanding and the subjects' mental load were considered to be the consequences of the use of different strategies.

The Cobol-experts who had a good deal of programming experience with the programming language used in this study (viz. Cobol) and with identical programming tasks (updating and checking input data), mainly tried to read the programs following the execution logic of the program. In addition to this, the Cobol-experts easily perceived large

pieces of information when reading the functionally subordinated paragraphs. These two tendencies, which were typical of the Cobol-experts, created the impression that they moved along with the programs. Their mental representations of the programs were dynamic so that they saw the programs running in the computer.

Similar results have been reported by Jeffries (1982) and Collins and Gentner (1982), who talk about runnable mental models of experts. Reading programs according to their execution logic is referred to as deep processing of information and is distinguished from line-by-line reading, which is called surface processing.

The strategies typical of the Cobol-experts had also another characteristic which in the literature concerning learning strategies (von Wright et al., 1979) is described as deep processing of information. Deep processing is manifested, e.g., in the tendency to transform the offered information so that it is more adaptable to the subject's own knowledge structures.

The use of the external auxiliary means in this study could partly be understood as transforming the verbal program code into a pictorial mode. We discovered that the experts used more external auxiliary means than the novices, and thus they also transformed more than the novices. The same tendency could be found in the thinking aloud protocols. The extent of direct repetition of the program code was about forty per cent among the novices while the experts had only thirty per cent direct repetition of the code in their thinking aloud protocols.

An active transforming approach to the offered material was manifested by the majority of the Cobol-experts, who expressed critical comments about the programming style and gave detailed alternatives for better solutions. Pea and Kurland (1984) have stated that this kind of critical attitude is possible only at the highest level of programming skill development.

The strong increase of fatigue and the prominent decrease of motivation in the novice group is largely understandable if we take into account the strategy typical of this group. We saw that the novices tried to read the program, like the Cobol-experts, following their execution logic, but many of them had great difficulties in the mastery of the programs' paragraph levels. The main reason for this

difficulty was the small size of perceived information. The novices could read and translate Cobol-code, but they probably did not have the general concepts about program structure, and so they could not transform the programming code into wider units or chunks. This gave rise to a considerable load to their short-term memory.

It was also noted that chunks received by the novices were not wide enough to form a runnable mental representation of the program. This lack of success in getting a runnable mental model appeared to be connected with a prominent decrease of motivation.

While the novices tried to use strategies which were too difficult for their competence and were thus strongly loaded, the non-Cobol-experts set themselves realistic goals and used strategies which were appropriate to these goals. It was found that the non-Cobol-experts did not try to read the programs according to their execution logic and that they, e.,g., utilized more than the other groups the natural language information included in the programs. The non-Cobol-experts who had a long experience of programming of similar tasks, could choose their strategies from a larger repertoire of alternatives than did the novices.

The use of the metastrategy, which served as a conscious choice of strategies suitable for the situation, was successful among the experts, but not among the novices, who obviously did not have enough alternative strategies. It would be very interesting to make further investigations of the overall function of the metastrategy, and to see if a critical attitude to one's own strategies could be developed in tasks with relatively complex cognitive demands.

Reference

Collins, A. & Gentner, D. (1982) Constructing runnable mental models. Proceedings of the Foruth Annual Conference of the Cognitive Science Society, Ann Arbor, JI, August, 1982.

Deimel, L. E. Jr. (1985) The uses of program reading. SIGCSE Bulletin, Vol. 17, No. 2, pp. 5-14.

Dörner, D. (1978) Self-reflection and problem solving. In Klix, F. (Ed.), Human and artificial intelligence, pp. 101-107. VEB Dt.Vlg.d. Wisenschaften, Berlin.

Hacker, W. & Richter, P. (Ed.) (1980) Spezielle Arbeits- und Ingenieurpsychologie 2. VEB Dt. Vlg. d. Wissenschaften, Berlin. (in German).

Jeffries, R. (1982) A comparison of the debugging behavior of expert and novice programmers. Papers presented at the Annual Meeting of the American Educational Research Association, New York City, March, 1982.

Järvinen, P., Kirjonen, J., Tyllilä, P. & Vihmalo, A. (1982) Analysis and design of jobs in a man-computer system. Department of Mathematical Sciences, University of Tampere, Report A74.

Neisser, U. (1976) Cognition and reality - Principles and implications of cognitive psychology. W.H. Freeman and Company, San Francisco.

Pea, R. D. & Kurland, D. M. (1984) On the cognitive effects of learning computer programming. New Ideas Psychol., Vol. 2, No. 2, pp. 137-168.

Rumelhart, D. & Ortony, A. (1977) The representation of knowledge in memory. In Anderson, R. C., Spiro, R. J. & Montague, W. E. (Ed.) Schooling and the aquisition of knowledge. John Wiley & Sons, New Jersey.

Schneiderman, B. (1977) Measuring computer program quality and comprehension. International Journal of Man-Machine Studies 9, pp. 465-478.

Ulich, E. & Frei, F. (1980) Persönlichkeitsförderliche Arbeitsgestaltung bei Qualifizierungsprobleme. In Volpert, W. (Ed.) Beiträge zur Psychologischen Handlungstheorie. H. Huber, Bern. (in German).

von Wright, J., Vauras, M. and Reijonen, P. (1979) Learning strategies of school-age children, I. Psychological Reports, University of Turku, No. 33. (in Finnish).

THIRD EUROPEAN CONFERENCE ON COGNITIVE ERGONOMICS
Psychological Foundations for Human-Computer Interaction
Paris, September 15-20, 1986

PROGRAM UNDERSTANDING AND KNOWLEDGE ORGANIZATION:

THE INFLUENCE OF ACQUIRED SCHEMATA

Françoise DETIENNE

Institut National de Recherche en Informatique et en Automatique, Domaine de Voluceau,
Rocquencourt B.P. 105,
78153 Le Chesnay Cedex, France

This study concerns program understanding by experts in programming. In the theoretical framework of schema theory, three experiments have been conducted. The experimental paradigm allows to study the process of schemata activation during reading and the influence of the activated schemata on the form and content of the program representation stored in memory. The results of these experiments indicate that the experts possess (a)schemata representing functions and data structure that are dependent on the task domain (b)algorithmic schemata (c)elementary schemata of programming that are independent of the task domain.

The topic of this study concerns the **understanding of program by experts in programming.**

Understanding a program, or more generally a text, necessitates constructing a representation of the program or the text. This elaboration involves different processes: decoding, interpretation of the linguistically coded information and activation of the knowledge stored in memory. The processes involved in the elaboration of a text representations are depending on the organization of knowledge in memory and of the mode of access to this knowledge. So, the topic of program understanding raises the question of **the processes involved in the construction of a circonstantial representation of a particular program with regard to the organization of the knowledge stored in memory.**

An attempt to construct a model that accounts for the organization of the knowledge in a particular domain, has to take into account the **expertise** of the subject in the domain. Many studies (ADELSON 1981, CHI 1981, McKEITHEN 1981, WEISER 1983) gave prominence to the different organization of experts' and novices' knowledge in a domain. The expert in a domain not only has more knowledge than the novice, but, more importantly, he/she organizes differently his/her knowledge. In this study, we are only interested in the analysis of the **knowledge organization of experts in programming.**

THEORETICAL FRAMEWORK: THE SCHEMA THEORY

Our theoretical framework is based on the schema theory. This theory concerns the knowledge organization in memory and the role of that knowledge in text understanding (ANDERSON 1981, RUMELHART 1978, 1981, SCHANK 1977). A **schema** is a data structure for representing, in memory, generic concepts of a domain. It includes variables and constraints for the instantiation of the variables. A schema is instantiated when a particular configuration of values is bound to the variables, at a particular moment. Those structures of knowledge account for the inferences realized in the processes of text understanding. While reading a text, a reader evaluates constantly hypotheses on the possible interpretation of that text. "Readers are said to

have understood the text when they are able to find a configuration of hypotheses (schemata) which offer a coherent account for the various aspects of the text" (RUMELHART 1978). The processes involved in the elaboration of a representation are interactive, i.e, conceptually driven and data driven. By a bottom-up process, information extracted from the code activate schemata: this activation creates expectations on the instantiation of others slots of the activated schemata. The main process of understanding is the elaboration and evaluation of hypotheses (schemata). Some indices extracted of the code suggest possible interpretations by the activation of schemata, then, those interpretations are evaluated by information decoded of the text.

EXPERIMENTAL PARADIGM

With this theoretical framework as a reference, three experiments have been conducted. Two of these experiments have been presented with more details elsewhere (DETIENNE 1984, 1985). The experimental paradigm implied studying two aspects of experts' understanding:

- on one hand, the process of representation construction was investigated using analyses of the subjects' activity during information gathering in a debugging task;
- on the other hand, the result of this activity, that is the representation stored in working memory, was investigated using a recall task.

This paradigm allows to study the process of schemata activation during reading and the influence of the activated schemata on the form and content of the program representation.

During the reading phase, the subjects had to discover an unknown program and to evaluate its correctness. The programs were written in Pascal. In a first experimental setting, we have recorded, during the reading phase, the verbalization of subjects, their written notes, the sequences in reading the different instructions. The study of the exploratory strategy of subjects shed light on the research of indices that activate their knowledge. To study the process of elaboration and evaluation of hypotheses, another experimental setting was designed in which subjects had to verbalize: the program was presented one instruction at a time, and the subjects had to tell the information provided by the new instruction and the hypotheses that they could elaborate concerning the other instructions and the functions performed by the program. This experimental setting, which is not very natural, allows nevertheless to find out the nature of the indices that activate, in general, the knowledge of the subjects, experts in programming. This information will allow to define the structure of the schematic knowledge possessed by experts. The verbal and behavioral protocols have been coded in the form of production rules: IF A THEN B. "A" is a piece of information extracted from the code or the memory, "B" is a piece of information allowing expectations on the progran. . These rules constitute a description of the activity of subjects. These rules have been categorised in general rules when different instantiations of the same rule have been observed. The regularities in the knoweldge that are revealed by these rules have been translated in the form of a schematic organization that is supposed to be possessed by experts.

The recall task allows to analyze how the experts' knowledge affects the form and the content of the resulting representation. On this aspect, the schema theory makes different predictions: (a)distortions of the form of the program, (b)thematic intrusions that is distortions of the content of the program.

The theory of schemata predicts distortions of the form of the recalled information, i.e the effect of the process of abstraction. A subject does not keep in memory the surface form of the information presented but this information is abstracted in a more general representation. The information selected by the encoding process is reduced by the abstraction process. This process encodes the meaning of a particular piece of information but not the surface form, therefore, the lexical form of a concept or the syntactic form of a sentence may not be kept in memory.

The theory predicts also distortions of the content of the information, called "thematic intrusions": they are observed when information typical of a schema is recalled while not included in the program text.

The results of these experiments give support to the hypothesis that the experts knowledge is organized in schemata of different levels of abstraction: according to the goals of subjects, we have constituted three categories of rules that are descriptives of the activity of subjects: (1)rules of identification of fonctions, data structure, interaction between schemata (2)rules of identification of algorithmic categories and particular algorithm (combination of elementary schemata) (3)rules of identification of elementary schemata. These rules allow to describe the organization of the knowledge that is involved in program understanding. **The experts possess schemata of different levels of abstraction: (1)Schemata dependent of the task domain (2)algorithmic schemata (3)elementary schemata of programming. In these schemata, prototypical values could be associated to variables.** These different issues are going to be illustrated in the following sections.

EXPERTS POSSESS SCHEMATA DEPENDENT ON THE TASK DOMAIN:

Part of the knowledge possessed by experts is organized in schemata that are dependent on the task domain. During the reading phase and representation construction, an exploratory strategy was observed: in the first phase, the experts read rapidly the program by taking information on the declarative section, on the names of subprograms and on the body of the main program. It is possible to assume that the experts search for indices that inform them about the data structure and about the different processing phases. The analysis of the simultaneous verbalizations obtained in this reading phase shows that some indices of the code allow the experts to elaborate hypotheses about the task domain, the data structures and the fonctions performed in the program. Here are some examples described by rules of the form "IF...THEN...". The indices that the subjects extract from the codes are on the left hand side of the rule. The hypotheses elaborated by the subjects are on the right hand side of the rule.

R
IF title of the program
THEN task domain
EX:
IF the title of the program is "stock management"
THEN the task domain is stock management

R
IF task domain
THEN data structure
EX:
IF the task domain=stock management
IF the type of variable1=record with two fields
IF the type of variable2=chain of characters
THEN the first field of the record=name of file
the second field of the record=descriptor of file
type of the first field= chain of characters
type of the second field= integer or real
first field=variable2

R
IF task domain
THEN performed functions
EX:
IF the task domain is stock management
THEN the performed functions are allocation(creation or insertion), destruction, search

The knowledge that allows to elaborate these expectation systems can be described under the form of schemata dependent on the task domain. The general structure of this schema includes slots on the data structure and on the possible functions in the particular task domain. The slots are the following:

task domain
data structure
functions

For the particular domain of stock management, the following values could be assigned to the slots:
task domain: stock management
data structure: record(name of file, descriptor of file)...
functions: allocation (creation or insertion), destruction, search

In one of our experimental settings, after the first phase of the exploratory strategy, the subjects had to perform a recall task. A thematic insertion was observed. The function of creation was not isolated in a subprogram and the subjects did not read any information concerning this function in the program. Nevertheless, a subject has reported this function as if it was a subprogram. The schematic knowledge dependent on the task domain has participated to the elaboration of the representation and to the process of recovering of stored information at recall.

EXPERTS POSSESS ALGORITHMIC SCHEMATA

The analysis of verbal and behavioral protocols of subjects obtained during the reading phase, allowed to elaborate rules for the identification of algorithmic categories and of particular algorithms(combination of elementary schemata). Here are some examples.

R
IF type of data structure (SD)
IF function
THEN algorithmic category for realizing function
EX:
IF type of SD=array of records without chaining structure(/with chaining)
IF functions=allocation, search, destruction
THEN algorithmic category=scanning(/chaining)
EX:
IF type of SD=array of characters
IF function=coding
THEN algorithmic category=haschcoding
EX:
IF type of SD=array of characters
IF function=reading
THEN algorithmic category=linear reading

R
IF ELEMENTARY SCHEMATA OF VARIABLE
IF function
THEN algorithmic category for realizing function
EX:
IF ELEMENTARY SCHEMATA OF COUNTER
IF ELEMENTARY SCHEMATA of READING VARIABLE
IF function=reading characters
THEN algorithmic category: linear reading
EX:
IF ELEMENTARY SCHEMATA of COUNTER
IF function=search
THEN algorithmic category=linear search

R
IF algorithmic category
IF ELEMENTARY SCHEMATA of VARIABLE
IF ELEMENTARY SCHEMA of Control structure
THEN combination of ELEMENTARY SCHEMATAS

These examples allow to describe the slots that could compose algorithmic schemata:

function
type of SD
elementary schemata of variable
elementary schemata of control structure

In a schema for linear search, the slots could take the following values:
function: search
type of SD: array
elementary schemata of variable: counter
elementary schemata of control structure: iteration, test

EXPERTS POSSESS ELEMENTARY SCHEMATA OF PROGRAMMING:

Some of the knowledge possessed by the experts is organized in schemata that are independent of the task domain. Several experimental observations support this idea.

During the reading activity and representation construction, the experts are able to diagnose some bugs without having understood what the program and the bugged subprogram perform: for example for the bugs of the initialization of variables used as counters. They have constructed a representation that is specific to a type of programming process and independent of the knowledge on the task performed by the program.

The analysis of the verbalizations obtained during reading illustrates the nature of the indices of the codes allowing the expert to make hypotheses on the semantic of the variables. Some indices allow the activation of expectation systems: the name of a variable and its type allow the subjects to expect particular forms of initialization, updating and a context. This happens whatever the knowledge the subjects have on the task domain is. Here are some examples:

R
IF type of variable
IF name of variable
THEN initialization
THEN updating
THEN context
EX: (in the case of a variable ,V, used as a counter)
IF there is a local variable named I, J or K,
of which the type is integer
THEN the variable is initialized at the value 0...n
The variable is updated by an incrementation: V:=V+1
The variable appears in the context of an iteration
EX: (in the case of a variable used as a flag)
IF there is a global variable named Termine,
of which the type is boolean
THEN the variable is initialized at the value false
the variable is updated at the value true
the variable appears in the context of an iteration

These two examples allow to describe the slots that could compose schemata of variables. The slots are of the following types:

type of variable
name of variable
initialization
update
context

The domain of the possible values associated to each of the slots is different for a counter and for a flag.

In a schema for a counter, the slots could take the following values:
type of variable: integer and local
name of variable: I, J, K...
initialization: V:=O to n
update: V:=V+1
context: iteration

In a schema for a flag, the slots could take the following values:
type of variable: boolean and global...
name of variable: termine...
initialization: V:=false
update: V:=true
context: iteration

These constraints on the instantiation of the slots allow subjects to elaborate expectation systems during reading of a program: knowing the value associated with the slots "type of variable" and "name of variable", the experts are able to expect the values associated with the other slots of the schema.

Our hypothesis is supported by the form of the distortions observed in the recall protocols. Some distortions are made on the name of a variable used as a counter: for example, the subjects recall I instead of J. Then, the variables are memorized in a category, a schema for a counter, and the lexical form is not kept in memory. In the recall process, the subjects use another possible value of the slot "name of variable".

Other observations suggest the existence of prototypical values in the slots of a schema. Each slot is associated with a set of possible values, as we have seen. Those values have not the same status, some values being more representative of a slot for a schema than the others. When there is a prototypical value in a category, this value comes first in mind when the category is activated.

In a schema for a flag, the slot "context" is an iteration that can take the values "repeat" or "while". In the program used, this variable, V, appears in a "repeat" iteration. During the reading phase, the value of the iteration expected by most of the subjects was "while not V do". In the recall protocols, a distortion was observed: a subject has recalled the instruction as "while" instead of "repeat". This subject has reported the prototypical value instead of the adequate value. These observations give support to the hypothesis of prototypicality of some values associated to the slots of the schemata.

RELATIONSHIP BETWEEN SCHEMATA:

The schemata represent knowledge at different levels of abstraction.

The elementary schemata cannot be broken up into sub-schemata and can be match against the code.

The schemata dependant on the task domain represent abstract knowledge.

The algorithmic schemata allow to construct a mapping between elementary schemata and schema dependant on the task domain.

Our observations support the hypothesis that some functions represented in schemata dependent on the task domain are related in memory to categories of algorithms. For example, the function of search could be related to many algorithms that could be organized, in memory, in categories and subcategories as algorithms of linear search or of direct access search. At a certain level of representation corresponding to a subcategory (for example linear search in a particular data structure), knowledge could be organized in a schema where slots are, on one hand, elementary schemata, for example schemata of count variables, and, on the other hand,

constraints for the combination of the elementary schemata, with some combinations being more prototypical than others. The elementary schemata could be defined as "primitives".

RELATIONSHIP WITH SOME MODELS OF PROGRAM UNDERSTANDING

These results provide an experimental validation to two theoretical models of program understanding: the models developed by Brooks and Soloway & Ehrlich.

We have observed that, very early, during the reading of a program, the experts were searching for indices that activate schematic knowledge dependent on the task domain. That knowledge allows the subjects to elaborate hypothesis on the data structure and on the possible functions of the program. This observation is consistent with Brooks'model (BROOKS 1983). According to that model, the process of understanding is based on the successive refinement of hypotheses about the knowledge domains and their relationship to the executive program. This theory asserts that the hypothesis generation and verification process begins with a primary hypothesis that is generated as soon as the programmer obtains any information about the task that the program performs: it will specify at least the global structure of the program in terms of inputs, outputs, major data structures and processing sequences.

Our observations suggest the existence of elementary schemata of programming. This notion is very close to Soloway & Ehrlich's concept called "Programming Plan" (JOHNSON 1985, SOLOWAY 1982, 1983, 1984). Programming plans are program fragments that represent stereotypic action sequences in programming. There are two types of schemata: variable plans and control plans. Variable plans are plans which generate a result that is usually stored in a variable. This is very close to the elementary schemata that we have described for two types of variables, counter and flag.

PROSPECTS: schema activation and evaluation process

The present study has analysed some of the knowledge possessed by experts. This knowledge has been described in the form of schemata. Other studies will have to be conducted to complete this analysis: the hypothesis of schemata dependent on task domain should be evaluated in other task domains than stock management, the hierarchical relationship between schemata should be described with more accuracy, and the hypothesis concerning the schematic organization of experts'knowedge should be evaluated with other languages than Pascal.

Whereas the process of schema activation and schema instantiation has been well described by different studies in the past, there is a fundamental process that has not been studied before. That is the process of evaluation by which the subject evaluate the coherence of the constructed representation . We think that the evaluation process could be studyied through the analysis of the failures to understand: when a subject does not understand, it means that he cannot integrate new information to the representation constructed before. The subject expresses a judgement of incoherence between new information and representation constructed before. We have already collected some observation on the failures to understand. The analysis makes clear that the subjects elaborate and evaluate hypotheses of external coherence and of internal coherence. The hypotheses on the external coherence between schemata are elaborated at the level of functional representation and depend on rules of interaction and causal relation between performed functions. The hypotheses of internal coherence depend on constraints internal to schemata. In order to judge that some portion of program is correct, the subject has to elaborate and validate hypotheses of coherence of these two types. Others studies will have to be conducted to test this hypothesis. These studies would give us information about debugging activity in which the evaluation process is most important.

Adelson, B., 1981, Problem solving and the development of abstract categories in programming languages. Memory and Cognition, 9(4), p.422-433.

Anderson, R. P., 1981, Psychological status of the script concept. American Psychologist, 36(7), p.715-729.

Brooks, R., 1983, Toward a theory of the comprehension of computer programs. International Journal of Man-Machine Studies, 18(6), p.543-554.

Chi, M.T.H., Feltovich, P.J. & Glaser, R., 1981, Categorization and representation of physics problems by experts and novices. Cognitive Science, 5, p.121-152.

Detienne, F., 1984, Analyse exploratoire de l'activité de compréhension de programmes informatiques. Proceeding AFCET, Séminaire "Approches quantitatives en Génie Logiciel". Sophia Antipolis, France, 7-8 Juin.

Detienne, F., 1985, Programming expertise and program understanding. Ninth Congress of the International Ergonomics Association, Bournemouth, England, 2-6 september.

Johnson, W. L.& Soloway, E., 1985, PROUST: Knowledge-based program understanding. IEEE Transactions on Software Engineering, 11(3), p.267-275.

McKeithen, K.B., Reitman, S.R., Rueter, H.H. & Hirtle, S.C, 1981, Knowledge organization and skill differences in computer programs. Cognitive Psychology, 13, P.307-325.

Rumelhart, D. E., 1978, Schemata: the building blocks of cognition. Center for Human information Processing, University of California, San Diego, CHIP 79.

Rumelhart, D. E., 1981, Understanding understanding. Center for Human information Processing, University of California, San Diego, CHIP 100.

Schank, R. & Abelson, R., 1977, Scripts, plans, goals and understanding. LEA.

Soloway, E., Ehrlich, K. & Bonar, J., 1982, Tapping into tacit Programming knowledge. Human Factors in Computer Systems, 15-17, p.52-57.

Soloway, E., Ehrlich, K., 1983, Reading programs is like reading story (well, almost). Proceedings of the Fifth Annual Conference of the cognitive Science Society, Rochester, NY, 18-20 May.

Soloway, E., Ehrlich, K., 1984, Empiricals studies of programming knowledge. IEEE Transactions on Software engineering, 5, p.595-609.

Weiser, M. & Shertz, J., 1983, Programming Problem representation in novice and expert programmers. International Journal of Man-Machine Studies, 19, P.391-398.

UNDERSTANDING THE COGNITIVE DIFFICULTIES OF NOVICE PROGRAMMERS: A DIDACTICAL APPROACH

Renan SAMURCAY
Laboratoire de Psychologie du Travail EPHE/CNRS
41, Rue Gay Lussac 75005 Paris FRANCE

1.Introduction

In this presentation we will focus our attention on the particular domain of HCI: programming learning. The cognitive analysis of programming learning process is important from two points of view. First it has significant practical values: programmer productivity is an important economic issue
and programming education is highly integrated in the school curriculum. Second, theoretically programming is an interesting domain of activities for cognitive psychology: it is not only a specific problem-solving domain but also a conceptual field involving particular concepts and procedures.

The specifities of programming as a problem-solving situation can be summarized in few points: the goal of the activity is not only to produce a solution as in classical problem solving situations, but also to explicit this procedure producing the solution; the activity is highly mediatized by a technological tool for which the subject has to construct functional representations; the elaboration and the expression of the procedure in a specific "action code system" necessitate the acquisition and the use of specific programming concepts.

These characteristics have important effects on the learning process. One cannot consider programming learning only in terms of the acquisition of syntax or method. For instance, the research works on learning programming methodology (Hoc, 1981; 1983) suggest that the lack of a "Representation and Processing System" (RPS), closely related to the computer operation, constitutes an obstacle for novice programmers in learning and using these methods who have very specific procedural constraints. This kind of results suggests that a way to conceive the initial stage of programming learning is to permit student to construct a RPS organized by a hierarchy of plans. However the acquisition of plans itself is in relation with the acquisition of programming concepts such as variable, iterative and conditional structures, etc.

The aim of this study is to focus on the conceptual aspects of the programming activity by analyzing an early learning process of some programming concepts by beginner

students. The didactical approach is developed above as a methodological issue to study this kind of long learning process in relation with the problem situations used in the teaching.

2. Learning programming

As revealed by many recent research works, programming even at a simple level is a difficult activity to learn. How can we characterize this difficulty? One way to answer the question is to analyze novices' bugs, as indicator of their representations and difficulties of learning. Given this context new questions are rising about the identification and the interpretation of the bugs. A same bug may be interpreted differently with respect to the theoretical and methodological framework in which the programming activity is analyzed.

Two opposite frameworks can be distinguished. The first one concerns theories without references to psychological mechanisms underlying programming activity: for instance, the early descriptive theories based on the syntactic categories of bugs. The second concerns theories without references to the specifities of programming knowledge, based on the general psychological mechanisms: for instance, the interpretation of the bugs as resulting from the working memory limitation (Anderson & Jeffries, 1985). Actually more rich attempts are done in the direction to coordinate these two poles and to build up some descriptive theories on the specific knowledge that the programmers have (Bonar & Soloway, 1985; Soloway & al., 1984). According to this approach the bugs result from the use of a single plan to achieve multiple goals. Here a plan is defined as a method to achieve a particular goal. For instance when the goal is to count the number of occurency of an event the corresponding plan may be a counter variable plan (i.e., initialization, updating and eventually test of a variable defined as a counter variable).

The richness of this approach may be summarized in two points: first it takes into account both of the representations related to the problem domain and programming domain. Second, to the same bugs correspond multiple explanations with respect to the specific context in which it occurs.

Knowing about the novices bugs is very important to understand how their actual programming knowledge is functionning in problem solving. However it is difficult to infer from this kind of data (corresponding to a certain state of novices' knowledge) information about the evolution of representations in learning process and to derive informations about instruction. Are the bugs the same at different moments of the learning? What are the relations between the bugs and the teaching situations encountered by students? Can we distinguish different levels of complexity between different types of task? Analysis of beginners' behaviour in learning process may

contribute to characterize the cognitive difficulties specific to programming domain and to propose some instructional help to teachers.

3. Methodological issues

The question we want to study necessitates some methodological options. First we need to collect data on students during the learning process. Second we need to control as much as possible the content of the teaching situations. One way to realize these options is to construct experimental teaching situations in school context: each session is prepared in advance and the teacher knows before each session what to teach and how to teach.

The teaching situations we have constructed are two types: (i) situations in which students have to solve problems involving knowledge acquired in previous sessions; (ii) situations in which students haven't yet all of the concepts and procedures needed for solution of the problems, then the new concepts and procedures are introduced as a solution of the problem.

Three groups of 18 college students (16-17 year old) were involved in the study. They worked 1 hour a week during 25 hours with PASCAL programming language, on 6 SILL'Z micro computers. They are asked to write programs about simple arithmetic sum problems. Their solutions use few repetitive actions and illustrate many of the basic programming concepts such as loop and variable. The choice of these concepts are based on the idea that iteration is one of the most important function of computers and the mastery of loop strategies is necessary to solve significant programming problems.

We have collected three types of experimental data: (i) clinical observations of students' work during the problem solving session; (ii) the notes taked by students; (iii) planified questionnaries at different moments of training. In the following session we analyze the concepts involved in the study and explain psychological hypothesis on which based the experimental teaching design.

4. Analysis of the concepts involved in the study and hypothesis

The loop-plan is an answer to problems for which the solution necessitates to repeat identical actions a necessary number of times. The construction of a loop-plan involves the identification of what will be repeated and the condition for which the repetition should to be end.

According to the semantic content of the actions and the nature of the condition, the construction of the plan may be more or less difficult for the novice programmers. For instance, it can be hypothesized that the situations in which the number of repetition is

known in advance, are conceptually more easy for students than those in which this number is a variable of the problem.

Pascal language presents three forms of loop-plan: for-, repeat-, while- loop. Formally these forms are equivalent: each of them may be rewritten in terms of the others. The question is to know if they are also equivalent for the novice programmers in terms of the cognitive difficulty and acquisition. For instance, the comparison of the repeat-loop with the while-loop shows that the plans are not equivalent : the first one necessitates a plan of actions of the form

process variable / test variable

which is close to the novice's familiar procedures than the form

test variable / process variable

involved in while-loop. It is difficult for the subjects to represent and to express a condition about an object on which they didn't operate yet. Thus, it can be hypothesized that the conceptualization of the while-loop will be more difficult than the repeat-loop.

When solving the loop problems the subject has to consider also the variables and the operations on them. From the programming point of view a variable is defined as an address but this formulation is not sufficient for analyzing this concept in its functional meaning for the subject (Samurçay, 1985b). In fact when the values of a variable change, its naming and its relationships with the other elements of the program are invariant. In the programming activity with procedural languages such as Pascal, Basic, Fortran etc..., this property of invariance appears particularly in the looping problems, in which variables are involved with a particular status such as

sum ←--- sum + number

We will call this type of variables "accumulator" for distinguishing them to other ones. The understanding and the construction of this kind of symbolic representations necessitates to consider a variable in its temporal definition: "the value of variable sum at n+1.th step is equal to its value at n.th step plus the value of variable number". The student has to designate by the same name both of the preceeding value and the present value which is a function of the former and has to process the assignement sign as an asymetric relation. The mathematical model of variable and the equality relation constitutes for a novice an initial but unsufficient model for operating on the programming variables when these are involved in the loop construction. The mathematical description of a variable is a static one and it designates either a generic name of a set

$$x \in [a,b]$$

or an unknown in an equation

$$3x = x + 10$$

The programming variable is new concept for the student and we hypothesize that operate on the variables in loop problems, in particular the operation of update, will be a difficult task for students.

The main hypothesis which guides the teaching situation design are two: (i) at the beginning of their learning process the students will use their "familiar procedure" of execution developed in problem domain; (ii) the construction of the procedures compatible with the functionning of the system necessitates to encounter problem situations in which these familiar procedures don't work.

5. Problem situation design and observations

The hypothesis developed above conduct us to design the following sessions. A first session is conducted with the aim of identifying the "spontaneous" procedures of students when solving a loop problem (we have asked to write a "natural language" procedure to calculate x*y ; the procedure will be executed by an other student on a theoretical device which has only two operations: + a number and +1). Analysis of collected data shows that in the majority of loop-plans written by students, the order in which the operations appear is "description of actions / repeat mark / end control". We have notice that the explication of the end control is not spontaneous. We think that these spontaneous plans correspond to a "go to" schema which is not recommended to use in Pascal.

In the second set of sessions, the repeat-loop is introduced as an answer to problems for which the number of repetition was known and the control was depending on this value (for instance, we have asked to write a procedure which calculates and prints out the average of 40 integers read in).

In the third step the for-loop is introduced as an economic way to realize the repeat-loop plan when the number of repetition is known.

The aim of the situations proposed to students after these sessions, was to generalize the acquired "process variable/ test variable" schema beyond of problems for which the number of repetition wasn't known and the end control depending to a calculated value of a particular variable (for instance, it is asked to write a procedure which reads in integers until their sum is greater than 10,000, prints out their average and the number of integers read in).

We analyze this data with the aim to describe different strategies used by students in looping (Samurçay, 1985a). We have categorized four types of hierarchized strategy, each caracterized by a certain level of conceptualization of variables and loop invariant. The first category is characterized by the lack of use of variables: the procedures are

focusing on the result corresponding to the specific data. The second category is characterized by the trancription of the word problem in an algeabraic formalism in which there is not yet procedural cues. The third one is characterized by the production of explicit procedure, but there are difficulties with its expression in programming code: the expressions are in a merged code composed by algebraic expression, natural language formulation and some programming codes. The last category is characterized by the elaboration and expression of procedures involving an analysis and use of the signs of elementary actions with structured elements: the student encounters some difficulties in the expression of initialization and the accumulating variables.

A last set of situations is proposed to introduce the "test variable / process variable" schema (while-loop plan) as a solution of problems in which the end control depends on a particular value of a variable which will be processed in the loop execution; i.e., the variable is processed if its values satisfy a given condition. The first problem is to write a procedure which calculates the average of a set of integers read in. The end of the set is indicated by the number 9999 which should not to be considered in the sum).

We assumed that to solve this problem the students will use a repeat-loop plan and some additional instructions, because they do not learn yet the while-loop plan. The analysis of the different procedures used in this session shows that the majority of students add first the 9999 to the sum in the loop body and then subtract it from the sum before calculating the average. The problem is then transformed to a new one in which it is assumed that this "adaptative" solution would not work: the students have to calculate the sum of the inverse of a set of integers, the end of the set is indicated by 0. Hence if the student use the preceeding procedure "add first and then subtract" the solution will not work because 1/0 is not defined. Then the while-loop plan is introduced as an appropriate solution of this problem.

The analysis of clinical observations shows that students' main difficulty is the elaboration and the expression of the loop invariant: instead of a representational system in terms of tranformations with an invariant of own actions, they have to construct a representational system in terms of successive states with a relational invariant.

Based on these observations we have constructed a questionnary with hypotheses extracted from the clinical data. The first hypothesis concerns the variable concept: we distinguish two types of variables according to their functional meaning for the student on problem solving: external variables corresponding to the values controled by the program user, i.e., variables which are explicit inputs and outputs of the problem; internal variables corresponding to the values controled by the programmer itself, i.e., variables which are necessary only for the programmable solution of the problem.

We hypothesize that internal variables are conceptually more difficult to construct than external variables. They request representations of the computer operation in terms of system states. The second hypothesis concerns operations on the variables in a loop plan: update, test and initialization. We hypothesize that these operations will not be on the same level of complexity for the student. For instance, the initialization (make an hypothesis on the initial state of the system) will be more difficult than the operations of updating and test, because it is not a relevant operation for the "hand solution" of the problem. Moreover these operations may be more or less complex with respect to the type of variables on which they will operate. For instance the updating of a counter variable is more easier than the update of an accumulating variable. One of the reasons is the fact that updating counter variable involves to add a constant

counter ←---- counter + 1

although updating an accumulating variable necessitates to add another variable to the last one.

The questionnary is composed by a set of computer program texts in which one or more instructions are missing: the missing instructions are chosen according to the hypothesis we have done above. The students were asked to complete these instructions. The results are summarized on the following table:

operations	variable type	
	internal	external
initialization	1	9
updating	12	22
test	10	20

N = 26

The results confirm our hypothesis and suggest that the counter variable schema is acquired and used better than the other schematas by our students.

6. Concluding remarks

A first glance, the population we have observed and the class of problem we have studied may appear not directly relevant for the studies of HCI in professional context with more experimented subjects. However the problems of training and learning for the use of complex computerized systems are actual and we argue that the didactical approach which consists to design experimental learning situations are useful to obtain interpretable experimental data on the process of learning and to elaborate instructional hypothesis.

The strategies we have identified in this study concerning the novice's behaviour in programming are convergent with some other studies which show also that the initial representations of novices are characterized by the transer of familiar procedures of problem domain and the mental execution of these procedures.

7. References

Anderson, J.R.& Jeffries, R.(1985). Novice LISP errors: undetected losses of information from working memory. **Human-Computer Interaction**, 1, 2,107-132.

Bonar, J. & Soloway, E. (1985). Preprogramming knowledge: a major source of misconceptions in novice programmers. **Human-Computer Interaction**, 1, 2, 133-162.

Hoc, J.M. (1981). Planning and direction of problem-solving in structured programming: an empirical comparison between two methods. **International Journal of Man-Machine Studies**, 15, 363-383.

Hoc, J.M. (1983). Analysis of beginner's problem-solving strategies in programming. In T.R.G.Green, S.J.Payne, G.van der Veer (eds), **The psychology of computer use**. London Academic Press.

Samurçay, R. (1985a). Programming learning: an analysis of looping strategies used by beginner students. **For the Learning of Mathematics**, 5, 1, 37-43.

Samurçay, R. (1985b). Learning programming: constructing the concept of variable by beginning students. **Proceedings of 9. PME Conference**, Netherlands, Juillet 1985, 77-82.

Soloway, E., Ehrlich, K., Bonar, J. & Greenspan, J. (1982). What do novices know about programming? In B. Schneiderman & A. Badre (Eds.), **Directions in human-computer interaction** (pp. 87-122). Norwood, NJ : Ablex.

Representation of Computer Programs in Memory: Comprehending and Recalling Computer Programs of Different Structural and Semantic Complexity by Experts and Novices

Detlef Widowski and Klaus Eyferth

1. Theoretical considerations

In one of the first empirical studies concerning the comprehension of computer programs SHNEIDERMAN (1976) used a paradigm developed by DeGROOT (1965,1966) to investigate expertise in complex cognitive skills. DeGROOT studied the factors underlying the performance of chess masters by presenting both experts and novices with the task of finding a good move in realistic game positions. Surprisingly, he found out that chess masters and weaker players do not use different thought processes and problem solving strategies. Both groups, for example, considered a similar number of moves, calculated to similar depths, made the same number of fresh starts, and analyzed a similar number of moves per minute. The only major difference noted by DeGROOT was that the masters invariably analyzed qualitatively stronger moves than the weaker players, which doesn't really explain very much. DeGROOT then confronted players of different strengths with chess positions taken from real games and with positions with randomly arranged pieces for a short time. The configurations had to be reproduced from memory. In this task the masters showed remarkable advantages in reproducing the meaningful constellations. It is important to note that this result could not be attributed to a generally superior short-

term memory capability of the masters because no significant differences were found in reproducing the random positions. CHASE & SIMON (1973) repeated this experiment with broadly the same results. Thus, years of chess experience seem to result mainly in gaining perceptual schemata which allow for encoding information in increasingly larger and more differentiated chunks. This leads to more and more adequate problem representation rather than the acquisition of sophisticated problem solving skill. Using a computer simulation model SIMON & GILMARTIN (1973) guessed that experienced chess players have stored about 10.000 to 100.000 patterns of chess positions in long term memory.

Similar experiments were conducted in other semantically rich domains (e.g. REITMAN 1976, EGAN & SCHWARZ 1979, LARKIN, McDERMOTT, SIMON & SIMON 1980). With respect to problem solving research these experiments signalled a switch from the classical analysis of problem solving strategies (NEWELL & SIMON 1972) to the analysis of knowledge structures enabling expert performance in semantically rich domains.

This paradigm was first applied to the domain of computer proramming by SHNEIDERMAN (1976) as cited above. SHNEIDERMAN presented two FORTRAN programs to four groups, ranging from non-programmers to professional programmers. One of the programs was presented with its lines in executable order. The other consisted of the same lines in randomly shuffled order. The results of his study resemble those of deGROOT, revealing large differences between experts and novices in reproducing the meaningful program, but nearly equally poor

reconstructions of the shuffled program.

SHNEIDERMAN & MAYER (1979) formulated a theoretical model of "syntactic/semantic" interactions to account for these results which presents a framework for describing different behaviors involved in computer programming e. g. program composition, program comprehension, debugging, modification, and acquisition of programming skills. Basically their model can be seen as an extension of the chunking concept underlying the research mentioned above.

SHNEIDERMAN & MAYER differentiate between two components in the understanding of programs: semantic and syntactic knowledge. "Semantic knowledge" means general concepts or schemata of programming that exist independent from specific programming languages. This is, for example, the concept of an array, or of conditional branching, or - on a higher level of conceptual structuring - knowing a variety of sorting routines. In contrast, "syntactic knowledge" involves knowledge of, e. g. the format of iteration, details of conditional or assignment statements, or valid character sets of specific programming languages. Syntactic knowledge is acquired by rote memorization, which must be frequently rehearsed to prevent forgetting, whereas semantic knowledge is acquired through dealing with programming problems and is stored as meaningful patterns or chunks of information.

The work of SHNEIDERMAN (1976) and of SHNEIDERMAN & MAYER (1979) has been quite influential and successful as can be seen by a number of studies concerning various aspects of programming e.g. a) organization of programming knowledge, b)

measuring program quality and c) measuring programmer ability (LOVE 1977, SHNEIDERMAN 1977, DI PERSIO, ISBISTER & SHNEIDERMAN 1980, ADELSON 1981, McKEITHEN, REITMAN, RUETER & HIRTLE 1981, MYNATT 1983, CHAUDHARY & SAHASRABUDDHE 1983, KAHNEY 1983).

Nevertheless, the model has been criticized for several reasons. ARBLASTER (1982) for example calls into question whether semantic knowledge can be as language independent as SHNEIDERMAN & MAYER's view suggests and ,thus, whether it can be separated from syntactic (and pragmatic) knowledge of a particular programming language.

Whereas he could not bring forward compelling empirical evidence, GILMORE & GREEN (1984) followed ARBLASTERS critical arguments and designed an experiment to test whether information extracted from different notations of programs are translated into the same "mental language" as the SHNEIDERMAN & MAYER theory would suggest. From their findings they conclude that a comprehended program is not stored in a uniform mental language and that a better explanation of their results can be achieved by their own theory, which stresses the mental operations actually performed in the process of comprehension rather than the postulated features of the internal representation of the program.

In our paper we want to put forward a similar argument, namely that for a theory of program comprehension (and composition) it might not be sufficient to focus only on internal knowledge structures of the chunk type as in the theory of SHNEIDERMAN & MAYER. It might also be of interest

to gain insight into the operative features of this knowledge. And moreover whether it might be necessary to take into consideration additional components of programming knowledge as for example domain specific strategies or heuristics for extracting meaning from complex or unfamiliar programs.

As an illustration of the latter point we want to report on an experiment conducted at the Technical University of Berlin as part of a research project that is concerned with human factors in man machine systems.

2. The experiment

We used SHNEIDERMAN's (1976) experiment as a starting point with three mayor differences.

I. The first difference concerns how the programs were presented to the Subjects: SHNEIDERMAN attributed the variations between experts and novices to different styles of encoding, but he uses reproduction as his only criterion. Reproduction, however, reflects: a) decoding processes, and b) only the final and static results of the process of program analysis. Thus, it seemed important to us not only to collect the reproduction data, but also to register the searching strategies. In order to do this, we asked our Subjects to work through the program by moving a keyboard-controlled window from line to line. In this procedure the Subjects can never see the entire program at once. The computer was programmed to register the reading time for each

line, the sequence of lines read, the runs of lines without repetitions, the leaps backward, etc. A complete account of the search could thus be obtained.

II. Instead of comparing meaningful programs with meaningless program sequences, in which the lines of the programs were randomly shuffled, we constructed programs varying in the extent of meaning between quite stereotyped, well known routines and non-stereotyped algorithms which are syntactically correct, but which can hardly be interpreted in terms of their content. We call this factor "semantic complexity".
As described above, SHNEIDERMAN & MAYER differentiated between structures of syntactic and semantic knowledge. According to their theory, the advantage of the experts in reproducing program statements lies in their capacity to match features of a given program with semantic schemata. However, their theory is not entirely convincing. It would be unable to account for our hypothesis that the experts will also be better at handling semantic complexity. I. e., experts will also have an advantage in reconstructing unusual, non-stereotyped programs. In these cases no specific semantic schemata are expected that might be matched successfully. Thus, we postulate an additional heuristic competence of experts.

III. A third point was the introduction of a new experimental factor: the varying of structural or logical complexity of the programs to be reproduced. According to a measure of

complexity by McCABE (1976) (based on the branching of a program) we constructed stereotyped and non-stereotyped programs on two homogeneous levels of complexity.
26 Subjects joined the experiment with repeated measurement and two factors, namely semantic and structural complexity. 13 were experts and 13 were novices.

The dependent variables were the number of functionally correct reproduced lines, the number of runs needed in reading the program, the length of such runs, and further the quality of handsimulation of each program. In addition we analyzed the reading times per line, and the reproduction errors.

Three three-way analyses of variance were run for the dependent variables a) correctly reproduced lines, b) mean number of runs, and c) mean length of the runs.
The primary results of the study can be summarized as follows:

I. The analysis of the recall data supported our hypothesis that the experts' advantage is not restricted to usual, well-known or stereotyped programs but that they also perform significantly better than novices when confronted with unusual, non-stereotyped programs. In other words, we found no interaction between the factors 'expertise' and 'stereotypedness' of the programs as should be expected from the SHNEIDERMAN & MAYER model. Thus, it would seem that factors other than semantic knowledge are involved in the

expert analysis of complex programs, such as domain specific heuristics for extracting and combining information as suggested above.

II. The measurements of reading behavior indicated that compared with the novices the experts' reading behavior varies much more with variations of the induced factors of structural complexity and stereotypedness of the programs. That is, experts show much more flexible reading strategies. The less stereotyped and more branched a program is, the more runs are chosen. This is true only to a much lesser extent for the novices. The experts generally use less (and the longer) runs the simpler the texts are. But in reading the non-stereotyped, logically complex program, they even choose more (and smaller) units than the novices.
We assume that the variability in the experts' reading behavior indicates the use of various distinguishable types of information processing. In the case of the stereotyped programs, which are read by the experts in a few, long runs, a top-down or highly conceptually driven processing seems to prevail. However, in the case of the unusual programs, read in many, short sequences, a more heuristic and bottom-up oriented processing occurs, whereby local features are thoroughly analyzed and transformed into chunks step by step.

III. An analysis of variance of the number of runs reveals a strong interaction between programming expertise and stereotypedness. It should be noted that we did not find this type of interaction in the recall data. But this lack of

interaction is consistent with our interpretation, that the experts profit more from their semantic schemata in stereotyped programs, whereas they switch to heuristic strategies in non-stereotyped programs.

IV. With respect to the complexity metric of McCABE (1976) we found, that this measure can only account in part for the mental effort in understanding programs. Although we did find significant differences between the two levels of structural complexity these were also obtained for the levels of semantic complexity. Moreover, we found a significant interaction between semantic and structural complexity, which indicates that both factors cannot be combined simply by addition to predict the difficulty of the task.

V. A detailed analyis of the other measurements taken e. g. reading times of single lines and errors in reproduction is still in progress.

3. General Conclusions

It would seem promising to investigate the operative aspects of knowledge structures in the process of comprehension in more detail than SHNEIDERMAN & MAYER and others did. While they were focussing primarily on features of static knowledge structures, we were interested in features of the processes by which a mental representation of a computer program is built up. In addition we argue that the SHNEIDERMAN & MAYER model of syntax/semantic interactions makes no account for heuristic capabilities of expert programmers.

References

ADELSON, B.: Problem solving and the development of abstract categories in programming languages. Memory and Cognition, 9, 422 - 433 (1981).

ARBLASTER, A.: Human factors in the design and use of computing languages. Int. J. of Man-Machine Studies, 17, 211 - 224 (1982).

CHAUDHARY, B. D. & SAHASRABUDDHE, H. V.: Two dimensions of program complexity. Int. J. of Man-Machine Studies, 18, 505 - 511 (1983).

CHASE, W. G. & SIMON, H. A.: Perception in chess. Cognitive Psychology 4, 55 - 81 (1973).

DeGROOT, A. D.: Thought and choice in chess. Den Haag: Mouton 1965.

DeGROOT, A. D.: Perception and memory versus thought: some old ideas and recent findings. In: KLEINMUNTZ, B. (Ed.): Problem solving. New York: Wiley, 1966.

DI PERSIO, T., ISBISTER, D., SHNEIDERMAN, B.: An experiment using memorization/reconstruction as a measure of programmer ability. International Journal of Man - Machine Studies 13, 339 - 354 (1980).

EGAN, D. E. & SCHWARTZ, B. J.: Chunking in recall of symbolic drawings. Memory and Cognition 7, 149 - 158 (1979).

GILMORE, D. J. & GREEN, T. R. G.: Comprehension and recall of miniature programs. Int. J. of Man-Machine Studies, 21, 31 - 48 (1984).

KAHNEY, H.: Problem solving by novice programmers. In GREEN, T. R. G., PAYNE, S. J. & VAN DER VEER, G. C. (Eds.): The psychology of computer use. London: Academic Press, 1983.

LARKIN, J., McDERMOTT, J, SIMON, D. P. & SIMON, H. A.: Expert and novice performance in solving physics problems. Science 208, 1335 - 1342 (1980).

LOVE, T.: An experimental investigation of the effect of program structure on program understanding. ACM SIGPLAN Notices 12, 105 - 113 (1977).

McCABE, Th. J.: A complexity measure. IEEE Transactions On Software Engineering SE 2, 308 - 320 (1976).

McKEITHEN, K. B., REITMAN, J. S., RUETER, H. H. & HIRTLE, S. C.: Knowledge organization and skill differences in computer programmers. Cognitive Psychology, 13, 307 - 325 (1981).

MYNATT, B. T.: The effect of semantic complexity on the comprehension of program modules. Int. J. of Man-Machine Studies, 21, 91 - 103 (1984).

NEWELL, A. & SIMON, H. A.: Human problem solving. Englewood Cliffs, N. J.: Prentice Hall, 1972.

REITMAN, J. S.: Skilled perception in GO: Deducing memory structures from interresponse times. Cognitive Psychology 8, 336 - 335 (1976).

SHNEIDERMAN, B.: Exploratory experiments in programmer behavior. International Journal of Computer and Information Sciences 5, 124 - 143 (1976).

SHNEIDERMAN, B.: Measuring computer program quality and comprehension. Int. J. of Man-Machine Studies, 9, 464 - 478 (1977).

SHNEIDERMAN, B. & MAYER, R.: Syntactic/Semantic Interactions in Programmer Behavior: A Model and Experimental Results. International Journal of Computer and Information Sciences 8, 219 - 238 (1979).

SIMON, H. A. & GILMARTIN, A.: A simulation of memory for chess positions. Cognitive Psychology 5, 29 - 46 (1973).

SIMON, D. P. & SIMON, H. A.: Individual differences in solving physics problems. In: SIEGLER, R. S., (Ed.), Children's thinking: What develops? Hillsdale, N. J.: Erlbaum, 1978.

TRANSFERT ET ANALOGIES

Session B1

TRANSFER AND ANALOGY

Analogical thinking in human-computer interaction

Carl Martin Allwood, Mikael Eliasson and Torbjörn Wikström
University of Göteborg, Sweden

The advent of the psychology of human-computer interaction has contributed to bring the phenomenon of analogical thinking to the forefront of researchers' interest. In this paper we present two studies (Allwood & Wikström 1986, and Allwood and Eliasson, 1986, referred to below as Study 1 and 2, respectively) that are concerned with the role of mistaken analogical thinking in learning to use three different programs on a personal computer. Only aspects relevant to this topic will be presented. Analogical thinking is usually described as thinking that involves the application of knowledge from one (source) domain to another (target) domain (Anderson 1983, Carbonell 1981, Douglas & Moran 1983, Gentner 1982, Gick & Holyoak 1980). For example, learning to use a new program involves drawing on old knowledge in order to develop new knowledge structures. It should be noted that there does not seem to be any minimum semantic distance that is necessary between the source and the target domain. Schank (1980, p.267) argues similarly that there is no basic difference between analogical thinking, reminding and understanding in general. Likewise, Gick and Holyoak (1980) argue that "the "semantic distance" between analogous domains can vary a great deal" (p. 307).

The generation of errors and difficulties may profitably be seen from a memory perspective. What knowledge will be retrieved from memory depends on what knowledge is stored, how it is stored and on what knowledge has recently been and is presently activated by the total situation that the person is in. Difficulties and errors will occur when the person in the retrieval phase overlooks constraints in the target domain which do not exist in the source domain. These considerations imply that errors and difficulties may be generated by multiple causes (compare Norman 1981, with respect to slips). In the studies presented below we try to take more of this complexity into account than has been the case in previous research.

In a previous study of novices' difficulties when using a text editing program, Douglas and Moran (1983) found that a large proportion of the users' errors (59%) could be analyzed as being due to erroneous application of knowledge of typewriting. Our conclusions modify those of Douglas and Moran (1983) on the role of the the typewriter analogy in human-computer interaction. Our results indicate that to fully understand the role of analogical thinking in human-computer interaction, analogy sources other than the typewriter or office analogy must also be considered.

Study 1

Two commercially available programs were used in the study: Multiplan, a spreadsheet calculator program and Condor, a relational data base management program. Accompanying each program was a self-study course which was used, without modifications, as the instructional material for the study. Subjects were instructed to "think aloud" while they carried out the exercises on the computer. Recordings were made of the computer screen and

subjects' verbalizations. Four subjects, three graduate students in psychology and one with a Ph.D. degree in psychology participated in the study. Two subjects were females and two were males. None of them had any previous experience of the types of programs used in this study, but they had all had several years' experience of computers for word processing and statistical analysis.

Results

The analysis concerned three sources of analogies, namely, 1) other parts of the program in use, 2) other computer programs and 3) the typewriter and paper/pencil. We used three criteria for deciding whether subjects' difficulties were due to analogical thinking or not. However, our analysis is still of a somewhat tentative nature. According to the first criterion, the analogy has to be the main cause of the difficulty. Secondly, if a subject thinks that he or she is in a part of the program where the erroneous command would have been correct, the erroneous command is not considered a case of analogical thinking. Thirdly, analogies have to be on a relatively concrete level since on a very abstract level almost anything can be analogous to something else.

This analysis showed that most (77%) of subjects' difficulties did not appear to be determined by any of the sources of analogies considered. Analogies from other parts of the same program were responsible for 15% of subjects' difficulties. Analogies from other programs and analogies from the typewriter and paper/pencil domains were responsible for respectively 8% and 6% of subjects' difficulties. Many of the difficulties induced by analogical thinking were found in the error categories: "issues inappropriate command", "confuses rules for moving the two cursors" and "concepts specific to current program".

A further result of interest in this study was the large differences in learning strategy evidenced by the subjects. Two of the subjects were responsible for 29 of the 30 episodes where the subjects did not even attempt to follow the instructions but instead acted according to their own inclinations. This difference may have had further effects in that the same two subjects were the ones who had the largest number of difficulties and also the least success with an extra task performed after the tutorials.

Study 2

In this study the program used was the text editing program WordStar. This program can be used with different levels of help; the maximum help level was used in the study. This means that the most commonly used commands were displayed on the screen. Twenty-eight subjects, including eight secretaries, participated in the study, all of whom had no experience of word processing programs. On average, the subjects had only used a computer on one or two occasions previously. Twelve of the subjects had never come into contact with a computer before.

The subjects first read a short instruction manual containing the information necessary execute the computer tasks. In total, the subjects learned about 26 commands. Next, the subjects attempted the computer task which included inserting the discs into the

computer and performing all the necessary steps to get access to the text file using the WordStar program. Then, the subjects carried out four basic text editing tasks. Subjects were provided with a paper copy of the text with the four text editing tasks marked out. The subjects were asked to "verbalize all thoughts aloud" and to report which keys they pressed on the keyboard. Subjects' reports were recorded on tape with their consent.

Results

The data used in the analyses was the tape-recordings of subjects' think-aloud verbalizations, including the reports of their key-strokes. The time used by the subjects to carry out the computer tasks varied between 18 and 45 minutes.

An error was definied as any command (or set of keystrokes) or action by a subject that did not, to any extent, bring him or her closer towards the goal state for the task at hand. A correct command or action was coded as inefficient when there existed more than one solution path towards the goal state and the subject did not use the most efficient solution path. Examples of inefficient commands are: moving cursor character-wise when word-wise is most efficient, moving cursor or scrolling screen row-wise when screen-wise is most efficient and deleting text character-wise when deleting text word-wise is most efficient. The inefficiency of combinations of different commands was not analyzed.

In total, the novices' efforts at the tasks yielded 245 errors and 83 inefficiencies. Of the errors, 141 were semantic errors and 104 syntactic errors. We define a command as syntactically erroneous when the system does not respond at all or gives an error message. A command is defined as semantically erroneous when it is syntactically correct but does not bring the subject closer to the goal state. A set of errors having to do with the subjects' deficient way of physically executing a command were separated from the syntactic and semantic errors. This category was called physical execution errors and consisted of 20 syntactic errors and 30 semantic errors.

Analogy analysis

In the analogy analysis we investigated two different sources of analogy errors, namely analogies from the set of 26 commands that the subjects had learnt (internal analogies) and analogies from the use of a typewriter. In this study, internal analogies were identified in two ways: first, when a subject had correctly used a command (or part of a command) and then used it erroneously (priming analogy error) and second, when a subject used a command before the appropriate situation was at hand (anticipation analogy error).

Each of the 26 commands were analyzed into two parts: a command group selector and a command indicator. Examples of command group selectors are the Return and Ctrl keys. A command indicator is that part of the command which indicates which specific command it is. In general, the analysis was first performed on the entire command and if the error was not coded as due to analogical thinking, the same analysis was performed on the two separate parts of the command, one at a time.

Both sources of analogies were not considered for all errors or parts of all erroneous commands, for example only 12 of the physical execution errors were included in the internal analogy analysis. It did not make sense to perform the internal analogy analysis on the other 38 errors in this category. Furthermore, the subjects' inefficient commands were only considered in their entirety and were not analyzed with respect to anticipation analogies. Further restrictions on the analyses made with respect to analogies are detailed in Allwood and Eliasson (1986). Table 1 shows the classifications made and not made (-) for all errors and inefficient commands.

All the types of errors, except one, that were classified as typewriter analogies by Douglas and Moran (1983) were classified the same way in our analysis. The one exception is the error where the subject places the cursor in the wrong position when inserting text. We did not classify this error as due the typewriter analogy because the character is added at the current position both when typing on the computer.

Table 1

Results of analogy analysis of errors and inefficient commands (Ci = Command indicator, CGs = Command Group selector, - = no analysis performed)

	Syntactic errors	Semantic errors	Physical execution errors	Inefficient commands
All errors	84	111	50	83
Priming analogies				
Ci	2	0	-	-
CGs	25	-	-	-
Entire command	10	39	12	33
Anticipation analogies				
Ci	10	5	-	-
Entire command	14	0	-	-
Typewriter analogies	12	23	37	75
Number of determinations	73	67	49	108
Number and percentage of errors determined	57 68%	41 41%	49 98%	76 92%

In addition to the error types used by Douglas and Moran, we included two additional types. In the first of these, the subject did not distinguish between the visual appearance of the symbol and the function of the symbol. The other type consisted of errors where the subjects substituted the shift key for the control key.

Only physical execution errors that occurred because of holding a key down for too long were included in the external analogy analysis. We note that neither this category nor subjects' inefficient commands were included in Douglas and Moran's (1983) analysis of typewriter analogy errors. In Table 2, we list the types of physical execution errors and inefficient commands that were classified as due to external analogies.

The reason why the commands in Table 2 can be classified as due to the typewriter analogy is that typewriters only have two short movement units: next column, i.e. space bar, and next row, i.e. carriage return. Likewise, only one character can be deleted at a time when typing. Furthermore, typewriters only have one delete key which deletes the character in the column one position to the left of the current column. For the first three inefficient commands in Table 2, the common characteristic is that the subjects use the commands that most resemble the typewriting operators, i.e. commands that produce a character-wise change, instead of the more powerful commands available in the program.

Table 2

Different types of physical execution errors and inefficient commands analysed as due to external analogies (number of cases within parentheses)

Execution errors
Issues more instances than planned of a command due to holding a key down for too long (37)
Inefficient commands
Moves cursor character-wise when word-wise is more efficient (31)
Moves cursor or scrolls screen row-wise when screen-wise is more efficient (30)
Deletes text character-wise when word-wise is most efficient (12)
Moves cursor to the right in order to be able to use the command for deleting character to the left of cursor instead of deleting directly (using another command) in the original position (2)

Table 1 shows the number of errors and inefficient commands classified as determined by each type of analogy. It can be inferred from this table that 60% of subjects' errors were determined by at least one of the three analogy types. This

figure increases to 68% if the subjects' inefficient commands are included. It is interesting to note that, when only the error types considered by Douglas and Moran (1983) are taken into account, only 18% of the errrors were classified as due to the typewriter analogy. When the physical execution errors and the inefficient commands are added, this figure increases to 45%.

Some of the erroneous and inefficient commands were classified as determined by more than one type of analogy, ("Number of determinations" in Table 1). In 18 cases, errors and inefficiencies were determined by both anticipation analogies and priming analogies and in 51 cases by both external analogies and priming analogies.

We also considered it of interest to analyze the number of steps in the task between the source of the analogy and the occurrence of the error, since such an analysis would give information about the nature of the internal analogies. For the priming analogies, the source of the analogy was within three steps from the occurrence of the errors in 65% of the cases (when subjects' inefficient commands are included, the corresponding figure is 74%). For all the errors determined by anticipation analogies, the source of the analogy was one or two steps ahead.

Discussion

The main result from our two studies is that the typewriter analogy may not play the large role in determining subjects' errors and difficulties suggested by the results presented in Douglas and Moran (1983). A large part of subjects' difficulties appear to be due to two other sources of analogies, namely analogies from within the program used and analogies from other programs, and to causes other than analogical thinking. The discrepancy between our two studies and the one by Douglas and Moran is probably due to differences in the programs and subjects studied. Although we also used a word processing program in our second study, there may have been differences between subjects studied, for example with respect to their experience of typewriters.

There was also a difference between our own two studies with respect to the role that mistaken analogical thinking played in causing our subjects' difficulties. This may mainly have been due to differences in the kind of subjects used in our two studies. Compared to the subjects in our second study, the subjects in our first study had considerably more computer experience, they were only novices with respect to the programs investigated. It seems reasonable that the more unexperienced a person is in a domain the more prone he or she is to mistaken analogical thinking. Furthermore, the high level of education of the subjects in Study 1 may have contributed to their lesser vulnerability to mistaken analogies. This seems reasonable considering the strong relationship between successful analogical thinking, inductive reasoning and intelligence in general reported by Sternberg (1982).

Another result of interest was the great difference between subjects with respect to learning strategy found in Study 1, more specifically, the degree to which they attempted to follow the instructions in the tutorial. It would be interesting to find out if subjects who do not attempt to follow instructions are more

prone to analogical thinking than subjects who follow instructions conscientiously. In any case, the difference in learning strategy between subjects poses an additional problem for the system designer.

In Study 2, we have extended earlier attempts to analyze the role of analogical thinking in novices' text editing by including syntactic errors, physical execution errors and inefficient commands in the analysis. Analogical thinking proved to be present in all of these types of difficulties. The typewriter analogy appears to have contributed to the physical execution errors and the inefficient commands whereas the syntactic errors seems to have been more affected by analogies from the same program.

For the future, it would be of interest to investigate the specific ways in which bad program design contributes to mistaken internal analogies.

References

Allwood, C.M., & Wikström, T. (1986). Learning complex computer programs. (Accepted for publication in Behavior and Information Technology). Dept of Psychology, Göteborg University.

Allwood, C.M., & Eliasson, M. (1986). Analogy and other sources of difficulties in novices' very first text editing. Unpublished manuscript, Dept of Psychology, Göteborg University.

Anderson, J.R. (1983). The architecture of cognition. Cambridge, Mass.: Harvard University Press.

Carbonell, J.G. (1981). Metaphor comprehension. Dept. of computer science, Carnegie-Mellon University, CMU-CS-81-115.

Douglas, S.A., & Moran, T.P. (1983). Learning text editor semantics by analogy. In Proceedings CHI 83: Human factors in computing systems. New York: ACM.

Gentner, D. (1982). A theoretical framework for analogy. Cognitive Science, 7, 155-170.

Gick, M.L., Holyoak, K.J. (1980). Analogical problem solving. Cognitive Psychology, 12, 306-355.

Norman, D. (1981). Categorization of action slips. Psychological Review, 88, 1-15.

Schank, R.C., (1980), Language and memory. Cognitive Science, 4, 243-284.

Sternberg, R.J. (1982). Reasoning, problem solving and intelligence. In R.J. Sternberg (Ed.), Handbook of human intelligence. Cambridge: Cambridge University Press.

On using production systems for cognitive task analysis and prediction of transfer of cognitive skill -

An application and evaluation of cognitive complexity theory for research on direct manipulation-style interaction

J. E. Ziegler, P.H. Vossen, H.U. Hoppe

Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO)
Rosenbergstraße 17, 7000 Stuttgart 1, West Germany

Abstract

In cognitive complexity theory (Polson & Kieras, 1985) it is assumed, that performance, learning and transfer of cognitive skills are directly related to structural and dynamical features of production system models of cognitive tasks.

Based on this approach, an experiment was conducted, in which four groups of subjects had to learn four simple editing tasks in succession. The groups differed with respect to the sequence in which the distinct tasks were presented. Four functionally similar types of task were used: text editing versus graphics editing, content editing versus form editing. The subjects were novices with respect to both the tasks and the tools.

As for learning and transfer, the results are in good agreement with the predictions based on representations of the tasks in terms of the production rules: the task novelty measured by the number of new production rules correlated well with the observed learning times. However, careful statistical analysis shows significant non-linear effects in the data. We will discuss a number of research issues in an attempt to highlight the conditions and implications of cognitive complexity theory for research on human-computer interaction.

Introduction

The development of workstations with multifunctional, integrated application software has led to attempts to improve the manner in which the user is able to access the different functions of such a system for the user interface should be easy to learn and efficient to use. The more the functionality of a system is broadened, the more the ease of learning is determined by the degree of consistency of use between the different functional domains.

Since the early 80's a new generation of office workstations has become available (Xerox Star /1/, Apple Lisa /2/ and Macintosh /3/ and others /4/) which are trying to solve these usability problems by introducing a highly visualised, object oriented interaction style between user and system. Shneiderman /5/ has coined the term "direct manipulation" (DM) for this new mode of interaction. Although some authors have described characteristics of this interaction mode /6/, /7/, the concept of DM is as yet only vaguely defined.

The above mentioned issues form part of a long term research programme, which is pursued in the HUFIT (**Hu**man **F**actors in **I**nformation **T**echnology) project. This international project in which seven industrial and university partners cooperate , is carried out within the European ESPRIT programme. One of the goals of the HUFIT project is to provide a methodological framework for the formal and empirical analysis of such usability characteristics as consistency, efficiency and learnability. These usability characteristics are neither well operationalized nor empirically tested so far.

Usability is a many-facetted concept, which has to be broken down into more elementary notions referring to observable and measurable phenomena at the human-computer interface. Also, these phenomena have to be related to existing psychological models of performance, learning, transfer of learning and development of cognitive skills and competence. Only in this manner we can establish predictive measures of usability which are needed as an alternative to the costly empirical evaluation of prototypes and products. For our work on DM interfaces, predictors for learnability of cognitive skills (as a measure of consistency) are of special interest.

In order to be able to construct such a kind of predictive measure of learnability of computerized tasks, we need a combined task-user modelling tool. By this we

mean a formal language with associated methods and techniques to specify components of a task in relation to operations to be performed by a user. The cognitive complexity theory developed by Polson & Kieras /8/ offers such a framework for task-user modelling. As we have chosen cognitive complexity theory as a starting point for the experiments reported here (see also /9/), we will briefly describe its main features.

Production Systems as Cognitive Models

The term "cognitive modelling" marks the efforts to construct computational models for mental processes /10/. Most of the current research in this field is done using production systems (PS) for the representation of procedural knowledge. A PS is a set of rules, each consisting of two components - a condition and an action of the following form:

(rule) IF (condition) THEN (action)

What makes PS-rules much more flexible and powerful then IF-THEN-statements in classical programming languages is the interaction of a rule base (set of PS-rules) with a working memory (WM). The basic mechanism of this interaction is the recognize-act cycle. In the first step of a recognize-act cycle the conditions of the currently used rules are matched against the WM entries in order to determine the possibly applicable rules. A conflict resolution mechanism decides, which of the selected rules will "fire" (i.e. the action part of the rule is evaluated). A very simple but in some cases sufficient solution is to write such specific rules, that only one rule applies in a given context of WM.

Rules interact with WM by adding or deleting items. Those "internal" actions are the only means for flow of control and information between different rules. In other words: rules communicate only via WM. WM contains descriptions of current goals, notes about actions that have been executed and assumptions about the current state of the "external" world. Therefore, in cognitive models WM is identified with short-term memory. PS-rules can be regarded as independent, atomic units of procedural knowledge in long-term memory.

The use of PS for modelling behaviour can be traced back to Newell and Simon's work on human problem solving /11/. Problem solving is modelled there as a goal directed activity, which is comprised of finding a sequence of operators leading

from an initial state to a final goal. Tasks, as opposed to problems, do not require search in a problem space, but are associated with a fixed solution strategy. The distinction between tasks and problems can indeed only be made with respect to a certain human subject.

But how is task knowledge represented? The notions of goals, methods and operators originally developed in the context of problem solving can easily be used to represent task-specific knowledge. The GOMS model introduced by Card, Moran, and Newell /12/ is a realization of this central idea. It uses higher-level descriptions than PS and is not directly computable. Polson and Kieras showed how GOMS-like descriptions of user skills can be mapped to PS and thus be made computable.

The already mentioned theory of cognitive complexity is used to analyze and predict learning and performance of routine skills as occur e.g. in the use of editors. Predictions of observable parameters are derived in the following ways: execution time for a given task is assumed to be proportional to the total number of recognize-act cycles in the simulation run. The time to learn a new method is considered to be a linear function of the number of new productions which have to be acquired.

In cognitive complexity theory, production rules are treated as elementary units which are assumed to be learned in an all-or-none fashion.In our study we used the following criterion: three correct task performances in succession without experimenters' intervention The amount of transfer between two different tasks can be defined by the number of shared productions relative to the total number of productions needed for the new task. This is an application of the classical "common elements" theory of transfer /13/.

The editing system

The tasks for the subjects consisted of editing specially prepared texts and graphics on a Xerox Star Office System. Two stand-alone systems were used, allowing us to manage two parallel experimental set-ups.

The specific system chosen exhibits many of the features characterizing the DM style of interaction. Functionally very similar tasks can be defined for text and graphics editing (extensively using universal commands), so that between both

domains there are a lot of common procedures, which makes it specially suitable for research on transfer of learning.

Selection and design of the tasks

Four types of tasks were used, differing with respect to the domain of editing (texts versus graphics) and with respect to the target of editing (content versus form). The four possible combinations are as shown in figure 1. For each task type we designed ten task instances of the same mental and motor difficulty, so that each subject was allowed a maximum of ten trials to reach the learning criterion.

In order to study transfer effects we chose four different task sequences to be given to four different experimental groups (figure 2). These sequences were chosen so that each group of subjects started out with a different type of task and such that there were marked differences in the change of task novelty over sequences (figure 3). With this design, each subject in each group would have learned exactly the same set of 82 production rules.

Subjects

88 paid subjects participated in our experiment, 22 in each of the four groups. Most of them were students in the range between 20 and 30 years. The only selection criterion was, that they had no previous experience with texts or graphics editing on a computer system.

Subjects were randomly assigned to the four groups, which were run quasi parallel in order to exclude as far as possible differences between the groups due to availability of subjects or experimenters.

Procedure

Duration of the experiment was limited to a maximum of four hours for each subject. At the start the subject received a general instruction which explained the basic aspects of the editing system. The subject was allowed to explore the functions of the pointing device, i.e. the mouse, and the associated positioning of

the cursor at the screen. Also, the overall course of the experiment was explained to him.

At the start of each session, a specific instruction was given in order to explain both the type of task and the task-specific object-selection method for that session. Subjects were encouraged to discover for themselves all other details of the task methods, i.e. how to apply the editing instructions on the mark-up documents to the electronically displayed documents. Subjects were allowed to make and correct errors while performing the task, as we felt that error detection and correction is part of normal cognitive skill acquisition. Subjects were free to ask for any additional explanation about the structure of the task or the method of performing the editing operations. However, the experimenters were instructed to give explanations about the interpretations of the editing instructions, but not about the physical operations to be executed. This prevented subjects from just imitating the manipulations of the instructor. The subject would indicate the end of a trial, which was then checked by the experimenter. Within each session subjects were allowed a maximum of ten trials of the same task type.

The order of instances of a task type was randomized for each subject. A session was terminated by the experimenter as soon as the subject had performed three subsequent instances of the task without any additional instructions by the experimenter. We assumed this criterion to be a sufficient indicator for learning.

Experimenters

In fact, the role of the experimenters was threefold: the usual role of managing the experiment, a second role as instructor/interpreter for the subject and a third role in recording the experiment data.

According to a Graeco-Latin design, four experimenters were randomly assigned to the four groups of subjects and thus to the four types of task. For organizational reasons we were forced to have another two experimenters act as substitutes. The experimenters were instructed, when and how to give help to the subject and how to classify errors made by them.

Data collection and recording

A separate data collection program was used to record times and errors. Time was accumulated in one of three counters according to the kind of interaction the subject was engaged in: non-task-specific (mainly external interruptions and intersession pauses), experimenter- or subject-initiated help time, and proper editing time.Three types of errors during proper editing time could be recorded: errors while selecting objects, errors in chosing editing functions, and errors while changing object properties. Apart from that the experimenter had to judge whether the trial was succesful or not and thus whether the session should be continued or not. This judgment was filled in on a separate form allowing us to cross-check the stored data.

Results of linear regression analysis

As a result of evaluating the course of the experimental runs and data per subject, 7 subjects were dropped from the final analysis.

First of all the data was checked for experimenter effects in groups. However, from the analysis of variances there was no indication of experimenters effects, due to the large variability between subjects. Therefore, there was no need to include the experimenter in the statistical model for further analysis. Likewise, no significant effects of the sex, age, or educational background of the subjects were found.

We did not analyse in detail the data about the number of trials per session per subject, the distribution of types of errors and the distribution of help given to the subject, as these data do not form part of the underlying PS-model. The experimenters did not report any systematic differences between groups or between subjects within groups in these respects. As is to be expected, the number of trials per session appears to drop over sessions.Table 4 shows the result of a linear regression analysis applied to the four groups simultaneously. Here the learning times have been averaged over subjects and regressed upon the number of new production rules, both per session per group.

The linear fit between these variables appears to be very good ($r^2 = .88$). We found a factor of 17.2 seconds for each production rule, which is smaller than the estimates found by Polson and Kieras (ca. 30 sec). A plausible explanation for this

difference is that our training procedure involving a human tutor is more efficient than the one used by Polson and Kieras, where subjects had to read unspecific help material presented on a separate screen.

Although we did not find significant differences between the groups in all other respects, we performed also regression analyses on the four groups taken separately. The resulting correlation coefficients are smaller then the overall correlation coefficient, but still good. Besides there seemed to be no apparent interpretation of the differences found between the four estimates of production rule learning times.

Discussion

Although our results confirm the results found by Polson and Kieras /8/, being in good agreement with the predictions on basis of the PS-models, there are some problems regarding the applicability of a linear regression analysis on the data.

The relevant condition for the application of linear regression analysis is homogeneity of variances over and equal covariances between the sessions. The reason for the non-homogeneity of variances and covariances over sessions certainly has to do with the fact, that the predictor variable "number of new production rules" is confounded with time, i.e. with the position of task types in task sequences. A further analysis of the distribution of learning times showed, that these cannot be treated as coming from a normal distribution. This fact excludes the use homogeneity tests like Bartlett's, as these are based on the assumption of normality of the data.

Currently we are looking at orthogonal contrast analysis (repeated measurement analysis) for a better understanding of the statistical characteristics of the data. The results found so far indicate that not only a linear effect (accounting for 45% of total variance) but also non-linear, i.e. quadratic (35% explained variance) as well as cubic, effects between the number of new productions and the actual learning times, over sessions, are present in all groups except the first. .

Looking for potentional explanations for these findings we have to consider a number of factors which could influence the observed learning times, and then have to be analysed in more detail.

- The overall learning times observed are composed of a part related to learning and a part needed for performing old and new productions. For the performance component, the number of recognize-act cycles of the PS-model can be used as an additional predictor.

- It might be the case, that in our experimental conditions productions are not yet sufficiently learned when the criterion is reached. This can invalidate the assumption that productions aretransfered at no cost.

- An additional amount of time may be needed for generalizing productions, i.e. it takes additional effort to apply old rules in a new context.

Cognitive complexity theory is not a theory of individual differences. Nevertheless, it seems that we need better explanations for the large individual differences. This could lead to a stochastic rather than a deterministic model. Also, the issues of errors and instructions which are not addressed in the current theory will require more attention in the future.

An important improvement of the applicability of the theory could be achieved by providing a formalism for task description which is more compact than production rules. Metarules for deriving productions from such a description (in an automatic manner) are required. Such a tool would also greatly facilitate the design of tasks for experimental studies.

By way of conclusion we like to recall that the predictions of cognitive complexity theory are in good agreement with our empirical findings; at the same time we call attention to the fact that refinements at the theoretical (Opwis/14/) as well as the experimental level are necessary .

Acknowledgements

The statistical analyses have been carried out by Miss S. Evans. We like to thank her as well as the experimenters (Miss M. Mikusin, Mister M. Wöhrle, Mr. F. Koller, Mr. M. Latzina and Mr. J. Herczeg) for their valuable assistance in various stages of our research.

References

/1/ Smith, D. C. et. al.: Designing the Star User Interface. In: Degano, P. and Sandewall, E. (Eds.): Integrated interactive computer systems. Amsterdam, North-Holland. 1983.

/2/ Williams, G: The LISA computer system. Byte 8, pp.33-50. 1983.

/3/ Williams, G: The Apple Macintosh computer, Byte 9, pp 30-54. 1984.

/4/ Hopgood, F.R.A., et.al.: Methodology of Window Management. Proc. of an Alvey Workshop at Cosener's House, Abingdon, UK, April 1985. Berlin, Springer-Verlag. 1986.

/5/ Shneiderman, B.: The future of interactive systems and the emergence of direct manipulation. Behaviour and information technology, vol. 1, no. 3, pp. 237-256. 1982.

/6/ Hutchins, E.L., Hollan, J.D. & Norman, D.A.: Direct manipulation interfaces.
In: Norman, D.A: and Draper, S.W. (Eds.): User Centered System Design - New Perspectives in Human - Machine Interaction. Hillsdale N.J., Erlbaum. 1986.

/7/ Faehnrich, K.P. & Ziegler, J.E.: Workstations using direct manipulation as interaction mode. In: Proc. Interact 1984, vol. II, pp. 203 - 208. London. 1984.

/8/ Polson, P.G. & Kieras, D.: An approach to the formal analysis of user complexity.
Int. J. Man - Machine Studies, vol. 22, pp. 365 - 394. 1985.

/9/ Ziegler, J.E., Hoppe, H.U. & Fähnrich, K.P.: Learning and transfer for text and graphics editing with a direct manipulation interface. In: Proc. CHI '86, Human Factors in computing systems, pp. 72 - 77. Boston. 1986.

/10/ Johnson - Laird, P.N. : Mental Models. Cambridge University Press.1983.

/11/ Newell, A. and Simon, H.A. : Human Problem Solving. Englewood Cliffs N.J, Prentice - Hall.1972.

/12/ Card, S.K., Moran, T.P. & Newell, A.: The Psychology of Human - Computer Interaction Hillsdale N.J., Erlbaum. 1983

/13/ Thorndike, E.L., & Woodworth, R.S.: The influence of improvement in one mental function upon the efficiency of other functions. In: Psych. Rev., vol. 8, pp. 247 - 261. 1901.

/14/ Opwis, K.: Produktionssysteme. To appear in: Mandl, H. & Spada, H. (Eds): Wissenspsychologie. München, Urban & Schwarzenberg. 1986.

Figure 1

Types of Tasks and their Mnemonics

	Content Editing	Form Editing
Text Editing	Delete Move Copy	Font Size Feature
Graphics Editing	Delete Move Copy	Linewidth Shading Texture

TC TF
GC GF

Figure 2

Sequencing of Task Types over four Sessions in four Groups

TASK TYPE	S_1	S_2	S_3	S_4
G_1	TC	TF	GC	GF
G_2	GC	GF	TC	TF
G_3	TF	GF	TC	GC
G_4	GF	TC	TF	GC

Figure 3

Number of New Production Rules per Session per Group

Task Novelty	S_1	S_2	S_3	S_4	Total
G_1	44	26	9	3	82
G_2	40	26	13	3	82
G_3	52	12	18	0	82
G_4	48	31	3	0	82

Figure 4

Predicted versus Observed Average Learning Times per Session per Group

pred *obs*	S_1	S_2	S_3	S_4	no of subjects
G_1	955 *822*	646 *636*	354 *299*	251 *218*	18
G_2	887 *791*	646 *806*	423 *412*	251 *287*	22
G_3	1093 *1143*	406 *383*	509 *376*	199 *252*	21
G_4	1024 *1280*	732 *542*	251 *275*	199 *304*	20

regression: $y = 199.5 + 17.2*x$ $r^2 = .88$

Figure 5

Mean and Standard Deviation of Learning Times per Session per Group

mean *var*	S_1	S_2	S_3	S_4	no. of subjects
G_1	822 *490*	636 *327*	299 *149*	218 *184*	**18**
G_2	791 *414*	806 *377*	412 *267*	287 *172*	**22**
G_3	1143 *466*	383 *222*	377 *204*	252 *177*	**21**
G_4	1280 *439*	542 *261*	275 *156*	304 *181*	**20**

THE INFORMATION-SEEKER´S MENTAL MODEL OF THE CONTENT IN A DATABASE

Lena Linde, Dept. of Psychology, University of Stockholm

Abstract

The effect of an information-seeker´s prior knowledge of informational content and organization on performance when learning database search principles is treated. It is suggested that knowledge of the informational content will aid the subject in constructing a mental model (Johnson-Laird, 1983) of searching in the database. Experimental data are presented from a study on computer-naive subjects´ learning of the search principles in a relational database. A significant facilitation from prior knowledge of the informational content was obtained. A theoretical relationship between these results and those obtained by Bransford and Johnson (1972) is pointed out.

Computer systems´ "invisibility"

The content and organization of information in computers are to a large extent "invisible"- only a small fraction of the content can be seen at a time and a user´s possibility to get an overview of the contents is therefore poor. Information and functions in a computer system have a logical, not a physical, structure. Whereas a book, catalogue or library has a spatial organization that is evident to an informtion-seeker, the information in computer systems is organized like, for instance, as records or networks, by file names or labelled links. The logical and conceptual structure of a database can be represented by e.g. tabular or tree-structural graphic models.

Another obstacle for the information-seeker might be the size of the informational content and the rate at which the information is updated. In large database systems, the information-seeker has little possibility of envisaging and getting an overview of the information. In a psychological experiment on database search, Linde and Waern (1985) found that the most "efficient" subjects started by searching for information that gave an overview of the informational content in the database.

The information-seeker´s possibility to find relevant information in a database system can also be assumed to be related to the limits imposed by human working memory on the logical complexity of searches that can be conceptualized.

The present talk will deal with the information-seeker´s mental representation of the informational content and organization of a database system and its relationship to his/her ability to comprehend the search principles. An information-seeker´s mental representation the informational content and organization of a database might be related to his/her prior knowledge of the domain (field) of the content, his/her familiarity with the particular

content of a database, manuals and instructions, and experience of other database systems. Results and theoretical implications of a study by Linde and Bergström (1986) will be discussed. In this study, effects of prior knowledge of domain and familiarity with content on an information-seeker's ability to learn search principles were modelled experimentally.

What is a mental model?

Mental model has been used as a psychological construct supposedly underlying the mental representation of text content (Johnson-Laird, 1983). The construct mental model pertains to an analog and visually elaborated representation, constructed from phonetic, syntactic and conceptual properties of the text. Thus, it is supposed to be a perceptual representation of propositionally encoded information, the properties of which can be tested by mental imagination, i.e. anticipation. For example, one might imagine oneself moving around in a particular room and anticipate how the pieces of furniture are placed relative to each other from different perspectives.

Linde (1986) suggested that the consequences of lack of domain-related knowledge in an information-seeking task may be either no alternatives ("feeling stuck") or too many ("feeling mental strain") in working memory. The task of figuring out the search principles in a database for a computernaive information-seeker can be regarded as a special type of problem-solving (cf Newell and

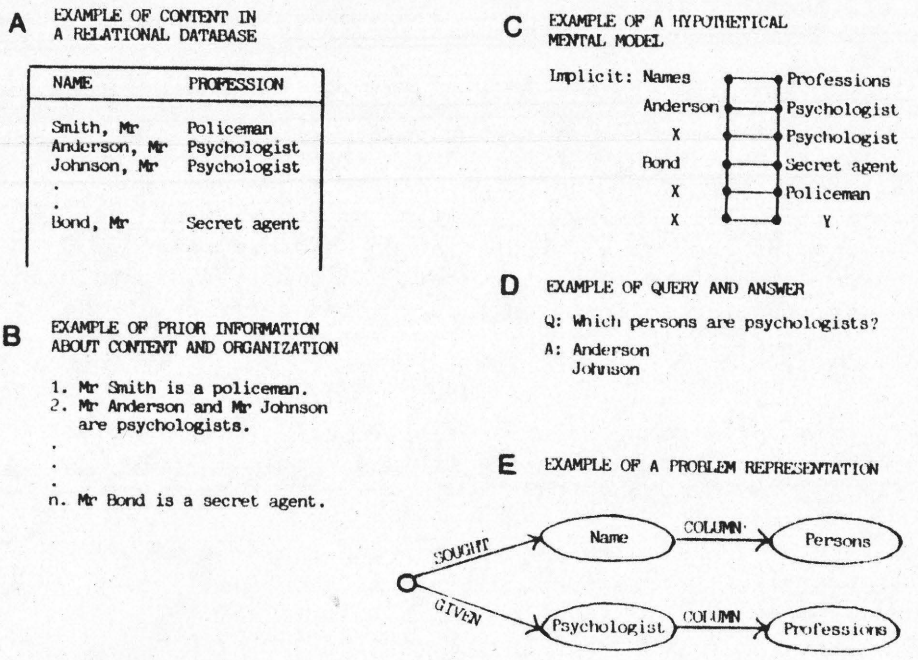


FIGURE 1. Example of informational content and mental representation of it in a database search task.

Simon, 1972). The information-seeker's problem is to analyse a query in textual form into its logical constituents and translate this result into a query language. Figure 1 gives examples to illustrate this task. First, some tabular organized information similar to that presented by Linde and Bergström (1986) is given as an example of informational content in a relational database. Second, the same information is presented in textual form. This form provided "prior information about content" in Linde and Bergström (1986). Third, a mental model hypothetically based upon the textually presented information about content (B) is shown. It is assumed that a list of names and a list of professions have been encoded and stored and that Anderson has been linked to psychologists, that name x (forgotten) also has been linked to psychologists, etc. The proposed mental model is similar in form to ones suggested by Johnson-Laird (1983). Fourth, an example of a query and an answer that a relational database typically might give, is shown. Fifth, a problem graph (cf Newell and Simon, 1972) is given to illustrate how the query can be represented in terms of logical and conceptual constituents. As I will suggest below, the result of not being able to set up a representation of the problem can be no alternatives in working memory, and the result of an unduly wide "first search" too many alternatives in working memory.

What functions can a mental model of a database have?

An information-seeker, planning to use a database, must consider questions such as : Can I find the type of content that I need in this database? Does information x have any reference to information y, and, if so, what kind of link is there between the bits of information? How do I retrieve the information I want? For example, suppose that the information-seeker wants an answer to the query: "What profession has Mr. Bond?". S/he plans to search in a relational database. S/he has to consider: Does this database comprise a class of persons including Mr. Bond? What table, or tables, and columns do I need to find to retrieve Mr. Smith's occupation? How do I translate my query into the required search dialog form. These questions pertain to the domain, conceptual organization and syntactical properties, respectively.

A mental model of the informational content in a database, similar to the one outlined in Figure 1, might have a mnemonic function when an information-seeker is introduced to a new database system and a mental model might be used to generate problem solving paths in working memory when translating queries (cf the dynamic, anticipation function suggested by Johnson-Laird, 1983).

How a mental model of the informational content can be formed

Figure 2 is a general overview of the relationships between different components in a database system and an information-seeker's knowledge. A mental model is supposed to be based on three "sources": prior knowledge of the domain, i.e. field, of the informational content in the database; actual

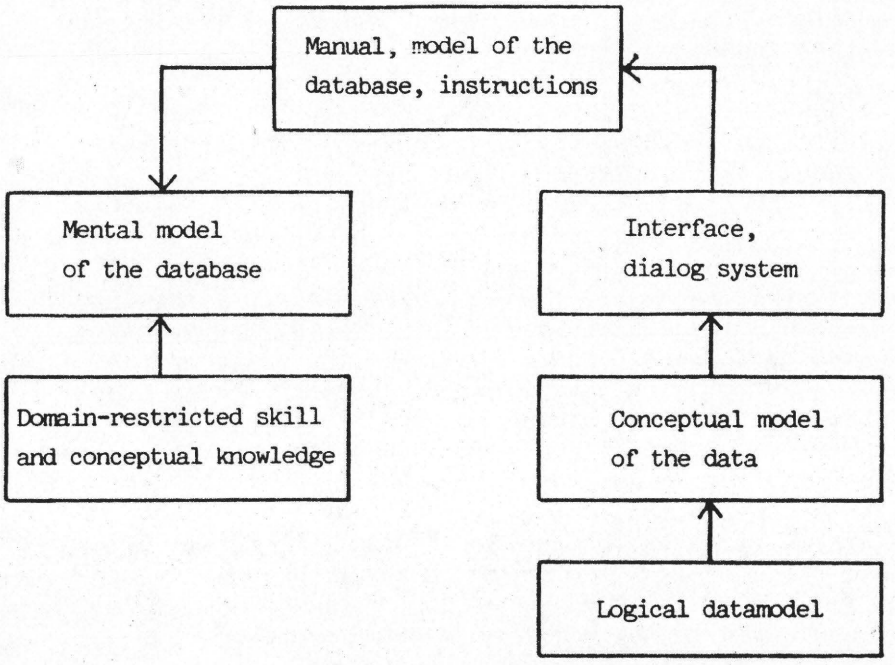


FIGURE 2. Relations between components in a database system and an information-seeker's knowledge of it.

experience of the database, that is familiarity with it; and instructions in various forms about the database. (It should be pointed out that the information-seeker's prior experience of other database systems might be an important aspect, which is not incorporated in Figure 2.)

Experimental data

Linde and Bergström (1986) made an experimental study with the aim of investigating the effects of procedural knowledge of table search and declarative knowledge of informational content on search performance in a relational database. The distinction between procedural and declarative knowledge followed Anderson (1976). Simply expressed, a relational database is based on a tabular organization of data. The database contained totally fictitious information. It was about "events" occurring at different times and different places in the south of Sweden. The content comprised three tables with five, two and three columns, respectively. The subjects had no experience of interactive computer systems. Their task was to retrieve information from the database in order to answer a set of questions varying in logical complexity and distributed into three blocks. Two independent variables were manipulated in a factorial design. One comprised prior training in

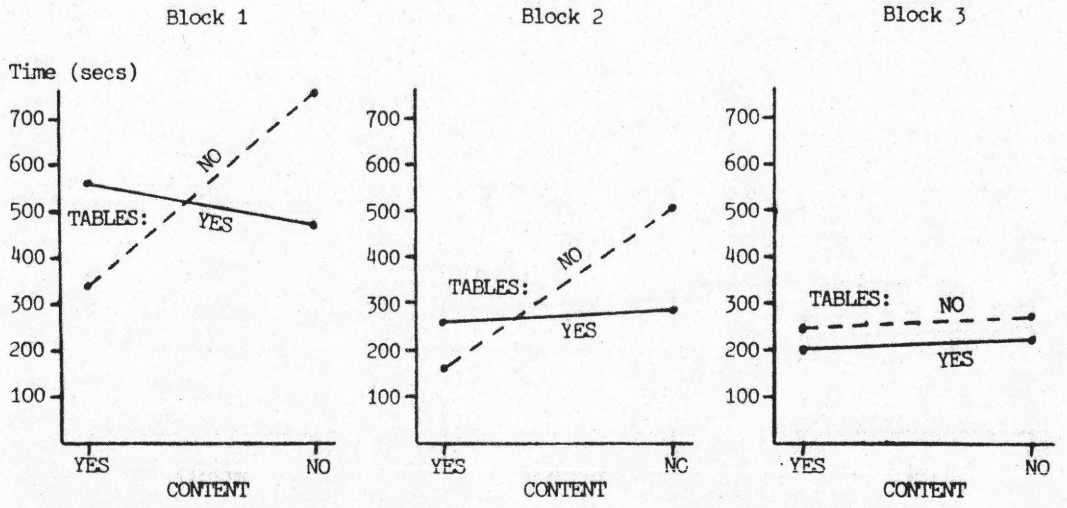


FIGURE 3. Search performance measured as average time over all search questions in a block.

the use of tabular information presented on paper vs. an irrelevant search task. The other involved making inferences from a set of written "facts" about the content in the database vs. an irrelevant choice task. The facts were about the content and abstracted essential information in the database. Search performance was measured as the time taken to complete a search task (retrieve the relevant information from the database) or as error scores for each search task. The results with search performance measured as average time over all search questions in a block are given in Figure 3. A significant effect of the training on the facts about content was obtained. A significant interaction between training on table search and on facts about content also appeared. Subjects given prior training on making logical inferences from the facts about the content, learned the search principles more quickly than subjects not given such training. However, as seen from block 1 in Figure 3, this effect was not obtained for subjects also given training on visual table search. On the contrary, table training gave a slightly reverse effect of content. The subjects given no prior training generally performed lower than subjects given either or both types of prior training.

Discussion

Why would knowledge about the semantic content in a database facilitate the learning of logical search principles and why was training on content more effective than training on visual table search? One proposed explanation is that knowledge of content affected the control schema, which guided the subjects' information analysis of the given queries, instructions and actual demonstrations of searches in the database. Bransford and Johnson (1972) found that prior information about contextual relationships, presented from a picture, in a text increased comprehension and recall. Linde and Bergström (1986) gave prior information textually about semantic links between informational bits in the database. These facts, given prior to the database search, also entailed macroinformation (implicitly) that some of the facts in the database were contextually related (e.g. that there was a relationship between events distributed into different records and tables). Thus, a mental model of contextual and semantic relationships in textual information, directly induced from a picture or indirectly from textual macroinformation, might enable a more efficient encoding and information analysis. Also, I will refer to Craik and Lockhart (1972). They argued against multistore theories of memory and suggested that persistence of memory traces was related to the "depth of elaboration", that is degree of perceptual enrichement and semantic elaboration of the traces formed from stimulus information. A mental model (Johnson-Laird, 1983) and semantically elaborated information (Craik and Lockhart, 1972) are assumed to have more precise and rich interpretation than what is available from the surface structure of a text. It might be that the training on drawing inferences from facts about the content enabled the establishment and elaboration of a mental model of the informational domain, that is "what the database was about", and that this mental model entailed contextual and semantic links between information distributed into different tables, columns and records.

Thus I am suggesting that the training on content in the study by Linde and Bergström (1986) provided the subjects with mental models similar to the example given in Figure 1, and that these mental models helped the subjects to envisage searches in the database. Also, I suggest that such a mental model of the content supported the subjects' information analysis of the query, instructions and demonstrations. Earlier, it was proposed that the translation of a search question into a proper search dialog form could be regarded as a problem solving task for an information-seeker, who is not familiar with the database system. In Figure 4 I have sketched the consequences that the lack of domain-related knowledge (and inability to construct a mental model) might have in a database search task (cf Linde, 1986). First, the information-seeker must set up a problem frame (i.e. a representation of the task). In this case I assume that this is done by analysing the query into a SOUGHT (to be retrieved) and a GIVEN (modifier) constituent and

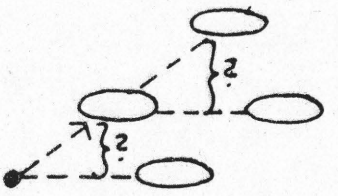
	PROBLEM FRAME (REPRESENTATION)	SEARCH (INFORMATION ANALYSIS)
PROBLEM SOLVING	e.g. SOUGHT: x SOUGHT: X COLUMN: C_x GIVEN: Y COLUMN: C_y	
INITIAL CONSEQUENCES OF LACK OF DOMAIN-RELATED KNOWLEDGE	NO ALTERNATIVES IN WM; INABILITY TO REPRESENT THE PROBLEM "STUCK"	TOO MANY ALTERNATIVES; POOR ABILITY TO MONITOR INFORMATION ANALYSIS "MENTAL STRAIN"

FIGURE 4. Query translation as problem solving; problem representation and search.

then linking COLUMN labels to these constituents. Second, the information-seeker sequentially must translate these constituents into the search dialog. In Figure 4 I have depicted this translation process in a problem space, in which the complexity (that is breadth and depth) is related to the logical complexity of the query. An information-seeker who is unable mentally to represent the basic principle of the search logic, is assumed to be "stuck" during the information analysis. On the other hand, an information-seeker who has acquired this basic logic but not yet a schema, controlling the choice and order of input of information, is assumed to be subjected to "mental strain", due to the fact that the search space is initially too "broad", i.e. too many alternative goals compete in working memory.

The presented experimental results and theoretical analysis are intended to indicate that the level of domain-specific knowledge about the content in a database affects the learning of logical search principles. Text recall and generation by individuals with a high and low level of domain-related knowledge has been investigated as a form of problem solving by Voss, Vesonder and Spilich (1980). They suggested that domain-related knowledge had a similar influence on comprehension and text generation and that the two groups of subjects differed in their ability to monitor a solution path. In a similar way it is suggested here that the level of domain-related knowledge of the content in a database will, on the one

hand, affect the efficiency of the information analysis of queries, instructions and demonstrations of the database and, on the other, the generation of a problem representation while translating the query according to the search dialog principles. The efficiency of a precise mental model of the form suggested above (and by Johnson-Laird, 1983) for information analysis and problem solving when learning search principles can be related to both encoding, inference and evaluation processes. First, it might be a sort of "mnemonic" for instructions and demonstrations initially during learning and facilitate the encoding of the queries. Second, it might help the informationseeker to make inferences related to information in the database beyond what is presented at the VDU. Third, a precise mental model can be a tool for evaluating feedback from the system. Thus, it might enable the information-seeker to detect logical errors during the search interaction. Linde and Waern (1985) in an experiment on database search, found that efficient searching was related to an ability to draw inferences from obtained information. Therefore, a lack of any mental model of the content might result in both slow learning of search principles and mental strain.

Practical implications

What practical implications regarding the design of databases, e.g. factual databanks, might be drawn from the present results and analysis concerning the importance of having domain-related knowledge of the content in a database when learning search principles? One is that incongruence between the information-seeker's prior semantic knowledge of the domain and the conceptual organization of a database, and not only syntactic properties of the query language, might obstruct the learning of search principles. Another implication may be that information-seekers might benefit from specific instructions about the informational content and organisation apart from the search principles, per se. Two ways can be suggested by which database users interactively can be taught about the organisation and content of the database. For one thing, minor prototypes, representing the essential content and organisation of large databases, may be used as instructional aids for the users. For another, Artificial Intelligence techniques for abstracting and presenting the essential informational content (gist) in databases might be developed.

References

Anderson J. (1976) Language, memory and Thought, LEA, New Jersey.

Bransford D.J. and Johnson M.K. (1972) Contextual prerequisites for understanding: some investigations of comprehension and recall, Journal of Verbal Learning and Verbal Behavior, 11, 717-726.

Craik F.I.M. and Lockhart R.S. (1972) Levels of processing: a framework for memory research, Journal of Verbal Learning and Verbal Behavior, 11, 671-684.

Johnson-Laird P.N. (1983) Mental Models, Cambridge Unviersity Press.

Linde L. (1986) What is domain skill: frames in information-seeking, Behavioral Science, in press.

Linde L. and Bergström M. (1986) Users´ mental image of the content and search principles in a database, Proceedings from International Scientific Conference: Work With Display Units, Stockholm may 12-15, 1986.

Linde L. and Waern Y. (1985) On search in an incomplete database, International Journal of Man Machine Studies, 22, 563-579.

Newell A. and Simon H.A. (1972) Human Problem Solving, Englewood Cliffs, N J: Prenctice Hall.

Voss J.F., Vesonder G.T. and Spilich G.J. (1980) Text generation and recall by High-knowledge and low-knowledge individuals, Journal of Verbal Learning and Verbal Behavior, 19, 651-667.

PREKNOWLEDGE AND THE EFFICIENCY OF SELF-GUIDED USER's LEARNING

R. Schindler, A. Schuster
Humboldt University
Berlin, GDR

Abstract

1. INTRODUCTION

At present there are many unsolved theoretical and methodical problems in designing external regulated and controlled training procedures which are adaptable to the individuality of the learner. So, the question arises if the problem of individualization can be tackled in a more appropriate way by means of self-guided learning. The problem is that totally self-guided learning, in which a learner is simply left alone in an environment does not lead to high levels of attainment.

So, we have to ask for efficient ("intelligent") combinations between external guided and self-guided learning phases.

This paper focuse on two questions

a) When people learn how to use an interactive software package, they must acquire a good deal of new knowledge in order to solve tasks/problems by means of the device. The process of designing efficient tutorial interactions has to start with the specification of the knowledge which users call upon in learning to use interactive systems. At present there are general hypothesis with regard to the knowledge required for system use (e. g. MORAN, 1981; CARD, MORAN & NEWELL, 1983; STREITZ, 1985; TAUBER, 1986) but for an efficient instruction, guidance or coaching the learner we need more detailed information of how that body of knowledge is acquired and how it change over time. So, we ask:

. what pieces of information do peoples utilize during learning at text-editing system to form a knowledge structure about the virtual machine.

b) With regard to more recent frameworks of the concept of learning (BROWN, 1984; JENKINS, 1979) one critical point of self-guided learning is, among others, that the learner is able to select and process that pieces of information from the learning environment which are relevant for meeting the goal of learning. If can be assumed that this requires some kind of prior knowledge about the learning domain. With regard to user's learning the text-editing system we ask: by what kind of knowledge about the learning domain can self-guided user's learning be make more effective?

2. METHOD

In our experimental investigations we used the method of question - asking (FLAMMER, 1981).

This method can be regarded as beeing a prototype of self-guided individual learning in communication with other persons, or written materials, as well as with the computer system itself.

The subjects had to accomplish 4 different text-editing tasks (ET) by means of the device. This tasks were selected in such a way that the cover the scope of functionality of the system. Each text-editing task consists of several subtasks (ST) each of which required different activities of the user. The sub-tasks of the text-editing tasks were partly completely different or identical or similiar to each other. The order of the 4 learning tasks was given by the leader of experiment; they was the same for all subjects.

The subjects were instructed to accomplish the 4 text-editing tasks without making errors. For meeting this goal they could ask the leader of experiment for all pieces of information that they supposed as beeing relevant

for understanding task accomplishment by means of the system. To stress this point of instruction the subjects were also asked to explain an unskilled person how the system is to operate after they had finished the self-guided learning procedure. The dialog between subjects and leader of experiment were recorded on tape.

3. CHARACTERISTIC OF USER KNOWLEDGE

Our first step of investigation was aimed at deriving all pieces of information which peoples utilize - that is,wich the ask for, or produce, or reproduce - during self-guided learning task accomplishment by means of the text editor. In other words, we were interested in the component of the human-computer interface of this particular system. Refering to MORAN (1981) and TAUBER (1986) the user interface of a system consists of those aspects of the system that a user comes in contact with - physically, perceptually, or conceptually. Any aspect of the system that enters the user's internal representations about the system is part of the human-computer interface.

To answer the question if this components only depend on the realized interface design with regard to the text-editing tasks, or if they also depend on the pre-experimental knowledge of the user we used two different groups of subjects in our first experiments. In the first group there were 6 students of psychology (VGO) having no specific knowledge about neither hard- and software of interactive computer systems in general nor about text-editing systems in particular. The second group of subjects consisted of 6 students of informatics (VGI) having knowledge about hard- and software of dialog systems, and about operating particular interactive systems but having no specific knowledge of how operating this text editor.

To meet the learning goal the subjects of both groups utilized the same classes of information. The most important one are the following:

a) Verbal statements to assign the actual user's goal(w). Such statements consist of a verb representing an anticipated transformation (e. g. "delete") and of a noun representing the objects (e. g. particular characters) on which the transformation works.

b) Combinations of alpha-numeric features displayed on screen or word markers to assign them (x. or x). They are used by subjects to represent both the intial or the anticipated perceptible state of the machine.

c) Units of actions (HE). Units of actions represent the physical action that the user has to perform in order to accomplish an anticipated object transformation. Mostly, the units of actions consist of a specific keystroke sequence on the keyboard.

d) Computer processes (SV) going on within the machine. They are utilized by the subjects to understand why a particular action unit, or a specific state of the system are to be realized by them in order to solve the text-editing tasks.

These classes of information are event types (see KLIX, 1985) having functional meaning for task accomplishment by means of the text editor. Generally, event types represent the categorisation and explanation of an external domain by a human being. They can be represented by a semantic kernel (a verb) that triggers a set of concepts which are typically related to the event (see figure 1). Event types are the building stones of the learner's internal representation. For task accomplishment by means of the device they are combined by the subjects to complex units of knowledge which we will call "interaction units". Interaction units are single or generalized steps of object transformations represented in the semantic memory of a user to accomplich tasks/ problems by means of the machine.

Figure 1 illustrate a possible interaction unit for the mental representation of a single step of object transformation.

Figure 2 shows that interaction units can consist of different combinations of event-types. It can be drawn from fig. 2 that in both groups of subjects system processes (sv) were utilized only very seldom. Hence, they can not be regarded as beeing relevant components of the human-computer interface.

Two conclusions can be drawn from these results:

1. The components of the human-computer interface do not depend on the pre-experimental knowledge of the user.
2. System processes should not play a significant role in user's training.

4. PRE-EXPERIMENTAL KNOWLEDGE AND LEARNING EFFICIENCY

In the second step of investigation we analysed the efficiency of self-guided learning of subjects having different pre-experimental knowledge about task accomplishment by means of the text editor. The groups VGO and VGI served as control groups.
Additional, three groups of subjects were used. Every group consisted of 6 students ob psychology having no relevant knowledge about the learning domain. Before starting the self-guided learning procedure by question-asking the subjects of one group - group VGR - had to learn 8 generalized rules for task accomplishment. These rules were constructed by generalization over the components of the whole number of dialog steps which are to be realized by the subjects to accomplish the 4 editing-tasks. They can be regarded as schematic interaction units which consist of classified event-types. System processes were not taken into account.

For instance:

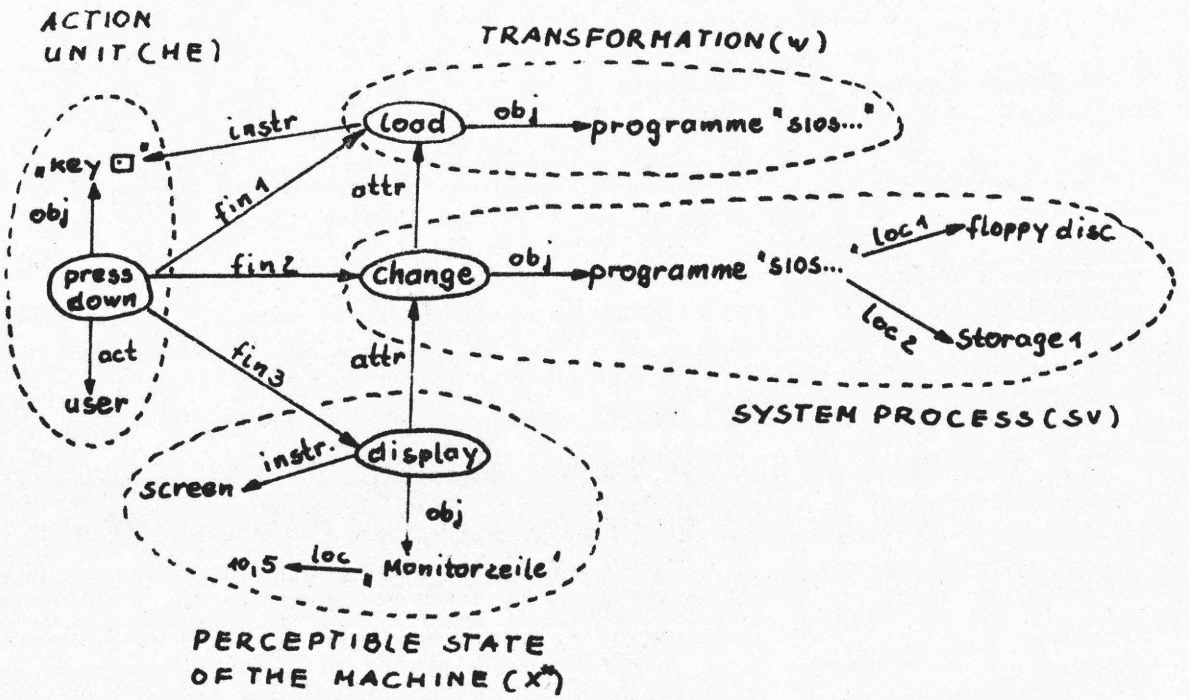


Figure 1: Interaction unit "load the programm SIOS..."

⬭: semantic kernel, instr.: instrument relation, act: actor relation, fin: finality relation, obj: object relation, loc: location relation, attr: attribute relation,

⬚: event-type

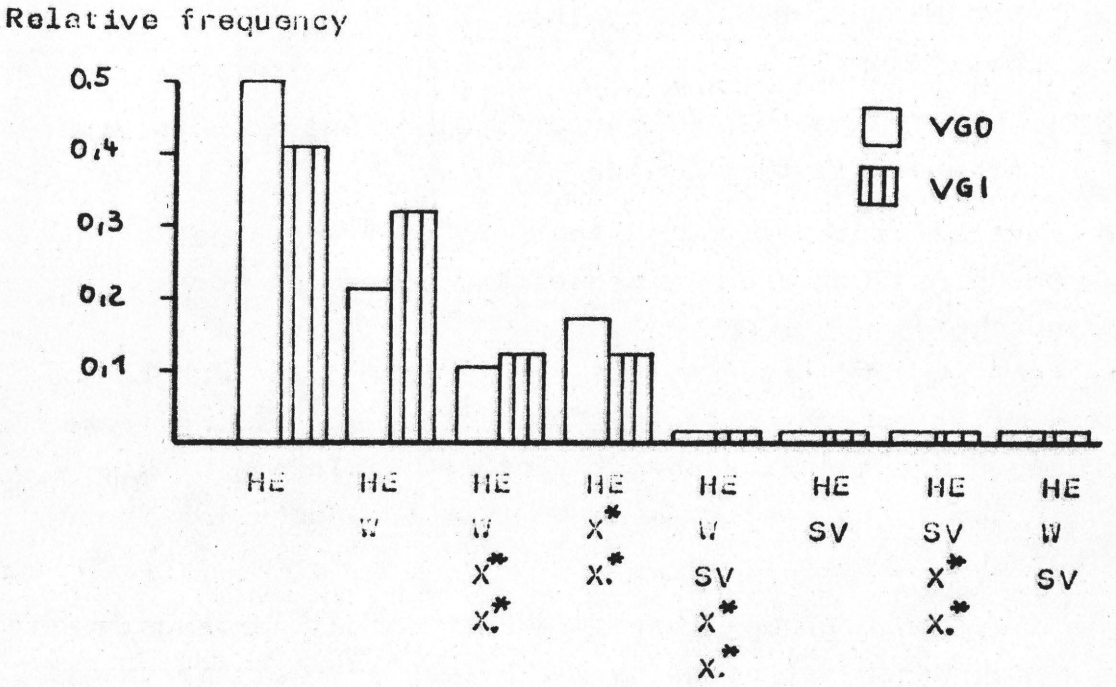


Figure 2: Types of interaction units

(HE = action units, i.e. keystroke sequences on the keyboard
W = Word markers to assign the actual goal of the user
X. = Combinations of alpha-numeric features displayed on screen after realizing a HE
X* = Word markers for X.
SV = computer processes)

Rule 1

IF (X: On screen is displayed a word and behind it there are 6 (or smaller than 6) marked positions and cursor is located at the first of the marked position

AND W: Your goal is to input a string to fill up all positions)

THEN (HE: You have to press particular numeric keys and after-wards the key FERT)

The actual state of the system were instructed not only verbally (X) but also by presenting schematic screen layouts (X.).
To test the understanding of this 8 rules the subjects were given concrete screen layouts and sub-editing tasks and they had to write down the order of classes of keys which had to be pressed down by them to accomplish the sub-tasks correctly.

The other both groups learnt task-oriented knowledge, that is, they were instructed by the leader of the experiment how the first editing task (ET 1) is to be accomplish correctly. One of these task-oriented group (VGT_1) learnt task accomplishment by instructing the goal (W), the system process (SV), the action unit (HE), and the alpha-numeric features (X.) displayed on screen after realizing the action unit of every sub-task that the editing-task consists of. The second task-oriented group (VGT_2) were only instructed the action unit (HE), system processes (SV), and the resulting alpha-numeric features (X.) displayed on screen.

Afterwards, the experimental procedure was the same for all 5 groups of subjects: All groups had to learn the accomplishment of the other 3 editing tasks by question asking until they met a particular learning criterion. So, the number of repetitions of the same editing task depended on progress of individual learning.
After finishing this learning procedure each group had to

solve a post test which consisted of particular sub-tasks of the learnt editing tasks.

To asses efficiency of learning the number of sub-tasks to meet the learning criterion were counted (see Fig. 3).

There are no significant differences in learning efficiency between the groups VGI, VGR, and VGT_1 but only between VGO and VGI, and VGT_2 and VGI. This is only true with respect to ET_2.

An other appropriate criterion to asses the influence of preknowledge on self-guided learning is the difficulty of learning task (see Fig. 4).

It can be seen in Fig. 4 that the second learning task (ET_2) is more difficult for the subjects of the group without any relevant pre-experimental, knowledge (VGO) than for the other groups. Furthermore, this task is also more difficult for the group VGT_2 than for VGI.

After finishing the learning procedure all groups had to solve a post test. The results are shown in Fig. 5.

It can be seen in Fig. 5 that the performance of the groups VGO and VGT_2 in accomplishing the post test is significantly lower than the performance of the students of informatics (VGI).

From the results obtained in the second step of investigations we can draw the following conclusions:

1. The efficiency and difficulty of self-guided learning depend on the pre-experimental knowledge that the learner already has about the learning domain. That means, self-guided user's learning can be make more effective if the user at first has to acquire a particular knowledge about the domain.
2. It makes no difference if this knowledge is task-oriented or if it consists of generalized rules for task accomplishment.

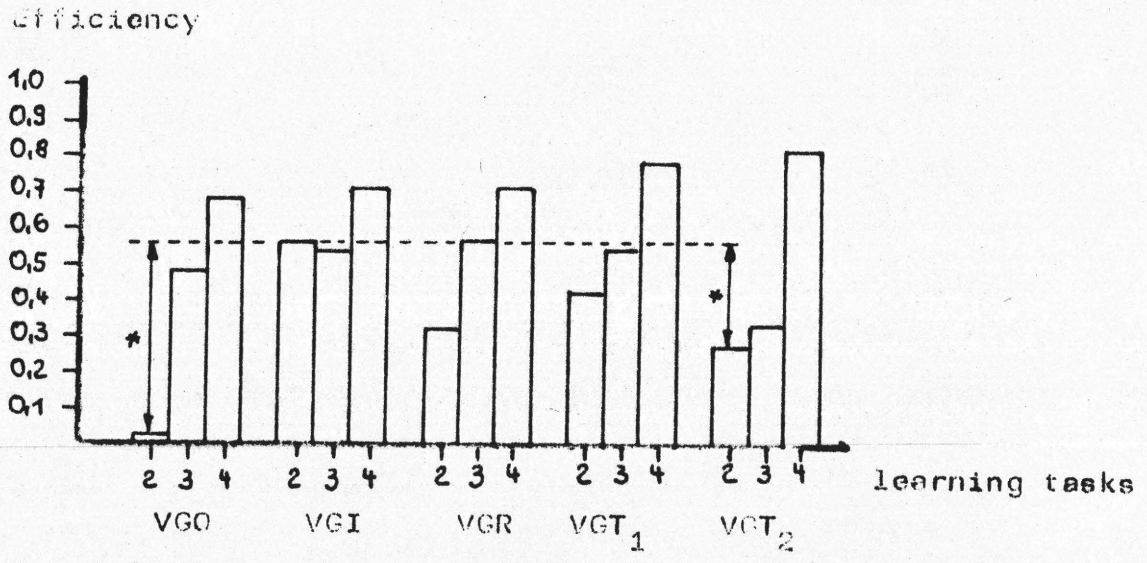


Figure 3:Efficiency of self-guided learning and preknowledge

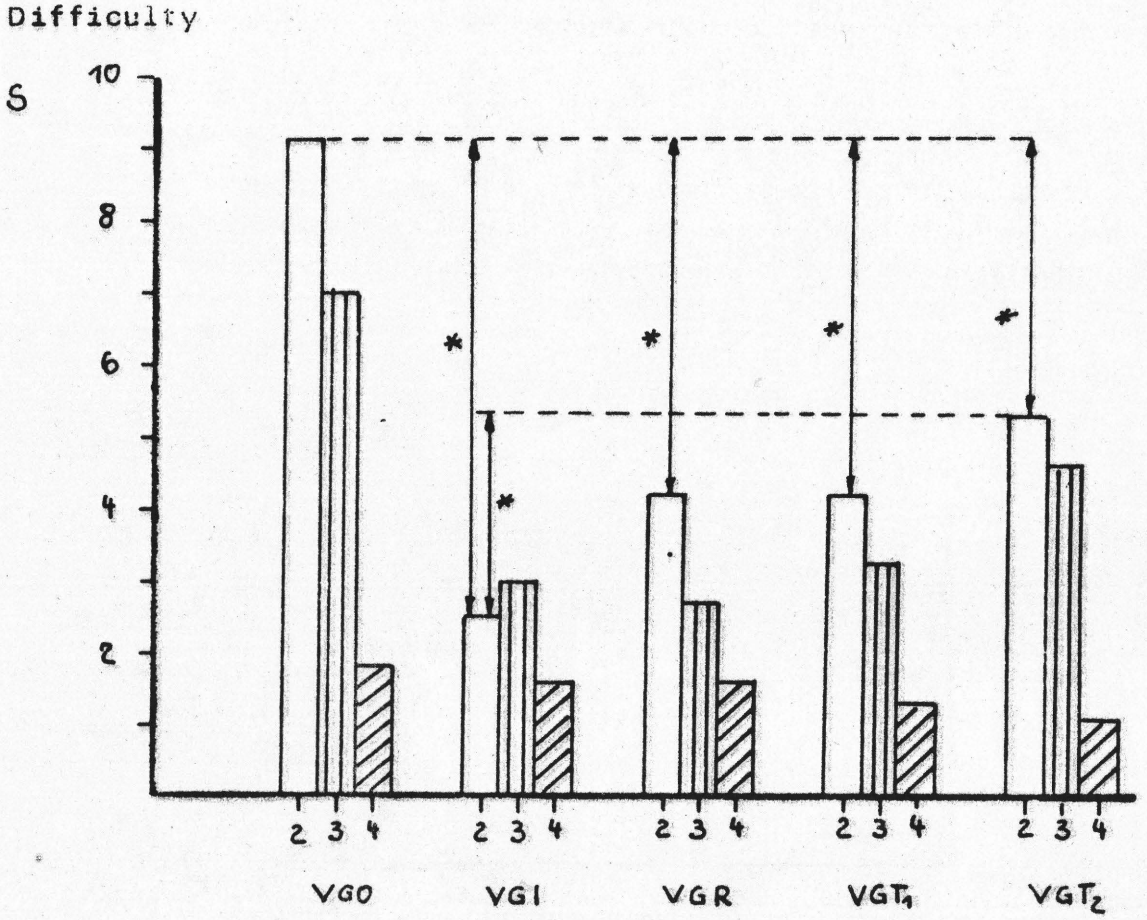


Figure 4: Difficulty of self-guided learning and preknowledge (Difficulty (S) were counted by

$$S = \sum_{i} i \times h(ST^{-})$$

i = number of repititions a learning taske to meet the learning criterion

$h(ST^{-})$ = number of unsolved sub-taske)

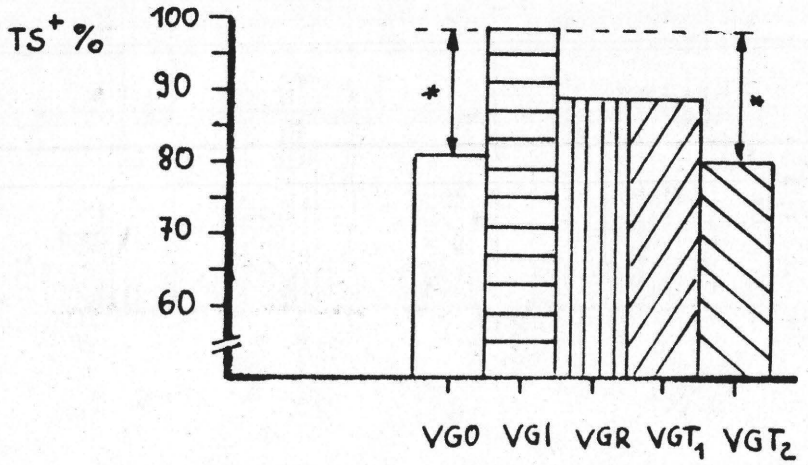


Figure 5: Performance in accomplishing the post test

3. We can repeat the conclusion which we had drawn from the results of the first step of investigation: system processes are of little importance for user's learning.

5. REFERENCES

A. L. Brown: Metakognition, Handlungskontrolle, Selbststeuerung und andere noch geheimnisvollere Mechanismen, in: F. E. Weinert, R. H. Kluwe: Metakognition, Motivation und Lernen, Stuttgart, Berlin, Mainz, Köln, 1984

Card, S. K., Moran, T.P. & A. Newell: The psychology of human-computer interaction. Lawrence Erlbaum Accociates, Hillsdale, New Yersey, 1983

Flammer, A.: Towards a Theorie of Question Asking. Psychol. Res., 1981, 43, 407-420

Jenkins, J. J.: Four points to remember: A tetrahedal model and memory experiments, in: L. S. Cermak & F. I. N. Crack (eds); Levels of processing in human memory.
Hillsdale, H. J.: Erlbaum, 1979, 429-446

Klix, F.: Über die Nachbildung von Denkanforderungen, die Wahrnehmungseigenschaften, Gedächtnisstruktur und Entscheidungsoperationen einschließen.
Z. Psychol. 193(1985), 3, 175-211

Moran, T. P.: The Command language Grammar: a represention for the user interface of interactive computer systems, Int. J. Man-Machine Studies (1981), 15, 3-50

Schindler, R. & F. Fischer: Effectiveness of training as a function of the teached knowledge structure. in: Klix, F. & H. Wandke (eds): MACINTER I, Elsevier science Publishers B. V., North-Holland, 1986, 151-159

Streitz, N. A.: Kognitionspsychologische Aspekte der Gestaltung von Dialogstrukturen bei interaktiven Lehr-Lern-Systemen, in: Mandl, H. & P. M. Fischer (eds): Lernen im Dialog mit dem Computer, Urban & Schwarzenberg, München u. a., 1985, 54-67

Tauber, M.: An Approach to Metakognition in Human-Computer Interaction, in: Klix, F. u. H. Wandke (eds): MACINTER I, Elsevier science Publishers B.V., North-Holland, 1986

LE CODAGE DES SIGNIFICATIONS

Session B2

THE ENCODING OF SEMANTICS

ON THE ROLE OF MEMORY IN PERSONAL INFORMATION MANAGEMENT

Mark Lansdale

University of Loughborough

Abstract

A requirement of 'The Office of the Future' is that it provides us with an effective way of managing (ie storing and retrieving) our information. But existing IT products go nowhere near supporting the variety of activities which can be observed in paper-based offices, and it is not surprising that concepts of the 'paperless office' are as far off as they were when the idea was first mooted. This paper argues that many of the issues involved in the automation of information management are essentially psychological in nature. These principally devolve upon the processes of recall, recognition and categorisation. Examples of exisiting information management techniques show how there is a trend to automate with a view to simulating office practices, or according to technologically-driven solutions. Both of these are inefficient with respect to the users psychological needs. A framework for developing user-oriented information management systems is discussed and some relevant research presented.

Introduction

We all keep information in our work. It may be books, notes, folders, diaries, or whatever. This is personal information not only in the sense that it is private, but that we have it for our own use. This paper is interested in the process of managing this information: the methods and procedures by which we handle, categorize and retrieve information on a day-to-day basis. This represents an area in which psychological theory can contribute to the satisfaction of a clear technological need. We are witnessing in the emergence of Information Technology a facility in which there is a pressure to create, store and process more and more information. But the purpose of IT should be to increase the quality, not merely the quantity, of available information. If all it achieves is to increase the volume of our filed information without an associated improvement in retrievability, then the reverse must be expected. The psychological aspect of this is that the management of personal information, by definition, involves psychological processes: The information to be retrieved has already been handled, categorised and filed away by the individual. It is therefore reasonable to ask how much the problems of information management represent a shortfall in these processes and the ability to remember what was done. Consider some examples:

'My boss wants to see all the project reviews I have carried out over the last six months. The trouble is, they are filed under each of the individual projects. It will take me ages to work through and dig them all out.'

'I know what the thing looked like: it had a blue and white stripe at the top, but I can't remember how I filed it, or even what it was about.'

'Yes I remember that paper. It came at the same time as the product audit. I can't remember what happened to it though'.

The document I want is the French Finance Committee's minutes, but I've

tried looking under "French", "Finance", and "Committees", and it's not there. Perhaps its under "France". I may as well search through the lot.'

I would not like to suggest that all the problems of information management are psychological in nature. But these examples illustrate two important issues in the development of information management tools on computers. First is that there is a general problem in categorising items,and second is that we remember far more about documents than can be used in retrieval procedures. Clearly if the first of these could be ameliorated, and the second exploited, powerful tools will emerge.

Studies of the 'Natural history' of offices

One approach to the psychology of information management is to study the natural history of offices (eg Gole 1982, Malone 1983). The notion is that by studying the behaviour of people in natural situations we may come to understand the underlying psychological principles. The small study by Malone is a good example of this. Ten office workers were interviewed about their jobs, their information management habits and their ability to access target documents. The disposition of their offices in terms of layout, distribution of papers, filing methods, etc, were also examined.

A principal distinction that Malone draws is between 'neat' and 'messy' offices. We should not be surprised to find out that occupants of neat offices reported fewer difficulties in information retrieval, overlooked fewer things they had to do, and were better able to find specified target documents on request. Clearly some of this is personal style. Some people are tidier than others. However, this may not be the whole story. Malone's analysis suggests that the 'mess' in offices is not arbitrary, but reflects a response on the part of the individual to particular needs or difficulties. Three in particular are identified. First, documents are left lying around in conspicuous places as reminders that something has to be done with them. Second, some tasks need information from several sources, and therefore require that several documents are kept close and 'active' while the task is being carried out. Third, Malone suggests that on many occasions the person is reluctant to file information away either because they cannot decide how to categorise it, or because they are not confident in their ability to retrieve it later. As a result, the compensating strategy is to pile documents around the office in relatively unstructured files.

If one accepts this view, people do not slip into messy offices purely as a matter of personal slovenliness. There is a mismatch between what the person needs to do and the facility to do it. All these things create the pressure to spread documents around the office in more or less unstructured piles. The 'technology' (in this case paper-based) does not adequately support these needs, and ultimately the strategy of using piles of information is counter-productive.

Problems in the interpretation of office behaviour

There are interesting things we can learn from observing natural information management behaviour. No doubt more detailed systematic studies will reveal more. However there is also a note of caution required. It might seem straightforward to suppose that we can translate observed strategies of information handling from existing paper-based methods to computers. This is, as we shall see later, the basic philosophy of many leading producers of office automation

equipment such as Xerox (Smith et al 1982). In principal however, this must be a mistake. Consider the use of piles identified in Malone's study. Noone would suggest the introduction of unstructured 'piles' of documents in a computer environment. Apart from anything else, they are evidently counterproductive. How then do we decide which aspects of office behaiour to emulate in office automation and which to avoid?

The problem is that the strategies used by people in one technology need not apply to another. The point is subtle but critical to a psychological analysis of information management. The piles that Malone reports are not, in a simple sense, representative of a NEED in the user. Quite the reverse, in fact. They are a compensating strategy for the problems of classification (eg see Dumais and Landauer 1983). In using piles, the worker is making a tradeoff along several dimensions of difficulty. To avoid the process of classification he puts objects in a particular place. With this he forgoes the opportunity to retrieve the document by any simple classification-based search. But he has other things up his sleeve. First he can remember what it looked like. Consequently a strategy of scanning the piles (they are of course open to inspection) may identify the target by visual recognition. Second, he may remember where it was left, thereby reducing the area for search. Third, the use of piles has an implicit element of ordering by time: the most recent documents are near the top of the piles. This may also be useful information.

Concentrating upon piles is to caricature what happens when procedures from office practices are reapplied to computers, but it illustrates the point well. Merely to invoke the concept of PILES in such systems without an understanding of the role played by recognition memory, or memory for the spatial and temporal attributes of documents, may be to entirely to miss the point: the piles may be the visible manifestation of an information handling strategy, but the psychology that underlies them may be much less direct. Only an understanding of this psychology can lead to genuine advances in information retrieval systems.

A Framework For Future Research

The framework I propose for the process of information retrieval is very simple: it is that EVERY attempt at retrieving information involves two distinct psychological processes: recall-directed search, followed by recognition-based scanning. Recall-directed search refers to the use of memory about the required item to get as close to the document as possible. At its most exact, this will give direct access, such as remembering a filename or in which sleeve of a filing cabinet drawer it is located. As it becomes less precise, memory might identify an area of the database: a drawer of a filing cabinet, a shelf, or perhaps a particular computer directory. In this case, more or less recognition-based scanning within that area is required depending upon the exactitude of recall. Recognition-based scanning is therefore the process we undertake when recall has failed to produce a unique item, and the amount we have to undertake depends upon how specific the recall was.

I propose that the observed behaviour with any information storage and retrieval system can be seen as a tradeoff between these two processes of recall and recognition. The users can be said to be balancing the two processes to maintain, in their eyes, reasonable utility. In other words, the strategies people adopt can be seen as a way of shifting this tradeoff to their advantage. For example, by leaving documents around an office in piles, people are using their recall for where it was

placed, when it was placed there, and what it looked like. The scanning process can therefore be reasonably efficient, at least until the office becomes clogged with papers, because recall can not only specify where to look, but also visual attributes which help in recognising the target.

This is also a way of looking at the usability of exisiting systems. For example, it can be seen why traditional operating systems running on the basis of using single filenames fail to provide an adequate solution to the storage and retrieval needs of the electronic office. The recall process is deficient because the user's ability to categorise documents with the appropriate filenames, and their ability to remember those filenames, is limited. On the other hand, the user cannot fall back upon scanning methods: it is enormously clumsy to search directories and open files one at a time to inspect their contents. Having done so, they can only be differentiated by reading them. The user simply has nowhere to go, no way of using the machine which will provide an acceptable tradeoff between the problems of recall and recognition. On the other end of the scale, the Spatial Data Management System, (Bolt 1979) relies exclusively upon recognition-scanning stategies, and although unevaluated, it is hard to see how this could provide realistic support for the management of real information systems.

Thinking in terms of this tradeoff also suggests a general method of approaching the design of an information management system. A future system should be as near to an optimization of the two processes as possible: recall processes should allow the users to use whatever memory they have to limit the area of the database to be searched; and then the information within this area of the database should be represented in such a way as to maximally assist the search process. Optimizing both processes (as far as is possible) should give any user enough room to manouvre such that they can accomodate their special needs and idiosyncracies. This leads me to a number of specific research issues which colleagues and I (see note 1) have been concerned with in the laboratory.

Experiment 1 : What attributes should electronic documents have?

Electronic systems introduce new degrees of freedom in the possibilities of information management. We are no longer constrained to see documents as strings of text with a name, and we are entitled to ask how all sorts of additional attributes such as icons, colours or methods such as those supported by SDMS can be used to create a more rich and varied environment which exploits aspects of human memory. This is the philosophy of 'cue-enrichment' (eg see Cole 1982). Almost no research has been carried out upon the memorabiity of different methods of coding documents, or which forms of coding should be employed. For this reason, an exploratory experiment was devised in which subjects were required code documents by associating a shape to them whose colour and location on the document were also at the users discretion. The coding method is therefore one of creating a coloured 'watermark' on the document which incorporated three dimensions. This was chosen because strong claims are being made for the memorability of icons (eg Lodding 1982), and because they provide a natural way of combining the visual attributes of shape, colour and location. Each of these are of mnemonic interest in their own right.

The users were provided with a simulated system in which they were asked to imagine they were working in an employment agency. They were then shown a sequence of 18 job adverts which they were required to file in

the following way. First, a selection of the required shape was made from a touch-sensitive array of 12 shapes, chosen such that six of them had some meaningful relationship with the adverts being presented. Following this, a selection of colour from one of twelve was made in much the same way, and finally, the user pointed to one of twelve locations on the advert to place the coloured icon. After a delay, memory was tested by re-presenting the users with the adverts (in a different sequence) and asking them to reconstruct the original icon in the same way as before. For comparison purposes, a second group of subjects were asked to carry out the same task using a verbally-based system in which they assigned three names, a category, colour name, and adjective, to the adverts. These were also chosen from touch sensitive menus such that the only differences between the iconic and verbal methods were the mode the file-designators were coded in, iconic or verbal.

The key results of this experiment were as follows. First, no simple picture of the benefits of iconic as opposed to verbal methods emerged. The users appear to perform at roughly the same level for both systems. More important to recall appears to be the degree of semantic link between the attributes associated and the advert. Thus the icon shapes and the category words are recalled well, colour and colour names less well, while adjectives and locations were recalled relatively poorly. This latter result confirms a study by Dumais and Jones (1985), that memory for arbitrary locations is not particularly effective. Further analysis of the coding strategies of the users implies that those attributes which were less well remembered were harder to code when assigning them to adverts. In any case, it is clear that iconic methods were not the panacea to recall that might have been expected (eg see Lodding 1983).

A second important result of this experiment was that the recall of the coding dimensions were statistically independent: the likelihood of recalling colour, for example, was not contingent upon whether the user had already recalled shape or location. The characteristic of an independent model of recall is that users memory for particular icons or word combinations will be predominately partial. It is unlikely that nothing will be recalled at all or that all of the combination will be recalled. This suggests that an information management tool which allowed users to employ their partial knowledge will be of considerable utility. Lansdale (1985) developed such a tool which increased the accuracy of access by an order of magnitude.

Experiment 2 : How easy should filing be?

One aspect of information retrieval which is important to emphasise is that although difficulties may be apparent at the retreival stage, in that one might not be able to find something, it is as sensible to see this as a problem of storage as of retrieval. We cannot find information because we did not file it in a form which is appropriate for future retrieval, or perhaps we didn't pay enough attention to it. It is a natural reaction for people not to spend a great deal of time filing information because it has no immediate pay-back and because they want to get on with the next piece of work.

A second experiment shows that users assigning retreival tags to documents (pictures and words) have much stronger memory for those tags if they assigned them themselves as opposed to having them assigned by the system. This was demonstrated by taking the experiment above and presenting subjects the adverts with the icons or file words already

assigned by a different subject. Thus the only difference between subjects in this experiment and the previous one is that they have no control over the assignation of the coding. This result ties in well with well established educational principles that memory is much more robust for self-generated material than when it has been provided for the subject (eg Bower 1970).

This leaves us with a dilemma. The more we ask the user to do at the process of information storage, the less likely he is to do it, creating retrieval problems. On the other hand, the more we automate the process of storage and take responsibility away from the user, the less he is going to remember, and therefore the less he is going to be able to retrieve. This leads to two classes of research issues which follow. First, we can ask how we might facilitate necessary tasks such that they are not so onerous that users will not carry them out. This is a matter of finding out what it is we want the user to do at the stage of storage and considering how the software could be devised to do this most easily. Alternatively, we can resign ourselves to automated systems in which the user's memory is intrinsically poorer. In this case we are interested to ask questions as to what, if anything, is best remembered under these conditions and how we can best exploit what is remembered. The key point here is that any future research must be sensitive to the tradeoff between the positive aspects of easy filing facilities and the negative effects they have upon the user's memory.

Note 1. These experiments were carried out with T.Stroud and M.Simpson as final year projects under the supervision of the author.

REFERENCES

Bolt, R.A. (1979). Spatial Data management. Boston: MIT Press. Library of Congress number 78-78256.

Bower, G.H. (1970). Analysis of a mnemonic device. American Scientist 58:496-510.

Cole, I. (1982). Human aspects of office filing: implications for the electric office. Proceedings of the Human Factors Society, 26th Annual Meeting, Seattle USA.

Dumais, S.T. & Landauer, T.K. (1983). Using examples to describe categories. Proceedings of CHI '83, Boston, USA.

Dumais, S.T. & Jones, W.P. (1985). A comparison of symboloic and spatial filing. Proceedings of CHI '85, San Fransisco, USA.

Lansdale, M.W. (1985). Psychological issues in the use of icons in information retrieval. Manuscript prepared for publication.

Lansdale, M.W. (1985). Strategy and Structure in Cognition. Paper presented to ESRC International Workshop on Modeling Cognition, Lancaster, UK.

Lodding, K.N. (1983). Iconic Interfacing. Computer graphics and applications.

Malone, T.W. (1983). How do people organise their desks?

Implications for the design of office information systems. ACM Transactions on Office Information Systems, 1, (1), 99-112.

Simpson, M. (1986). Unpublished final year project, Department of Human Sciences, Loughborough University, UK.

Smith, D.C., Irby, C., Kimball, R., & Verplank, W. (1982). Designing the Star User Interface. BYTE, April 1982 pp 242-282.

Spence, R. & Apperley, M. (1982). Data Base Navigation: an office environment for the professional. Behaviour and Information Technology (1), 1, pp 43-54.

Stroud, T.R.J. (1986) Unpublished final year project, Department of Human Sciences, Loughborough University, UK.

Conceptual primitives and their representation in understanding complex operations

Gabriele Rohr

Science Center Heidelberg, IBM Germany

abstract

According to the conceptual basic categoties Jackendoff (1983) found out by language analysis to be the frame of knowledge about the environment, we specified operations of a text editing and file handling system developed by us for experimental purposes. It was aimed to analyse understanding and use of these operations in complex tasks dependent on the kind of representation (iconic or verbal). Analysed were the use of those operations with the most extreme conceptual primitives, i.e. being either best described by spatial functions (objects from place to place) or by existential functions (existential event or object, e.g. create, delete etc.) in simple and complex editing tasks with respect to visual spatial (iconic) and verbal representations. Furthermore, a questionaire was applied asking for the primitive elements within each of the basic categories of functions (spatial functions, property functions, posession functions, existential functions) associated with each command either expressed as a visual spatial symbol or as command words.

65 unexperienced users took part in the experiment, a pretest maintest posttest design with two control groups. Some of the results are: Visual representation of commands helps building up spatial concepts. A spatial concept for an operation being best described as an existential function create interferences in handling this operation (more time is needed) and vise versa. Complex editing tasks are best performed having a spatial concept of the operations needed to perform them. In the questionaire command words are better described as existential and property functions and icons (visual spatial symbols) as spatial and property functions. The results are discussed also with respect to memory and coding strategies.

L'HOMME COMME MODELE POUR LA MACHINE

Session B3

HUMANS AS MODELS FOR COMPUTERS

Search strategies in internal and external memories

J.J. Beishuizen & F.M.T. Brazier

Vrije Universiteit

Department of Cognitive Psychology

Amsterdam

Should the structure of information retrieval systems correspond to the structure of long term memory? This question is relevant to human factor specialists engaged in the design of user-friendly interfaces for database systems. The question presupposes agreement on the assumption that dealing with external information sources resembles internal mental activities. Although some a priori objections can be made against this assumption, we will follow this line of argument in an attempt to link some research findings with respect to information retrieval in long term memory to research and development related to consulting external databases. We will furthermore consider a numer of aspects concerning the design of a user interface with coaching facilities.

Norman & Bobrow (1969) proposed a model for a kernel retrieval cycle in long term memory. This cycle consists of three stages : (a) specification of the target description, (b) selection of memory elements, (c) evaluation of the retrieved information. An iteration of retrieval cycles leads to the gradual refinement of the target description until the information to be retrieved is found. This kernel retrieval cycle has been a cornerstone in several models of long term memory. Kolodner's (1983) CYRUS system, an intelligent fact retrieval system, contains a model for the storage of events in long term memory and a search algorithm based on Norman and Bobrow's kernel retrieval cycle. Two important assumptions have been made in CYRUS: (1) that memory elements are not enumerable, and (2) that the succession of the retrieval cycles is recursive. These characteristics are responsible for the fact that

CYRUS has been based on a hierarchical tree structure in which the memory elements have been placed.

The claim that the accessibility of databases depends on the degree to which the interface reflects hierarchical memory models can be criticized on formal and empirical grounds. We conducted several experiments in which 14/15 year old students searched in a database with approx. 200 enumerable items. The usual instruments (operations for creating subsets based on a keyword, operations for creating intersections and unions, etc) were available. The students were provided with a target description and a list of keywords and had to find information relevant to the target description. The students used two retrieval strategies: a depth-first strategy and a breadth-first strategy. The depth-first strategy is characterized by the successive reduction of the search space using a few keywords at a time, as needed. The breadth-strategy is characterized by the fact that an attempt has been made to construct a complete representation of the target description by entering all of the necessary keywords before reducing the search space by intersections of subsets. Both strategies turned out to be equally effective. One of the conclusions concerning the design of database interfaces is that a rich keyword structure is necessary. However, a hierarchical organization of keywords is not advocated. Unfamiliarity with the implemented hierarchical structure implemented in the interface can have fatal consequences - the database becomes practically inaccessible. The interface should allow users to choose their own entry point into the system, allowing him/her to proceed towards the goal along a diverse number of routes. There should be room for different search strategies. The coaching component of an information retrieval system should be flexibile enough to allow the user to follow different paths, providing sup-

port when needed. Our experience has shown that the advice the coach gives should be based not only on its resemblance to an expert's strategy, but also on the contents of the advice previously given, and on the assumption that beginners do not always have to follow an expert strategy. The complexity and broadness of a domain may make the implementation of an interface based on a hierarchy inevitable. The inclusion of a facility which makes it possible to look ahead (to preview the consequences of the choice before it has been made) increases the value and usability of such systems. For all sytems holds that the inclusion of an option which makes it possible to return to a previous decision (an undo facility for backtracking) increases the value of the interaction.

Man-man communication: a model for computer-human communication ?

Pierre Falzon

I.N.R.I.A.
Rocquencourt - BP 105
78150 - Le Chesnay, France

The validity of the man-man model

The title of this communication can be interpreted in two different ways, either as "Is the man-man communication model a relevant model for computer-human interaction?" or as "What lessons for computer-human communication can we learn from studies of man-man communication?"

Both questions are worth considering. However, there has been very few correct attempts to provide answers concerning the relevance of the man-man model.

Richards & Underwood (1984) have addressed the question in a study of spoken communication with computers. The results indicate some modifications in the use of language as a function of the supposed nature (man or machine) of the dialogue partner: when believing to speak to a machine, subjects tended to restrict their vocabularies, to acknowledge good reception of the messages less frequently, and to seldom use indirect questions. These results are debatable since, in the pseudo-machine condition, the voice heard by the subjects was distorted by use of a vocoder. This may have influenced the subject's utterances.

Onorato & Schvaneveldt (1986) have studied the task of specifying procedures to different partners, among which a human partner and a computer. They conclude that the relevance of the man-man model is not independent of the experience of the subjects: users differing in experience with computers have different expectations about what the computer is capable of doing. Totally naive users tend to behave with computers as they do with human beings. Beginners and experts behave differently.

This communication will be focused on the second question: lessons to be learnt from studies in man-man communication. Several works in that field will be presented, with emphasis on the implications of their results for the design of interactive systems.

The expression "computer-human communication" (instead of the more usual "human-computer communication") has been used purposedly, to indicate specific emphasis on the use of the man-man communication model to adapt the behavior of the computer. The question is not "How do people talk to computers", or "What happens (for the humans) when they talk to computers?", but rather "What characteristics of man-man interaction could we implement in computers?" Relevant problems are then: How should the computer understand? How should it process users' inputs? How should it react? How should it adapt its strategies in comprehension and production?

The language and the mode of interaction

Is natural language natural ?

I will begin with a contribution to a long lasting debate, concerning the language that should be used in human-computer interaction. Two positions have been held, the first advocating natural language, the second in favor of restricted command languages.

Several experimental results cast doubts on the use of natural language in man-computer dialogues:

- given the choice between natural language and a restricted language, users often choose the latter (Gould et al, 1976; Hauptmann & Green, 1983);
- even in the absence of all constraints, subjects tend to restrict spontaneously the language they use (Scapin, 1985);
- natural language does not allow a better performance than a well-designed restricted language (Borenstein, 1986; Hendler & Michaelis, 1983; Bailey, 1985).

However, two objections can be raised.

First, most of these studies have concerned typed interaction, considering that the state of the art in speech comprehension forbade the possibility for oral communication with machines (with the exception of the study by Richards & Underwood presented above). One can argue that different results would have been found, had oral communication been considered. As a matter of fact, the preference for restricted languages, observed in typed interactions, has its origins, at least in part, in the constraints of keyboard inputs. This effect is of course stronger if subjects are not experienced typists (on this point, cf the studies of Chapanis & coll, summarized in Chapanis, 1978).

Second, it appears that humans have a natural tendency to restrict their expression, not only when typing instructions to a machine, but also when interacting with other humans. This tendency, described by many authors (e.g., concerning the process of referring, the studies by Krauss & Glucksberg, 1977, Krauss & Weinheimer, 1964, and Clark & Wilkes-Gibbs, 1986) may even lead, especially in work contexts, to the elaboration of specialized sublanguages, *operative languages*, restricted and distorted as compared to natural language (Falzon, 1983, 1984, 1986). Very briefly, operative languages have the following characteristics:

- their vocabularies are limited in size, and are neither a representative sample nor a subset of the total vocabulary of the language: they include words which are rare or unknown in the language as a whole;

- their syntax is limited (their grammar have a limited set of rules), and specific in two ways: some of the syntactic rules of operative languages do not exist in the general grammar of the language, each operative language has its own grammar;

- the words of operative languages tend to be monosemous. The meanings of the words may be rare or unknown in the language as a whole. The same word, if used in several operative languages, will have a specific meaning in each operative language. Moreover, semantic restrictions tend to limit potential ambiguities;

- pragmatic interpretations are limited by restrictions on the set of possible goals of both dialogue partners, and by a tighter application of conversational rules.

The fact that humans elaborate operative languages does not mean that designing natural language understanding systems is useless, but certainly indicates that the decision to use natural language in computer interaction is not an obvious one, and that restricted command languages are not as unnatural a solution as some claim it to be.

Interaction modes

Here again, we face a well-known question, namely: What is the optimal mode for man-machine communication: menu vs form fill-in vs command language vs natural language, etc.

Experimental results tend to indicate that the choice of a mode depends upon the users' experience: the menu is better suited to novices, the command language is better adapted to the experienced operators (or to those who are to become experienced operators). In other words, experiments seem to demonstrate that, for dialogues with non-experts in the domain, machine guidance is more efficient than user guidance (Kupka & Wisling, 1980; Benbasat et al, 1981; Ogden & Boyle, 1982; Hauptmann & Green, 1983; Ramsey & Grimes, 1983; Potosnak, 1984).

Studies of human dialogues certainly indicate variations of the interactions according to the level of knowledge of the dialogue partner (e.g. Krauss & Glucksberg, 1977). However, I will here emphasize another factor of variation: the goals of the speaker. As will be seen, this is not independent of the choice of an interaction mode.

Two dialogue situations can be differentiated: those involving two experts in a domain and those involving an expert and someone seeking some advice, information, etc. These two types of situations call for different dialogue strategies.

Expert-information seeker dialogues

Conversations between doctors and patients are examples of expert-information seeker dialogues. In these situations, doctors can be considered not only as specialists of a domain (medicine), but also as specialists in a specific situation of communication, the medical interview.

These conversations have been studied by Evans (1976). The goal of this author was the automatization of some medical interviews and the conception of an expert system for medical diagnosis which could be consulted by the patients themselves (and not only by the specialists, as it is the case generally). In this perspective, Evans analyzes the strategies used by the doctors during the interview, and the patients' answers: the complexity of these answers determines the complexity of the understanding system to be designed. Evans' conclusions are:

- although physicians experience much trouble in trying to describe the procedures they use in their interviews, they do use a dialogue strategy, that can be formalized by rather simple flow-charts;

- the patients' answers use a very restricted vocabulary, and are often limited to 'yes', 'no', 'don't know' answers. Even when answers are longer, physicians always code them as simple statements. When answers get verbose or inconsequential, physicians ask for a 'yes or no' answer.

These results are interesting in terms of system design. Evans even got to the point of designing a keyboard with three keys: 'yes', 'no' and '?' (evaluations of the system are presented in Bevan et al, 1981, and Card and Lucas, 1981). These results are also interesting in terms of the physicians' cognitive processes.

For the physicians, comprehension means getting answers that allow to organize the patient's symptoms, and to classify his/her illness in a known category. The physician is thus looking for answers that allow to make choices between alternatives. In this purpose, they take control of the dialogue as soon as possible, and seek one of the three types of answers described above, even if, in order to get this answer, they have to translate the patient's utterance or to force one of these answers when translation is impossible.

The results presented ealier concerning man-machine dialogue situations seem thus in agreement with Evans' observations on medical interviews: guidance by the expert (physician,

or machine) is recommended when the dialogue partner is a non-expert (patient, or domain-naive subject).

Expert-expert dialogues

Nevertheless, it must be stressed that Evans' results are interesting only in as much as we deal with the consultation of an expert system by a domain-naive subjects. Supposing that experts (e.g. physicians) consult the same system, they probably would not be satisfied by it. When human experts consult an expert system, their goals differ from the goals of a patient. Human experts want to check their diagnosis. Of course they are interested in the *result* proposed by the system, but, more important, they are interested in the *reasoning process* that yielded this result.

The results given by Clancey (1983) are consistent with these remarks. Clancey has worked on the expert system MYCIN, with the goal of extending the system's explanation capabilities, in order to make it usable for teaching purposes. In this perspective, although the prospective users (the students) are not experts in the domain, their goals are very similar to those of the experts. For example, they also are interested as much in the processing logic and in the justification of the rules as in the diagnosis itself.

The analysis of MYCIN by the author is thus focused on the rules and their justifications. Clancey describes different shortcomings in the existing explanation system, caused mainly by the lack of hierarchy among the rules and by the absence of some levels of explanation.

Consequently, the dialogue system necessary for the consultation of an expert system by an expert will differ from the dialogue system necessary for the consultation of that same system by a naive subject. The expert-expert dialogue system must be able to give explanations about its own functioning, about the rules underlying its choices, about the reasoning steps, etc.

Communication competence and processing strategies

Communication competence

When communicating, humans manage to lower the level of complexity of cognitive processing, by using different strategies. These paragraphs will consider different (human) ways of coping with complexity, and their applicability to human-computer interaction.

The following remarks deal mostly with expert- information seeker dialogues. In these dialogues, professionnal competence can be divided in two:

- domain competence, i.e. the technical knowledge of a profession;
- communication competence, i.e. expertise in dialogue procedures.

Roman Parré (1984), who has analyzed different situations of professionnal dialogues, stresses that, while technical competence is of course domain-dependent and shared with operators whose task does not include communication with non-experts, this is not the case for communication competence. This second level of competence is independent of the domain and is common to operators in charge of answering queries, whatever the technical domain.

For example, income tax advisers in charge of answering telephone calls have the same theoretical knowledge as their colleagues at the same level of the hierarchy, but their "know-how", e.g. the techniques they use during query answering are similar to those used by other "query answering" specialists studied by Roman Parré.

Communication competence is the result of a learning process. In a study of telephone operators of a medical centre (Falzon et al, 1986, reported below), we have been able to study two poperators equal in domain experience and unequal in experience in communication. The operator having less experience in communication proceeds differently than the more experienced operator.

Some of the operators studied by Roman Parré are aware that this competence has been acquired, and are able to spot some of the errors that experience has allowed to eliminate. For example, operators have doscovered that they waited for too long before speaking; they have learnt to interrupt the users' discourse with yes or no questions. Expert communicators take control of the dialogue as early as possible, in order to save time: experts know the relevant questions, those which are necessary in order to quickly classify the requests.

Processing strategies

Most present understanding systems are characterized by a single mode of functioning: they try to analyze the inputs by using *all the time all the available procedures and knowledge structures*. The observations below indicate another possible organization, in which systems would be able to use two different modes of functioning.

Simplifying by assuming a correct problem space

In most work situations, a single operator processes different kinds of problems, whatever the level of complexity. This makes it difficult to point out the variations in cognitive processing according to problem complexity. This analysis is easier when problems of different levels of complexity are allotted to different operators.

Such a situation is described by Roman Parré (1984), in a study of the task of specific telephone operators (those in charge of providing information on telephone numbers). Two categories of operators are involved, each category being allotted a specific task. Operators of the first category receive all calls, and are not allowed more than three attempts for each call. If these attempts fail, the call has to be transferred to an operator of the second category, in charge of 'special queries'.

The underlying assumption of this organization is that two classes of situations occur. First, situations where the request can be processed directly, assuming its formulation to be correct: the task consists only in translating the request into a query to a data base (the telephone directory). Second, situations where the call is more difficult, and where, before querying the data base, the operator has to analyze the request more fully.

"Standard" cases are then processed by the first category of operators, more unusual cases by the second category. Thus, each category of operators is allotted a specific mode of reasoning: if the routine mode fails, calls are transferred to the operators in charge of developing more elaborated search strategies.

Of course, this type of organization may have negative consequences on the interest of the task and on job satisfaction, but this is not the point here. The point is that it clearly shows the variability among the requests to be processed, and, as a consequence, the variability of the reasoning processes involved. Operators deal with that variability by attempting first to apply directly the search procedures; requests are processed more fully only in case of failure of the initial attempts.

Simplifying by hypothesizing a problem space

Another aspect of communication competence is described in a study of the verbal activity of the secretaries of a medical centre (Falzon et al, 1986). These secretaries are in charge (among other activities) of taking appointments for some of the specialists of the centre. The study was focused on the activity of two of the secretaries, equal in domain knowledge, but differing in experience as operators.

An interesting result of this study concerns the dialogue strategies. The experienced operator's dialogue strategy enables her to reach very quickly (in usual cases) an agreement on a date and time of appointment. The expert strategy has two main characteristics:

- firstly, it is based on a very early hypothesis on the nature of the request: this hypothesis is made on the grounds of a *model of the standard dialogue partner* (i.e. the usual client). This model postulates that the client want an appointment for himself/herself or for a close relative, that the appointment is wanted as soon as possible, and that the client is in a standard administrative situation as far as the Sécurité Sociale (Medical Welfare) is concerned;

- second, it uses *dialogue speed-up mechanisms*, allowing to avoid some converation steps. For example: if an hypothesis of "request of appointment with specialist X" has been made, do not ask for an explicit confirmation of the hypothesis, but propose a date and time of appointment (on the basis of the above model). In general, it is more effective to act as if the interpretation is correct: if it happens to be wrong, it will provoke a negative and focused reaction of the user; if it is correct, it will speed up the dialogue.

The interest (for computer understanding) of using a model of the standard dialogue partner has been stressed by Shapiro & Kwasny (1975), Leroy (1985) and Lubonski (1985). The use of early interpretations of the queries in order to speed-up the dialogue has been advocated by Hayes & Reddy (1983).

Although the expert strategy is very efficient in most cases, it may lead the operator along wrong paths in case of very unusual requests. For instance, if the client does not want an immediate date of appointment, the operator has to consider the date constraints of the user. The operator may experience some difficulty in some case to get rid of her erroneous interpretations. The recovery procedures she uses prove sometimes ineffective when the actual case is too far from the model.

Levels of processing

These two examples show an organization of the processing strategies which differs sharply from the functioning of present interactive systems. It indicates a possible structure for the understanding units of these systems.

At first, a superficial, economic mode would be attempted; this mode should be enough to process quickly the standard cases (i.e. the more frequent cases). By economic mode, we mean an analysis based on the use of knowledge structures strongly dependent on the domain, the task, and the prototypical situations. This mode could proceed by searching for key-words, sufficient to evoke schemata corresponding to the more usual problems, these schemata being then used to guide the comprehension process. Syntactic analysis, for example, can be simplified to a large extent

It is only in the case of failure of the economic mode that the system would switch to a more elaborated processing mode. This mode would use more complex programs of phonetic decoding (in the case of oral input), of syntactic and semantic analysis. This mode would use other knowledge structures, less stereotyped and less dependent on the domain.

Of course, one can suppose that this second mode will be less economical not only in terms of the complexity of its subprograms (and of the required processing power), but also in terms of the necessary response time. But this is true also of human processing: rare problems take longer to solve than easy, usual ones.

Within this framework, two hypotheses can still be considered. In a first one, the more elaborated mode would be triggered by a call for help from the economic mode. This solution (which corresponds to what we observe in the first situation described above) has notable advantages, particularly in terms of the low workload implied for the machine when the economic mode is sufficient (i.e. in a large number of cases).

In a second hypothesis, the more elaborated mode would operate permanently, off-line, observing the utterances of the user and the answers of the economic mode. This could allow not only for its intervention in case of failure of the economic mode, but also for a "meta-

dialogue" behaviour, thus avoiding some communication errors, caused for example by the use of erroneous schemata.

Hierarchical cognitive processing

One aspect of cognitive economy lies, as we have seen above, in the ability to use different modes of reasoning, as a function of the complexity of the problem. A second aspect of cognitive economy lies in the ability to choose an appropriate level to process the problem, in order to lower the cost of processing. The goal of this choice is to maximize the number of situations where operative or routine knowledge structures can be used.

An example of this second aspect is presented in a study of the functioning of an administrative information center (Roman Parré, 1984). In this center, which can be consulted by telephone, two categories of operators are involved, generalists and specialists.

Calls are first processed by generalist operators, in charge of answering the more usual queries (i.e. to process the routine requests), and of transferring the calls to the adequate specialist operator. Generalists must then know the answers to the more usual queries, be able to detect questions outside their competence, and be able to choose the relevant specialist. This choice is made on the basis of some key-words found in the caller's initial request. Concerning this process, the author spots two interesting elements:

- *understanding, for the generalists, does not mean being able to give a definition of the key-words, but being able to categorize these words* . Generalists have to be able to react to very different specialized vocabularies. In some cases, they know the meanings of these words; but it may also happen that, without really knowing the meaning of a word, they are still immediately able to specify the domain to which it belongs, and then transfer without hesitation the call to the appropriate specialist;

- *the key-words for the generalist differ from the key-words for the specialist.* This means that the elements used for categorizing the query are not those which allow the elaboration of an answer. In other words, the operative processing schemata of the specialist and of the generalist are not the same. This is of course a consequence of the fact that their tasks differ.

The task could have been organized in other ways. There could be a single category of operators, who would receive and process all calls, whatever the nature of the query. But, considering the extent of the domain, this would mean that, in a large number of cases, these operators would be in a problem-solving situation, facing totally new requests. Thanks to the organization described above, the operators are always able to use operative knowledge structures (except in very unusual cases), and then proceed quickly and efficiently (as far as the mental processes involved are concerned).

The operators (generalists or specialists) use the same (operative) mode of functioning, but with different knowledge structures. What we see here is not (as it was the case above for telephone information operators) the allocation of different modes of reasoning to different operators, but rather the allocation of different hierarchical steps to different operators.

These results provide hints for the design of intelligent interactive systems. They suggest a multi-expert system structure, in which one of the experts would be in charge of allotting the job to the other experts of the system.

Here again, this is quite different from what we usually observe in the functioning of present understanding systems. In most cases, these systems behave like an all-knowing operator, expert in several domains. For this reason, the specific competence consisting in determining the domain of expertise which must be triggered does not exist in these systems.

As we have seen, understanding systems are not economical because they use a single mode of functioning. A second drawback is that they are not given the capability of operating at the level of abstraction at which economical procedures can be used.

References

Bailey, G.D. (1985) The effects of restricted syntax on human-computer interaction (Report MCCS-85-36). Las Cruces: New Mexico State University, Computing Research Laboratory.

Benbasat, I., Dexter, A.S. & Masulis, P.S. (1981) An experimental study of the human computer interface. Communications of the ACM, 24 (11), 752-762.

Bevan, N., Pogbee, P. & Somerville, S. (1981) MICKIE - A microcomputer for medical interviewing. International Journal of Man-Machine Studies, 14, 39-47.

Borenstein, N.S. (1986) Is English a natural language? In: D. Beech (Ed) The Future of Command Languages. Amsterdam: North Holland.

Card, W.I. & Lucas, R.W. (1981) Computer interrogation in medical practice. International Journal of Man-Machine Studies, 14, 49-57.

Chapanis, A. (1978) Interactive communication: a few research answers for a technological explosion. 86th Annual Convention of the American Psychological Association. Toronto, Ontario, Canada.

Clancey, W.H. (1983) The epistemology of a rule-based expert system. A framework for exploration. Artificial Intelligence, 20, 215-251.

Clark, H.H. & Wilkes-Gibbs, D. (1986) Referring as a collaborative process. Cognition, 22 (1), 1-39.

Evans (1976) Improving the communication between people and computer. In: Proceedings of the NATO ASI on Man-Computer Interaction. Alphen aan den Rijn, Netherlands.

Falzon, P. (1983) Understanding a technical language (Research report 237). Rocquencourt: INRIA.

Falzon, P. (1984) The analysis and understanding of an operative language. In Proceedings of INTERACT'84, 1st IFIP Conference on Human-Computer Interaction, Londres, 4-7 Septembre 1984.

Falzon, P. (1986) Langages opératifs et compréhension opérative. Thèse de Psychologie, Paris: Université Paris V.

Falzon, P., Amalberti, R. & Carbonnel, N. (1986) Dialogue control strategies in oral communication. In: D. Beech (Ed) The Future of Command Languages. Amsterdam: North Holland.

Gould, J.D., Lewis, C. & Becker, C.A. (1976) Writing and following procedural, descriptive and restricted syntax language instructions (Report RC 5943). Yorktown Heights, New Jersey: IBM Watson Research Center.

Hauptmann, A.G. & Green, B.F. (1983) A comparison of command, menu-selection and natural-language computer programs. Behaviour and Information Technology, 2 (2), 163-178.

Hayes, P.J. & Reddy, D.R. (1983) Steps towards graceful interaction in spoken and written man-machine communication. International Journal of Man-Machine Studies, 19, 231-284.

Hendler, J.A. & Michaelis, P.R. (1983) The effects of limited grammar on interactive natural language. In: A. Janda (Ed.). Proceedings of the CHI'83 Conference on Human Factors in Computing System, Boston, Mass., 12-15 Dec. 1983.

Krauss, R.M. & Glucksberg, S. (1977) Social and nonsocial speech. Scientific American, 236 (2),100-105.

Krauss, R.M. & Weinheimer, S. (1964) Changes in reference phrases as a function of frequency of usage in social interaction: a preliminary study. Psychonomic Science, 1, 113-114.

Kupka & Wisling, (1980) Conversational languages. New York: Wiley.

Leroy, C. (1985) Structure des requètes. In Analyse linguistique d'un corpus d'oral finalisé. Etude collective sous la direction de M.-A. Morel. Report to the GRECO Communication Parlée (GRECO n° 39).

Lubonski, P. (1985) Natural language interface for a Polish railway expert system. In V. Dahl & P. Saint-Dizier (Eds.), Natural language understanding and logic programming. Amsterdam: North-Holland.

Ogden, W.C. & Boyle, J.M. (1982) Evaluating human-computer dialog styles: command vs form/fill-in for report modification. Proceedings of the Human Factors Society 26th Annual Meeting, Seattle, Washington, 25-29 octobre 1982.

Onorato, L.A. & Schvaneveldt, R.W. (1986) Programmer/nonprogrammer differences in specifying procedures to people and computers. In E. Soloway & S. Iyengar (Eds.) Empirical studies of programmers. Norwood, N.J.: Ablex.

Potosnak, K.M. (1984) Choice of interface modes by empirical groupings of computer users. In Proceedings of Interact'84, Londres, 4-7 September 1984.

Ramsey, H.R. & Grimes, J.D. (1983) Human factors in interactive computer dialog. In: M.E. Williams (Ed.) Annual Review of Information Science and Technology (Vol. 18). White Plains, N-J: Knowledge Industry.

Richards, M.A. & Underwood, K. (1984) Talking to machines: How are people naturally inclined to speak? In E.D. Megaw (Ed.) Contemporary ergonomics 1984. London: Taylor & Francis.

Roman Parré, M. (1984) La fonction de renseigner. In: INA (Ed.) La communication dans la ville: du discours municipal à la mission de renseigner. Paris: INA.

Scapin, D.L. (1985) Intuitive representations and interaction languages: an exploratory experiment. In F. Klix (Ed.) MACINTER I. Amsterdam: North Holland.

Shapiro, S.C. & Kwasny, S.C. (1975) Interactive consulting via natural language. Communications of the ACM, 18 (8), 459-462.

LE PROBLEME DE LA DESCRIPTION

Session C

THE DESCRIPTION PROBLEM

HUMAN-COMPUTER INTERACTION MODELS AND THEIR USE IN COMPUTER SYSTEM DESIGN

Andy Whitefield

Ergonomics Unit, University College London

1. INTRODUCTION

The research literature on cognitive aspects of human-computer interaction has grown rapidly in the last ten years (see Megaw and Lloyd, 1984). One common way of presenting the outcomes of this research is as models of computer users, programs and systems. Often accompanying these models is the suggestion that they could be useful in computer system design. Unfortunately, few of the authors suggest how the models might be used, and fewer still actually demonstrate their use.

This paper critically examines the use of human-computer interaction models in computer system design. It considers a number of the models that have been proposed and asks what kinds of contribution they offer designers. Section 2 briefly discusses the range of models in the literature. Using a classification scheme, four models are selected and described. Section 3 considers the roles these types of model could play in computer system design. Finally section 4 compares and contrasts the roles from a number of perspectives.

2. MODELS OF HUMAN-COMPUTER INTERACTION

Models are a favoured way of expressing much of the knowledge acquired in human-computer interaction research. Even if, as will be done here, one restricts one's interests to those models concerned with individual users rather than with groups or organisations, there are still many models in the literature. In an attempt to clarify some of the relationships between them, Whitefield (1986) has proposed a simple classification. This divides the models according to who or what is being modelled, and who or what is doing the modelling. Space does not permit a full discussion of the classification here, but it identifies three classes of model that are of particular interest for this paper. These are researchers' models of the program, of the user and of the system. They are of particular interest here because they can potentially be used at early stages of system design, before even a prototype version is available for user testing. These early stages of design are those with which this paper will be concerned. The models are thus what Reisner (1983) calls analytic tools, which she contrasts with empirical testing. That is, the models are abstract representations of part of the proposed system and can be examined or manipulated to predict some human factors aspect of the system in use. Example models of each of the three classes mentioned above will be described in turn, plus a fourth type arising from a model being classified as belonging to two classes.

A researcher's model of the program is concerned with some aspect of the structure or operation of the software, independent of the user. Examples are the state transition model of Embley and Nagy (1981), the generalised transition network of Kieras and Polson (1985), the surrogate model of Young (1981) and the formal grammar of Reisner (1981). The latter will be

used here as an example. Reisner describes an interactive graphics system in a grammar using a BNF-style notation. It has a set of terminal symbols (e.g. position cursor, rotate knob) and a set of rules for combining these in various ways to generate legal input strings. These legal strings represent sequences of actions that will invoke certain command operations. Although termed a cognitive grammar, what it really describes is the command language of the program. Thus the basic entities refer to parts of the hardware and software rather than to the user's view of the task or program. The relationship of this model to the user's view poses an interesting question that will be discussed below.

A researcher's model of the user is concerned with some aspect of the user's behaviour relevant to the task, but which does not include details of the software being used. Examples vary widely in what they address, but include the model human processor of Card, Moran and Newell (1983), the block interaction model of Morton, Barnard, Hammond and Long (1979), and the author's blackboard design model (Whitefield, 1985, 1986) which will be described further here. This gives a decomposition into approximately twenty classes of the knowledge applied by a mechanical engineering designer during a design task. These classes can be grouped into four types. In addition the model provides an architecture for the application of this knowledge, in a form derived from the blackboard models of artificial intelligence (Hayes-Roth, 1983). Thus the model describes the knowledge the designer brings to bear upon the task plus the way in which this is done. Both these static and dynamic aspects will vary for different tasks and for different design strategies. The model is intended to be used during the design of CAD systems to improve their interfaces.

A researcher's model of the system is a single model containing a combination of elements describing both the user and the program. The most detailed are undoubtedly the models of text-editing known as GOMS, and the unit-task level and keystroke level models derived from them (Card et al, 1983). These latter two models are intended as practical design tool versions of GOMS. The keystroke level model describes system performance in terms of time on task for an expert user of a given program on a given task using a given method. It contains a set of operators (at the level of keystrokes, pointing, mental operations, and so on) and a method describing sequences of operators for accomplishing a task. Given the command language of the program, the designer can construct methods for any task. Then standard times assigned to the operators enable the calculation of predicted performance times for each method.

Using this classification, there are some models in the literature that do not clearly fall into only one class. Although presented as single models, they really consist of two separate models with a defined relationship. These two models are usually of the user and of the program. They differ from models of the system as described above in that there are two separate constituent models. Examples are the task/action mapping of Young (1981), the external/internal task mapping of Moran (1983), and the goal structures model of Morton et al (1979). Although the latter is not worked through in detail, it will serve here to illustrate this type of mixed model. Again using the example of text-editing, it describes the hierarchic task structures of both the user and the program. In practice, the user's task structure may be partly pre-determined and partly developed during the interaction. The user and program models are separate descriptions but use the same notation. They can then be explicitly

mapped, and the similarities and discrepancies noted. This should indicate likely areas of difficulty.

The four models outlined in this section, each concerned with its own task domain, will be used in the remainder of the paper to illustrate different applications of models in computer system design. That a classification was used to select them gives reason to believe they are representative of the kinds of model in the human-computer interaction literature, or at least are not simply four random examples. However, what is said about the roles of each particular model should not at this stage be taken as applying to the other models in each class. Further analysis is needed to establish the relationships between roles and classes. The models will usually be referred to by the following abbreviations: FGM (Reisner's Formal Grammar Model); BDM (Whitefield's Blackboard Design Model); KLM (Card et al's Keystroke Level Model); and GSM (Morton et al's Goal Structures Model).

3. ROLES

All four models make some claim to be useful in computer system design. Indeed, this is the major point of the KLM and BDM, the ultimate aim of FGM, and an important if unexplored consideration for GSM. One can therefore ask what it is that is being offered to designers - what roles are these models meant to play? The four models will be considered in turn.

Using the example of an interactive graphics system, Reisner tries to use the FGM as an evaluative tool for predicting areas of performance difficulty. Thus, having described the program in the grammar notation, she examines this to assess likely problems for the user. For example, learning to select text is likely to be problematic because the commands vary from those previously learnt to select shapes. The kinds of additional assumptions needed to derive such predictions from the model will be discussed in the next paragraph. All the predictions discussed by Reisner concern errors and learning difficulties. Other aspects of performance are not addressed.

It should be noted that Reisner compares the models for two versions of a program to highlight probable difficulties. But her stated aim is "to identify intrinsic characteristics of a language which make it easy to learn and use" (1981, p.237). That is, she wishes to develop psychologically based representations of programs. But there are two assumptions involved here that make this an implicit match with a model of the user. The first is that the user's representation of the command language will be a grammar of this type. The second concerns the desirable psychological features of the grammar. For example, Reisner assumes, on the basis of human memory limitations, that it should have few rules and a large number of terminal symbols (at least she states clearly some of the assumptions made). Such assumptions about the psychological relevance of features of the grammar may be appropriate and they may not. Thus although the FGM is a model of the program, its use involves a match with an implicit model of the user. Although this class of model has potential for improving design, this kind of match must be made more explicit if the models are to have a sound basis for their use.

Thus the FGM is to be used to predict aspects of performance. In its current form it does require a knowledgeable designer, partly to flesh out

the implicit match. It is not a metric or formula to be applied automatically, but requires some skill from the designer in the way the description is interpreted.

The BDM plays a different role for the designer. It is intended to allow the assessment of user/program compatibility. The designer uses it to reason about the effects possible CAD program features might have on the user's knowledge recruitment. Relevant criteria about what constitutes an appropriate and desirable use of knowledge enable decisions to be made between possible systems. For example, the system designer might be able to reason about what would be appropriate combinations of displays to reduce navigation in a two-screen CAD system, or to reason about suitable program methods for encouraging the user in a breadth-first design strategy. In each case the system designer uses the model to predict the user's knowledge recruitment under certain circumstances.

This approach has various disadvantages. Perhaps most serious is the lack of a direct relationship between the user and program representations. It is not clear from Whitefield's work how the program should be represented to match the model to it. This makes the model application a very flexible process open to different interpretations, but it does mean it can address a wide range of issues. It can also avoid a problem facing other models such as the GSM - that of matching new programs with novel capabilities to the user's existing task structures. This might lead one to miss many potential benefits. By having a model in terms of knowledge and not in terms of a particular task structure, one could assess the potential benefits and drawbacks of a novel program.

The use of this type of model demands considerable effort and skill from the designers. They would need to learn the structure of the model and how it operates under different circumstances. The model could not be applied in a rote manner and a good deal of scope for interpretation is available.

The role of the KLM is as an evaluative performance metric for choosing between system configurations. Given a detailed description of a text-editing program, then a set of rules assign operators to each step of a specified task and method. The times associated with the operators allow the KLM to model system performance time for that task. The designer can then compare this with some benchmark or criterion, or use it to select between alternative possible systems in terms of speed.

There are several assumptions within the KLM (e.g. when 'mental' operations occur; all mental operations take the same time), but there are also assumptions about the model (e.g. performance time is an adequate discriminator between systems; semantic and syntactic differences between the systems will not affect performance time). Card et al try to deal with some of these issues by the validation of the GOMS models from which the KLM is derived. But this is not possible for all of them.

Card et al are very clear about the way in which the KLM could be used. They suggest that just as the engineering designer uses calculational formulae, the computer system designer should do the same.

> "The ability to do calculations is the heart of useful, engineering-oriented applied science. Without it, one is crippled." (1983, p.10)

Thus the KLM has a clear analogy with the use of, for example, daylight factor indices in architecture and stress analyses in civil engineering. It has a routine application method producing a single result. The designer is required to have little or no understanding of user behaviour to apply it. Although the KLM clearly has the potential for automated application, thus increasing its ease of use, it is not clear this would be the case for all models of this type.

The role of the GSM is to allow the assessment of user/program compatibility in terms of task structure. The example given concerns a text-editing task. The hierarchic task structure brought by the user from previous work experience, and that implicit in the program are represented similarly. An explicit comparison between them highlights mismatches in both identities and arrangements. For example, it may be that the output file is opened at different points in each description, or, at a more detailed level, that the sequence of operations required to underline text varies. Such mismatches identify areas of difficulty, thus raising questions about the design.

There are two particular problems with this approach. The first is appreciating the significance of mismatches, both individually and comparatively. For example, how important is it if the model indicates the user's and the program's views involve opening the output file at different points? Is this problem more or less serious than a mismatch in the point at which the edited text is filed? The second problem is the danger of placing too much emphasis on maintaining the user's existing task structure. It may be that the program offers potential improvements that may not be implemented because they do not match the user's task structure. This was mentioned above in connection with the BDM.

The GSM does not involve any quantification and could not be applied automatically. The designer must be skilled enough to develop and interpret the task structures and appreciate the significance of matches and mismatches.

The four models under discussion play different roles for the designer. The FGM produces a representation of the program that can be examined for likely errors and learning difficulties. The BDM produces decompositions of the users' knowledge into classes and is aimed at user/program compatibility in the sense of an appropriate application of this knowledge. The KLM produces predictions of time on task for a system and is oriented towards evaluating performance time. The GSM produces explicit analyses of users' and programs' task structures and attempts to assess compatibility in those terms.

4. COMPARISON OF ROLES

The model roles discussed in the previous section vary in a number of ways. Four of these variations will be discussed here: the ease of use of the models; the aspects of performance they address; their products for the designer; and their functions for the designer. These different perspectives will serve to clarify the nature of the models' design roles.

Comments have been made throughout the previous section on the ease of use of each model. At one extreme the KLM has a straightforward application and an apparently meaningful quantitative result. At the other extreme the

BDM has a very flexible application demanding considerable understanding on the designer's part, and producing qualitative analyses of the user's knowledge recruitment. These alternatives correspond respectively to what Harker and Eason (1984) call the 'information acquisition' and 'knowledge transfer' approaches to the use of ergonomic knowledge by designers. Information acquisition approaches (e.g. guidelines and checklists) try to provide designers with specific information that requires little understanding of user behaviour to be applied. Knowledge transfer approaches involve designers learning human factors skills. These may be either methodologies like task analysis, or technical expertise in particular aspects of user behaviour.

Each approach runs its own risks. The knowledge transfer approach risks unskilled use resulting in misinterpretation both in its application and in assessing its outcomes. The information acquisition approach risks being misleading in its apparent quantification and being too simplistic and restricted for the complexity of human-computer interactions. This approach tends to treat designers as if all they can use is a tool with a straightforward and prescribed application. While such tools can be useful, the general approach does little to acknowledge the creative variety of the design activity. If instead we treat designers as able to make intelligent use of the available information - the knowledge transfer approach - it is not clear they could not make good use of models like the BDM, even if their exact mode of use is not prescribed. Thus Hammond, Jorgensen, MacLean, Barnard and Long (1983) report that designers do use their own, ill-specified models of users. They suggest that, rather than detailed guidelines,

> "Appropriate 'mini-theories' might be more suitable tools for the designer to view the user and the user's task." (p.44)

Hammond et al also report system designers as claiming their greatest need for human factors information was help with initial task analysis. Putting these two things together, non-prescriptive task-oriented models of users might be very appropriate to give designers.

A different way to look at the models is to ask what aspects of performance they address. The aim of applying ergonomic knowledge to design is to improve performance in some way. But performance is not a unitary measure and an improvement in one aspect could easily lead to deterioration on another. What are the variables by which one might assess performance? Card et al (1983, p.404) suggest the basic performance variables of a human-computer system are: functionality; learning; time; errors; quality; and robustness. While the KLM (time) and FGM (errors and learning) clearly address performance variables directly, the BDM and GSM do not. These latter models more directly address the compatibility of the parts played by the two contributors making up the system. They are thus concerned not so much with performance as with system organisation underlying performance.

A further perspective on the roles is to ask what the products of the models are - what is it the designer is supposed to be able to use? The various products might be listed as follows: FGM - program representation allowing identification of likely errors and difficulties in performance; BDM - decomposition of the user's knowledge plus a scheme for its application; KLM - prediction of performance time from a task decomposition in terms of low-level operations; GSM - hierarchic

structuring of the user's and the program's goals, illustrating mismatches.

These products do not clearly fall into any natural divisions. One distinction is between a concern for user knowledge and a concern for some aspect of the task. But it is difficult to carry this distinction very far, since the knowledge required to organise and execute sub-tasks soon intrudes upon a simple task analysis of actions. Nevertheless, the knowledge-oriented BDM is concerned with the classes of knowledge recruited by the user in the interaction, while the other task-oriented models are concerned with some relationship between user actions and the program. Of the latter models, some attempt to predict performance, while others attempt to lay out the structure of the task.

Although they make useful distinctions, neither of these last two perspectives provides a reliable discrimination between the models that is insightful about their design roles. A more fruitful view is to ask what functions the models perform for the designer. This can then be related to explicit characterisations of the design activity.

The KLM is meant as an evaluative formula with which to calculate performance time. The major problem with this sort of approach is the assumed relationship between the measure and the concept of usability. The dangers are that what is measured may not be meaningful in terms of usability, and that only the easily measurable aspects of performance will be considered, while other important aspects will be ignored. The KLM is clearly susceptible to both these dangers. The second one is more difficult to avoid in view of the complexity of large interactive computer systems. Card et al are of course aware of this danger but they fail to do more than simply acknowledge it (1983, p.410). We need to know that the emphasis on performance in terms of time is not leading to a deterioration on other measures, such as functionality or quality. Time on task is not by itself necessarily a good measure of performance. With its restrictive scope, the KLM can clearly form only a small part of any human factors analysis of a projected system.

Although the FGM is not currently an evaluative formula in the way the KLM is, Reisner wishes to move towards such a form. The development of the approach reported briefly in Reisner (1983) involves converting grammar sentences into equations and solving these to predict time or error performance. Given this similarity with the KLM in overall approach, one can see the FGM suffers similar problems concerning the relationship of the output with the concept of usability. Since the FGM is a model of the program, there is the additional problem mentioned in section 3 of the implicit match with a model of the user, and the psychological assumptions of this.

In terms of their function for the designer, the KLM and FGM are evaluations of system performance. An important point is the stage at which the evaluation is used. Thus the models are concerned with the output performance of the proposed system. They do not assess the two constituent parts of the system, nor are they generative about the content of possible systems.

The distinction between the generation and evaluation of solutions is common in the design literature (although different terms may be used).

The KLM and FGM can be clearly identified as contributing to the evaluation function. The BDM and GSM have a different function which can contribute to both generation and evaluation. This is not to evaluate system performance but to assess user/program compatibility.

This assessment is done in one of two ways. The first is to compare, in similar representations, the content and organisation of the user and program halves of the system. This is the role of GSM. This kind of role can be evaluative in that one could compare alternative programs for their overall match with the user. But also the use of the model raises questions for the design by highlighting particular areas of difficulty, and in that sense it plays a more generative role than the strict evaluations.

The second way of assessing compatibility is to utilise a model of the user's knowledge to reason about the effects of possible program features. This is the role of BDM. Again this can play an evaluative role by helping to decide whether particular features are beneficial. But it can also play a more directly generative role. The BDM, in conjunction with the criteria for an appropriate use of knowledge, constitutes a view of the skill of engineering design that the CAD program is meant to support. As such it directly suggests improvements to the system (e.g. the provision of prompts to stimulate domain knowledge use). To what extent this effect depends upon the type of skill being studied is currently unclear.

Viewing the model roles in terms of their functions for the designer has been the most fruitful perspective. The distinction between assessments of user/program compatibility and of system performance indicates the orientations of the models, and relates well both to their ease of use and to the different design activities of generation and evaluation. The correspondence between the different functions and the classes of models outlined in section 2 is something that needs further consideration with more models.

5. SUMMARY

Four human-computer interaction models have been examined for their potential use during computer system design. They all conform to the general view that models can be a productive way to get ergonomic considerations into design (a view that is opposed by, for example, Gould and Lewis, 1983). But within this general view, the models vary both in what it is they model and in the roles they are meant to play for designers. Viewed in terms of their functions for the designer, the roles broadly fall into two complementary classes. These relate strongly both to the ease of use of the models and to their contributions to the evaluative and generative activities of design.

ACKNOWLEDGEMENTS

I would like to thank John Long and Clive Warren for their comments on an earlier version. Most of this work was done while the author was supported by an SERC research studentship. The author is now working on a project funded by GEC Research.

REFERENCES

Card S.K., Moran T.P., and Newell A. (1983): The Psychology of Human-Computer Interaction. Hillsdale, New Jersey: Lawrence Erlbaum.

Embley D.W. and Nagy G. (1981): Empirical and formal methods for the study of computer editors. In: M.J. Coombs and J.L. Alty (eds): Computing Skills and the User Interface. London: Academic Press.

Gould J.D. and Lewis C. (1983): Designing for usability - key principles and what designers think. In CHI'83 Proceedings, 50-53. Published by ACM

Hammond N., Jorgensen A., MacLean A., Barnard P., and Long J. (1983): Design practice and interface usability: evidence from interviews with designers. In: CHI'83 Proceedings, 40-44. Published by ACM.

Harker S.D.P. and Eason K.D. (1984): Representing the user in the design process. Design Studies, 5(2), 79-85.

Hayes-Roth B. (1983): The blackboard architecture: a general framework for problem solving? HPP Report No. HPP-83-30, Stanford University, Computer Science Dept.

Kieras D. and Polson P.G. (1985): An approach to the formal analysis of user complexity. Int. Journal of Man-Machine Studies, 22, 365-394.

Megaw E.D. and Lloyd E. (eds) (1984): Human-computer interaction. Special issue of Ergonomics Abstracts, 16(1).

Moran T.P. (1983): Getting into a system: external-internal task mapping analysis. In: CHI'83 Proceedings, 45-49. Published by ACM.

Morton J., Barnard P., Hammond N., and Long J. (1979): Interacting with the computer: a framework. In: E.J. Boutmy and A. Danthine (eds): Teleinformatics '79. 201-208. Amsterdam: North-Holland.

Reisner P. (1981): Formal grammar and human factors design of an interactive graphics system. IEEE Transactions on Software Engineering, SE-7(2), 229-240.

Reisner P. (1983): Analytic tools for human factors of software. In: A. Blaser and M. Zoeppritz (eds): Enduser Systems and their Human Factors. Lecture Notes in Computer Science No. 150. Berlin: Springer-Verlag.

Whitefield A.D. (1985): A model of the engineering design process derived from Hearsay-II. In: B. Shackel (ed): Human-Computer Interaction - INTERACT'84. Amsterdam: North-Holland.

Whitefield A.D. (1986): Constructing and applying a model of the user for computer system development: the case of computer aided design. Unpublished PhD thesis, University of London.

Young R.M. (1981): The machine inside the machine: users' models of pocket calculators. International Journal of Man-Machine Studies, 15, 51-85.

DEVELOPMENT OF MENTAL MODELS OF OFFICE SYSTEMS

Gerrit C. van der Veer, Michael Felt, Bert van Muylwijk

Free University, Amsterdam
Dept. Psychology

Learning an integrated office system (editor, spreadsheet, database, file manager, mail system, graphics tool) is analysed from the point of view of the development of a mental model. To this end a conceptual model of the user interface has to be defined, and adequate analogies and metaphors have to be found that apply to knowledge that is available before the introduction of the system, which may be different for different user-groups.

In the case of a new course about such a system, a field studie is conducted, in which the conceptual model is the starting point of the design of the course. This model is developed according the levels of Moran's CLG. Metaphors are constructed in relation to the prior experience of the intended user groups. The modes of the metaphors are chosen in order to deal with individual differences in preference for representation mode of information (imagers vs. non-imagers).

The quality of the resulting mental models is investigated by measuring the correctness of knowledge on semantic level, the preference for different modes of interaction, and the representation the users apply in communicating about the system. The representations used reveil individual differences in the use of images and verbal predicates, and in the preference for the levels of Moran's CLG. Representations are found on task level, semantic level up to key-stroke level, for students who have never heard of this distinctions.

Individual differences in preferred mode of representation in the first field study showed an advantage of imaging. In a second course the analogies and metaphors used are adjusted in order to cope with the learning difficulties of non-imagers.

ASSESSING COGNITIVE ASPECTS OF LEARNING AND USING COMPUTER SYSTEMS

Michael Wilson and Allan MacLean
MRC Applied Psychology Unit
15, Chaucer Road, Cambridge, CB2 2EF, U.K.

Introduction

It is widely acknowledged that the behaviour of users and their underlying cognition is complex. It is unlikely that any single theory or assessment technique will tell one everything which might want to be known about this cognitive activity. Consequently, a number of specific assessment techniques and theoretical interpretations must be matched to specific purposes. In order to match these two, it is necessary to understand the properties of the possible theoretical interpretations and assessment techniques.

Assessment techniques can be used either for research purposes, to provide information about how users represent and process information, or to assess commercial products as part of the design process. In the design process, this assessment can be at an early stage, in order to provide ideas, or late, in order to locate user difficulties. If one wishes to obtain quick results with minimal analysis effort in the design process there must be a conscious trade-off between the effort to complete any test and the speed and amount of information provided. Therefore, it may not be feasable to apply the full range of detailed and laborious measures used for research purposes. Nevertheless, the interpretation of research tests and the relationships between them can provide criteria for selecting which tests may be appropriate, and their validity for the more practical problems of designing and assessing a system.

This paper considers different assessment techniques in the context of a study of users learning a commercial integrated office system (VisiOn; see Lemmons, 1983). The purpose of the study was to trace the development of different aspects of the cognitive processes and representations which support user performance during the learning of a system. Since it is acknowledged that no single assessment technique could deliver this, several different techniques were used. In order to incorporate the results delivered by these different techniques, it is necessary to develop explicit models of what each test delivers, and the view of the user onto which such results will be mapped. An attempt will be made in this paper to be explicit about both. The techniques used involve three basic strands: performance measures; measures which allow access to articulatable knowledge; and measures which implicitly require subjects to use knowledge about a system. There were overlaps and differences in the aspects of performance and cognition captured by the tests used in the study. This paper will outline the techniques used, and illustrate both the similarities and distinctions in the results, in order to understand what aspects of cognition and behaviour each test is best at capturing. In particular, the view of learning which can be captured by these tests will be described.

Details of Study

There were 4 different performance measures (called the 'A', 'B' and 'C' set tests, and a data entry test). Although these included subjects verbalising occasionally, there was one explicit test of verbalisable knowledge: the Prompted Knowledge Elicitation test (PKE). There were two tests that implicitly required the use of the subjects knowledge of the system, a questionnaire study and a task which involved the sorting of command terms into the system hierarchy.

The study was arranged so that each of the 8 users in the study attended 8 sessions of two hours each. The sessions were systematically structured to enable the training, exercising and testing of, a broad sample of VisiOn functionality. The overall design of the study is presented schematically in Figure 1.

In session one, users familiarised themselves with the keyboard and mouse as well as the basic concepts associated with windows and menus by using the in-built tutorial within the VisiOn system. Users then began a systematic training programme. The training was by example and illustration. Users were shown videotaped examples of methods, asked to copy them and then practice them. As the sessions progressed more and more functions were illustrated, copied, and practiced. Details of individual tests are given later.

FIGURE 1. The structure of the testing sessions in the observational study on VisiOn.

SESSION					
1	TUTORIAL	VIDEO PRACT A1			
2	VIDEO PRACT A2	A SET			
3	VIDEO PRACT B1	A SET			PKE
4	VIDEO PRACT B2	A SET	B SET 1		
5	VIDEO PRACT B3	A SET	B SET 2		
6		A SET	B SET 3	C SET	PKE
7	RECALL, QUESTIONNAIRE, DATA ENTRY TEST				
8	DATA ENTRY TEST				

Key: Video Pract - watching a video of command sequences, imitating them, and then practicing each command sequence. A1 & A2 for A set test. B1 for B set 1, etc ...
A, B & C SETS - three performance tests.
PKE - Prompted Knowledge Elicitation test.
RECALL - A command name sorting test.

The training was arranged with sessions 1 and 2 on consecutive days and the remainder at intervals of 3 to 7 days. The duration of each session was fixed at 2 hours since this was thought to be a reasonable sample period for the intended users of such systems.

Performance tests

Each of the performance tests in this study was selected to answer specific questions about the process of learning. Firstly, how does performance improve with practice and is there any evidence of proceduralisation? Secondly, do users incorporate new functions more

easily as a result of the accrual of general system knowledge? Thirdly, how does this knowledge generalise to untaught functions? Fourthly, when users reach a state of expertise in a specific task, do they make judgments as to the most appropriate method to perform that task on the basis of the time to use the possible methods, or on the basis of other knowledge?

A general view of learning (e.g. Anderson, 1983) suggests that the information that supports behaviour passes from being declaratively represented to being procedurally represented. During this change in the representation of knowledge, there is an hypothesised parallel reduction in the time to perform the task and the number of errors during performance. It is often assumed that as experience with a computer system increases, the general task performance time and error count will reduce. The 'A set' test was designed to assess these assumptions. A detailed report of this test is given elsewhere (Wilson et al, 1985), so only a brief summary will be presented here. In the A set test, subjects imitated and practiced nine basic command sequences in the first two sessions and were then tested on the performance of these nine commands on sessions 2 to 6 (but using different materials), giving five progressive samples of performance on frequently used commands.

To assess whether users incorporate new functions more easily as a result of the accrual of general system knowledge the 'B set' performance test was designed to assess the acquisition of new functions as learning progressed. Thus command sequences learned in one session were tested in a subsequent session. However, in each case the command sequences were tested only once. In tests from sessions 4 to 6, 18 different command sequences were tested.

To decide if the knowledge acquired generalises to untaught functions a third performance test (the C set) was administered to capture expert performance on commands unknown to the expert. This test was administered at the end of session 6, and included users attempting six tasks which required the use of novel system functions. However, the test was only attempted by the three fastest subjects, who made only five errors between them. This is too little data on which to answer the question conclusively, although this was a reasonable indication of the generalisation from knowledge of other parts of the system, for at least the fastest subjects.

A simple psychological model of expert performance (e.g. Card et al, 1983) assumes that expert users make decisions as to the optimal method to perform, only on the basis of the time to complete a method. This expression may be too simple to explain performance and other factors may have to be taken into account. To assess if other factors are involved in the decision, the fourth performance test was designed. A data entry test was used to sample the choices made by experts between methods to achieve tasks. The data entry test was designed as a study of the tradeoff between two methods of entering data into a spreadsheet (this study is reported in detail elsewhere: MacLean et al, 1985). This test took place in sessions 7 and 8. The two methods compared were 1) selecting each cell in a matrix using a mouse, and typing in numbers; 2) using a menu to establish an automatic cursor movement from cell to cell, then typing numbers into the cells of the matrix. The first method took longer for each cell,

but the second included an overhead of a long setup time. Subjects used both methods for matrices of various sizes for one hour after their other experience with the system. Following this practice, they were asked to choose, and use, one method to enter data in each of 12 different matrices in both directions. For small matrixes the mouse method was faster; for large matrices the setup time for the menu method became proportionally smaller, so that method was faster. Since subjects used each method frequently before they were asked to choose the optimal method for each matrix, it was possible to calculate the time each subject took to perform each stage of the entry methods. It was also possible to calculate at which matrix size the temporally optimal change in methods occurred for each subject.

All performance tests were recorded on a timestamped videotape. For the first three tests, four different analyses were then performed: 1) the time subjects took to perform the task; 2) the number of tasks achieved without major re-attempts; 3) the number of tasks achieved in a single goal; 4) and an analysis of all deviations from an optimal route for performing the tasks (for the A set, these are reported in detail in Wilson et al, 1985). In order to make such analyses, a task analysis and model had to be constructed to specify tasks, goals, major attempts and local deviations. The explicit model used for this study contained two parts. The first assumes that users have a top level goal to complete what is presented on the editing manuscript. Within this they then have subgoals to complete each of the three tasks. Users then make attempts to achieve each subgoal by constructing and executing a method. During the execution of this method, they test the consequences of actions. If tests fail they have options: 1) to make a local correction to the attempt; 2) to construct a new attempt to complete the subgoal; 3) to postpone the present subgoal and attempt another. If the test passes, the goal is completed. This model was further specified to account for the exact nature of the local correction, re-attempts, and goal changes within the command language of the VisiOn system, but this is presented elsewhere (Wilson et al, 1985) and will not be described here. There is a special case where, a goal is completed by the user, but the route prescribed was not used. These were classed as 'Ignored Failures'.

This formal model of the task permitted a very exact classification of the actions of users and the results are quantifiable. This contrasts with the informality found when observers note their reactions to a user's performance. Unsurprisingly, in the A set test the times to complete these overall tasks fall across sessions ($F(2,14) = 24.62$; $p < .001$) and the number of overall tasks completed without major errors increased ($F(4,28) = 3.72$; $p < .02$). These gross measures appear to support the assumption that general learning reduces performance time. However, a more exact analysis shows that the times to perform three tasks falls dramatically (e.g. from 254 to 80 seconds) while the times for the other five remain constant. Similarly, a gross measure of the number of command sequences performed without major re-attempts rises significantly from 65% to 80% ($F(2,14) = 4.82$, $p<.03$), although for five of the nine commands, and three of the eight users they do not.

In more detail, there was also a change in the classes of errors across sessions. The proportion of local corrections while attempting tasks doubled (see Table 1); in contrast the proportion of major

re-attempts dropped by two-thirds, while the proportion of errors due to changes to goals remained constant.

Table 1. Percentages of deviations from optimal route by class.

Test Session	Local Correction	Re-Attempts	Goal Changes	Ignored Failures	N
A set 2	39.7	39.7	17.6	3.0	68
6	67.8	12.9	16.1	3.2	31
B set 4	37.5	40.0	15.0	7.5	40
6	64.9	27.0	8.1	0.0	37

Overall, there are both users and tasks for which performance remains poor, and others for which it improves. It was also true that neither the time to carry out a correct command sequence nor the number of retries after a failed command reduced significantly. These results offer little evidence for the speeding up of performance for well formed sequences or their proceduralisation. Rather, the reduction in performance time is due to the reduction in major re-attempts.

It can be concluded that a view of the learning captured by this performance test which suggests that all sequences, or performance on all aspects of the system would improve synchronously, is inappropriate. Rather than a single progression, the data suggest that it would be more helpful to consider the gross performance measures as reflecting an averaging of individual command sequences each of which could lie at different stages of development within a user's repertoire of knowledge and skill. This repertoire as a whole develops, but parts of this knowledge are inaccurate, and these may persist and cause consistent errors.

The B set test was analysed in a similar fashion to the A set. Since the B set involved the testing of different command sequences of varying length on each session, it would be inappropriate to present time data. However, to answer the question as to whether the general improvement in system knowledge aided the acquisition of new command sequences, one can rely on error data. The number of errors for the sequences in the first B set test was the same as that for the third (see table 1), whereas it fell by half over the four sessions of the A set test. Although this would suggest that the acquisition of later sequences was not aided, an analysis of the pattern of errors leads on to the opposite conclusion. The shift in error types across sessions for the A set test that accompanied the reduction in performance time, was from major re-attempts to local corrections. The same shift, so that errors are more likely to be recognised in time to avoid the need for major re-attempts, exists in the errors for the B set from sessions 4 to 6. This suggests, that the information in the user's repertoire which is permitting them to move from time consuming major re-attempts on well practiced commands also affects the acquisition of new sequences.

The data entry task was designed to assess whether time alone was the criterion which experts used to select the best method to achieve a

task as is often assumed. The method allowed the calculation of the temporally optimal entry method for any data matrix for any subject. After much practice and experience, the users then selected the method they wished to use, and used it. The choice points actually used by subjects were not the same as the temporally optimal but showed a consistent bias in favour of one of the methods. This simple study of a small set of expert performance illustrates that this assumption is inadequate to explain the actual choices made. Further factors of cognitive load, or focus of attention must be included in the criteria, and these criteria must be made explicit. If the purpose of the analysis is purely to estimate performance time, then such models as the Keystroke model (Card et al, 1983) will be adequate. However, for other aspects of usability and learning, more complex models are required.

Tests of Articulatable Knowledge

If users give verbal protocols during performance, one assumes that this protocol reflects both the way that users perform the present task, and the way they would perform that task outside the laboratory. Both of these assumptions have been challenged. It has been suggested that such techniques present a danger of the verbalisation distorting the normal operation of cognitive processes (e.g. Ericsson & Simon, 1984). Consequently, alternatives to the simple talk aloud protocol have been suggested, either to the form and volume of data collected (e.g. Hammond et al, 1983), or in the method of eliciting protocols through taks other than the concurrent verbalisation of a single user performaing a task(e.g. Suchman, 1985).

An attempt to combine these advantages in a task which probes users knowledge was made in the prompted knowledge elicitation (PKE) test in the study of VisiOn which would give access to the users articulatable knowledge, without imposing a concurrent protocol task. The purpose of this test was to assess the changes in verbalisable knowledge during learning, and to assess the relationship between this and performance measures. In this test users are presented with pictures of screens, about which they are asked focused questions. The picture based interviewing technique, reduces the quantity of protocol to be analysed by presenting pre-specified questions about specific sample system states. It also permits a formalisation and quantification of the analysis by having pre-specified sets of target claims which could be produced by users in reply to the questions. The formalism of the task and dialogue, also permits a specification of the structure of the task, and of its analysis. The sources of information which are likely to influence a protocol content include the knowledge of the domain of application, knowledge of the system being studied, information in the photograph, the prompts from the experimenter, and the interpretation of the picture probe task itself.

A detailed report of this study is presented elsewhere (Barnard et al, 1986), so only a brief outline of the method and results will be presented here. The test was administered at the end of sessions 3 and 6, where the same 16 pairs of pictures were used. The experimenter prompts in this study, focused about: 1) the task the user would be performing to be in the state presented in the picture; 2) the command sequence that would have reached this screen; 3) the sequence from this screen that would be used to achieve the suggested task.

Information that did not fall in these classes was judged non-target and was dropped from the analysis. Within these classes of target information, user claims can be scored as, true (accurately describing system operation); false (in contrast to system operation); inexact (a claim neither clearly true nor false); or indeterminate (for ambiguous claims and explicit statements of ignorance).

There was a rise in the mean number of target claims from session 3 to session 6 (Wilcoxon $T=3$, $p<.05$) indicating that the users' verbalisable knowledge was increasing. It is possible that this rise was due to increased familiarity with the test, or ease of talking to the experimenter, but there is little grounds for this explanation. This rise is due to an increase in the number of true claims (subjects: Wilcoxon $T=3$ $p<.05$; materials: Wilcoxon $T=3.5$, $p<.05$), since the number of false, inexact and indeterminate claims did not rise significantly. However, within this rise there is a complex change in the exact claims which were repeated at the two sessions (46% of true claims at session 6; 31% of false claims).

The rise in true claims is not unexpected, and the fact that some false claims recurred is consistent with prior observations based on protocol examples and with the repertoire view of knowledge outlined above. However, two thirds of the false claims made in session 6 were new. Although these new claims arose for a number of reasons, more detailed analysis suggests that false claims were made in spite of known facts that were accurate. With one of the probes for example, only one user correctly identified it as the prompt state for editing cells in the spreadsheet. Although true claims were elicited concerning cells and work area actions, those concerning the task and prior menu selections nearly all focused on the entry rather than the editing of material. This confusion between menu paths was understandable in terms of their gross similarity. However, other probes elicited true claims from users which accurately described prompt states for entry. They had apparently not accessed that knowledge to block false inferences concerning the edit prompt. Thus, while the number of true claims may rise, the underlying knowledge from which the claims were derived may remain compartmentalised and functionally inaccessible in a novel context.

Tests Requiring Implicit use of Knowledge

There is a large set of possible off-line recall tests, however no strong relationship between recall and the usability of computer systems can be shown (MacLean et al, 1984). Despite this, if some evidence exists for a particular form of user representation being important in determining performance, other indirect measures can often provide useful converging evidence to form a more detailed conclusion. Two off-line tests were included in the VisiOn Study which were designed to elicit different aspects of users knowledge of the system and its operation. These were a questionnaire and a command name sorting task which were administered only in session 7.

The questionnaire included 40 questions to which users had to give true/false answers and confidence ratings on a four point scale. Specific questions were constructed in the light of the general user performance on previous tests. The purpose of the questionnaire was to test if users would recognise states of affairs or system rules even

if they could not articulate them, and conversely, to see if they would endorse false claims that were consistent with common user errors. The mean correct response rate was 70%, and of the 96 errors there were 73 false positives and 23 false negatives, with 11 questions accounting for 63% of the errors.

Of the questions which were consistently answered wrongly, some support user performance errors and others contradict them. One question asserted that the terminator 'done' occurred on all command menus. This is false, and yet was affirmed by 5/8 subjects (with a mean confidence of 1.6). The item 'done' was responsible for a large number of performance difficulties in the A set test, but the performance problems were mostly due to users forgetting to select it when available, rather than attempting to select it when it was not available. This affirmation of a falsehood supports the idea that where there are performance difficulties, they are due to ambiguity or uncertainty in the subjects mind, and since it contradicts performance, this iindicates uncertainty rather than the presence of false knowledge. In the light of this, it is surprising that users were extremely confident in the questionnaire that all their answers were correct (mean confidence = 1.35; 1 = very sure, 4 = Complete Guess).

The most common user error in the A set test was associated with typing text into an editor. This is the only task for which no menu selection need be made - it is the default state. However, many users searched the menu hierarchy for a command which would allow them to do this. The only question which all users answered incorrectly (with a mean confidence of 1.9), stated that there was an option at a specified point which enabled the typing of text. This again is totally consistent with user errors, however, all users managed to type text for each of the A set tests and on many other occasions although they never made a menu selection. Further, if they had a knowledge of the menu hierarchy at the point specified in the question (which they had frequent experience of) they would have ruled out the possible presence of a menu item there. These two examples both support the view that users are willing to endorse false statements about the system which are consistent with user errors. Such endorsements of false statements provide useful information about areas where errors in performance are likely to occur.

Both the PKE task and the questionnaire suggest that users are not good at recalling the menu structure in order to rule out interpretations, the name sorting task was designed to test incidental learning of the menu structures of the system. This data would supplement that for the C set performance data, and the PKE, to support a view of the representation that could support problem solving behaviour.

The VisiOn system contains three environments. Each environment uses a top level menu to select commands. Many of these menu selections result in the presentation of a second menu list. The 'cue' component of the task was comprised of the main menu lines from each of the three environments. The list of stimulus words to be sorted under each of these headings were taken from the menus that arise when menu items on these lists are selected. A total of 72 secondary menu items were selected from four menu headings in each of the environments. Subjects were presented with a large sheet of paper on which were headings

consisting of the main menu lines from the system. They were also given an alphabetic list of the 72 target words. They were instructed to write each target word under menu item that would be selected to get that word on a menu. Out of 576 items there were 314 assigned to some heading of these 141 were assigned correctly and 173 incorrectly. For the items that were used in the A set test, on average 6 subjects placed them correctly. For items used in the B set test, on average 3.3 subjects placed them correctly. Of the items not used, those that occurred on menus with the A set items were placed correctly by 1.6 subjects on average; those that occurred on menus with B set items were placed correctly by 0.8 subjects; and those that occurred on C set menu were placed correctly by 0.1 subjects.

This evidence suggests that users are developing some representation of the menu structure and that the accuracy of this representation corresponds to the frequency of usage of a particular menu. However, it would appear that there is very little representation of items that have not actually been used themselves.

Interrelationship of Tests

This paper has briefly described the results from six assessments of user knowledge which provided sufficient data to be analysable. Obviously a lot could be said about individual methods and their significance. In specific instances these have been fully discussed elsewhere. However, there are important similarities between several of these techniques. Common errors in the A set, are associated with false confirmations in the questionnaire. Commonly used functions in the performance tests are associated with correctly placed menu items in the sorting task. The distribution of errors in the A and B sets follow the same pattern, although they test different stages of the development of the repertoire of user knowledge. The A set and the PKE in different ways illustrate the development of user knowledge as learning progresses, although that knowledge appears to incorporate inconsistencies, and some acquired as well as original knowledge appears erroneous. There are also significant contrasts between the tests. Users can accurately perform tasks in the A set, where they cannot describe them in the PKE, or assign their menu items in the sorting task. This supports the argument that performance and non-performance tests do not access exactly the same user knowledge, or access the same knowledge differently (e.g. Ericsson & Simon, 1984).

Finally, one measure of the overlap between tests, is to rank the eight subjects from the 'best' to the 'worst' on each of the measures used, and compare the extent to which the relative ranks overlap. The rankings for selected measures are presented in table 2. The highest correlation between tests is between the questionnaire and the A set test ($r = .97$, $p < 0.01$). If these are tapping different knowledge, then this may seem surprising, but the questionnaire was designed to test points which arose during the A set test. It is also possible that although they test different knowledge, subjects who are good at one test are good at the other. The speed of keying in the data entry task, is the worst correlate with the other measures. However, since all the other measures sample learning, and the data entry task samples expertise on a manual task this is not surprising.

TABLE 2: Intersubject differences and variability. The A set performance test is reported by the percentage of tasks completed at the first attempt. The B set performance test, by the total number of errors made by subjects. The recall test, by the number of items correctly assigned. The questionnaire, by the number of questions correctly answered. The PKE test, by the number of true core claims produced. There are two measures derived from the data entry task: the time to complete simple manual tasks (a human simple speed measure) and a distance measure from the point of optimally efficient tradeoff for each subject (users 5 & 6 always chose one method, hence their tradeoff point is infinity).

Subjects	Ranks and Ranges							Mean
	Aset	Bset	Recall	Quest	PKE	Data Entry		Rank
	Tasks %1st AT	Number Errors	(No)	(No)	(claims)	Completion time (sec)	T/O point	
1	5	8(61)	6	5	6	3	3	5.1
2	6	5	8(7)	6	7	4	2	5.4
3	4	4	4	3.5	4	8(83)	4	4.5
4	3	2	5	3.5	3	2	5	3.4
5	7	6	7	7	8(22)	7	7.5	6.7
6	8(51%)	7	3	8(23)	5	6	7.5	6.3
7	2	3	2	1(33)	2	1(54)	1(0.6)	1.7
8	1(98%)	1(7)	1(30)	2	1(90)	5	6	2.4

Conclusion

No simple technique can be expected to adequately capture the complexity of using a computer system, although it may provide useful local information about a specific aspect of performance. However, a range of carefully related different measures can give sets of detailed information which provide converging evidence on which to base a more detailed conclusion. This paper has shown the application of such an approach in one study for which it has proved to be extremely successful in increasing understanding of how various aspects of the user interface affect both user performance and the underlying mental representation on which it is based.

The representation that appears to underly the learning captured by these tests is best described as a repertoire of knowledge fragments. This repertoire develops over the course of learning, and different measures and performance tasks sample different sets of this repertoire. This repertoire contains both 'correct' and 'false' knowledge about the system being considerd. Some of these false knowledge fragments are acquired early in performance, but many of them are acquired later. Errors in performance appear to arise when there is a conflict between elements in this repertoire, leaving an ambiguity in the creation of attempts to perform tasks and for user action. Under these conditions users are willing to endorse false statements about the system, and the inconsistency of the knowledge leads to confused and contradictory verbalisations.

References

Anderson, J.R. (1983). **The Architecture of Cognition**. Mass.: Harvard University Press.

Barnard, P., Wilson, M.D., & MacLean, A. (1986). The elicitation of system knowledge by picture probes. In **CHI '86: Human Factors in Computing Systems**, Boston, Mass.: ACM.

Card, S., Moran, T. & Newell, A. (1983). **The Psychology of Human-Computer Interaction**. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Ericsson, K.A. & Simon, H.A. (1984) **Protocol Analysis, Verbal Reports as Data**. Cambridge, Mass.: MIT Press.

Hammond, N., MacLean, A., Hinton, G., Long, J., Barnard, P. & Clark, I.A. (1983). **Novice use of interactive graph-plotting system**. IBM Hursley Human Factors Laboratory Technical Report HF083. Hursley, U.K.: IBM.

Lemmons, P.A. (1983) **Guided Tour of VisiOn**. BYTE, 8(6), 256-278.

MacLean, A., Barnard, P. & Hammond, N. (1984). Recall as an indicant of performance in interactive systems. In **INTERACT '84**, ed. B. Shackel. Amsterdam: North Holland.

MacLean, A., Barnard, P. & Wilson, M.D. (1985). Evaluating the human interface of a data entry system: user choice and performance measures yield different tradeoff functions. In **People and Computers: Designing the interface**, eds. P. Johnson & S. Cook. Cambridge, U.K.: CUP.

Suchman, L.A. (1985). **Plans and Situated Actions**. Xerox Palo Alto Research Report, ISL-6. Palo Alto, Ca.: Xerox.

Wilson, M.D., Barnard, P. & MacLean, A. (1985). **User Learning of Core Command Sequences in a Menu System**. IBM Hursley Human Factors Laboratory Technical Report, HF114, pgs 117. Hursley, U.K.: IBM.

Notes

VisiOn is a trademark of VisiCorp

[1] Only data from the first 9 picture pairs is presented.

LANGUAGES FOR USERS USED BOTH IN FAMILIARIZING THEM WITH COMPUTING SYSTEMS AND IN THEIR COMMUNICATION WITH THE SYSTEMS: four cases

Pekka Tyllilä
University of Tampere
P.O.Box 607
SF-33101 Tampere, Finland

Summary

The usual approach of the computing experts to office automation has been to formalize office functions (Barber et al. 1983). To this end the experts have attempted to learn from professionals in office what are office activities. Higgins and Safayeni (1984) have noted that the information about a particular organizational situation is distributed among the people who work there. Moreover, the amount of information on organizational activity is far greater than the amount of information on the technology for office automation decisions. Therefore office workers should learn about computing technologies in order to be able to adequately combine computing systems with their old organizational activities.

When professionals are introduced into development and use of computerized information systems there are two interfaces, on one hand, professional (end-user, user)/ systems (adp) expert interface and on the other hand the user-computer interface that play a central role. In this paper we present four cases of systems development with the users; our emphasis is in user learning. The purpose is to pick up the most important success factors of user teaching in these cases. The problems of the language used in learning situations are emphasized

1. THEORETICAL ASPECTS

Because our empirical approach is not the traditional experimental one but an actional one (action research) we shall consider theoretical and methodological problems more thoroughly.

As a reaction to scientific management and Taylorism there have been many approaches in job design (job enlargement, job enrichment, autonomous groups, socio-technical design...) which all have a common feature that they believe it is possible to fulfill two fundamentally different goals by reorganizing work: first, to increase the overall productivity and effectiveness of the work organiztion and second, to enhance the fulfilling of the goals related to better job content (for example job satisfaction). (Buchanan, 1979)

In our mind this has been the starting point of psychologigal job design theories, but our starting point is somewhat different: a kind of technological determinism. By this we mean that design and implementation of a computing system in an organization creates always necessarily a job design problem. In job design it is not only a question about increased human productivity or better job content but job design is a necessity caused by new

technical solutions in work. We have been conscious about this already before the first empirical research cases.

Because in our approach there is not any predefined set of accepted goals and values we really have to face the problem of freedom. We can not accept the conception of causality behind the ideal of controlled experiment as an overall methodological starting point. Since we have sarted to plan some concrete research cases dealing with the problems of user participation and user driven methods in systems development we have been familiar with a quite recent sociocybernetical theory that gives an exact mathematical articulation about the is - ought problem and about different conceptions of causality and research methodology which are very important general theoretical aspects in our approach. In the frame of this theory it is possible to conceptualize man and machines in an integrated way when building up theories dealing with the problems of interaction between technology and man in micro (job, task, operation) or in macro (job, organization, society) scale. We have build up our theoretical thinking mainly on sociocybernetics.

1.1. Sociocybernetics as a theoretical background in our empirical cases

In this problem setting the starting point of our theoretical thinking is two fold: first, analysis of roles of researchers and research objects in the study and second, consideration of basic characteristics of human beings as research objects.

Let us consider the first point. From two roles of the researcher in relation to research objects two interpretations of science are derived (Aulin, 1982): (A) the objectivistic one where the researcher is assumed to be a neutral observer and (B) the actional one where the researcher interacts with research objects.

Assumptions of type A (for example ideals of controlled experiments) are usually quite harmless. However, when we consider more deeply the relation between researcher and research object in empirical research methodology, we see that these kind of assumptions are wrong. the observations do not depend only on the objective properties of the research object but on the properties of the observing subject (plus equipment), too. If we encounter this problem of interaction between researcher and research object already in physics then it is more evident that this problem is inherent in behavioral and social sciences where the research objects are intentional human beings. The action research approach (Oquist, 1978, Susman and Evered, 1978) is based on subject-object interaction, and hence it belongs to type B. Rapoport (1970) defines it as follows: "Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework". Foster (1976) would supplement the definition with "and in the intention of the parties, although with different roles, to be involved in a change process of the system itself". We have used action research as an overall methodological approach in our empirical cases. We would not stress very much the role of mutually accepted ethical framework

- the minimal prerequisite needed is that the different interest groups (researchers, clerks, management, adp-professionals...) do not completely stop communicating with each others.

Now we shall go back to the basic characteristics of human beings as research objects. For a research methodology based on the ideals of controlled experiment two assumptions concerning research objects are made (consciously or not) (cf. Aulin, 1982): first, (a1) the researcher is observing, in terms of an observable z, an object whose initial state z(in) either automatically repeates itself, of can be repeated by the researcher, or the object itself can be replased by an identical object having the same initial state (repetition of the initial state); second, (a2) the researcher is able, by varying the initial state z(in) to control the final state of the same or some other observable z(out), concerning the object investigated (control of the final state).

We think that the assumptions (a1) and (a2) hold for human beings if the research problem concerns some proper subprocess of man (an objective aspect of man). But if man as a whole, his whole intellectual process, will and consciousness as charasteristics of man are emphasized in research then assumptions (a1) and (a2) do not hold.

Assumptions (a1) and (a2) refer to a deterministic (repetition of states) causal world. Hence, this kind of causal world is not sufficient when man as a whole is the research object. The indeterministic intentional aspect of man is in conflict with assumptions (a1) and (a2). In sociocybernetics a new concept of causality has been developed. It gives a solution to the coexistence of universal causality and freedom, a solution to the old philosophic dilemma between determinism and indeterminism.

The solution has been axiomatized (Aulin, 1982). The core of the axioms is in the following:

1. The cybernetic world of causality is state-defined. The past defines the present. Transition from one state to another is defined by a recursive function.

2. However, this does not imply determinism because the set of states in the causal world is expanding. The total causal world does never come to the same state again.

This kind of causal world does not involve only state-defined repetitive (deterministic) systems but also state-defined non-repetitive (indeterministic) systems. The latter systems are called in sociocybernetics self-steering systems. Based on the axioms of the causal world the concept of self-steering system is defined as follows: A causal system is self-steering at moment τ only if its state $x_\tau(t)$ when left after the moment τ to develope according to the own laws of the system, asymptotically approaches the same fixed trajectory of state-space, definable in terms of a non-constant and non-periodical goal function, whatever be the initial state of the system at the moment τ, provided that it is within the domain of self-steering of the system D_τ. The transition of states is defined as a causal state transition recursion.

Note that because our purpose is to study real human beings in their work, then it is relevant to say something about our ontological assumptions when we are applying sociocybernetic theory. We don`t claim that human beings are self-steering systems but in our mind all theories which have to deal with human values, free will, indeterminism... have to include at least as indeterministic systems as self-steering systems are. The advantage of sociocybernetic theory is the exactness of the mathematical language used in this theory.

When searching for suitable empirical research methdology with human beings as research objects we can consider the properties of self-steering systems: domain of self-steering Γ_τ, state defined causal recoursion φ and non-periodical, non-constant goal function g (t). Relaxing, one at a time, the properties of self-steering systems we shall see a whole family of causal systems emerge (Aulin, 1982). For our purposes it is enough to divide causal systems to two categories: (1) Self-steering systems (for example man as a whole "is" of this type) and (2) other causal systems from mechanistic to adaptive.

Mechanistic systems are classical examples of research objects which are especially suitable for controlled experiments. Other systems in class involve an increasing tendency towards behaviour guided by internal laws of the system. Hence, it is more difficult to achieve deeper understanding of the system using controlled experiments or other methods with the same ideals. However, it is in principle possible. In system class 1 the methodological principles of controlled experiment are excluded by definition. System class 1 does not obey assumptions (a1) and (a2).

In conclusion, we have two cases when man is a research object: we can study either a certain aspect of man or man as a whole. In the former case, we can use methods based on the ideals of controlled experiments and build up theories based on those methodological principles, but in the latter case this is not allowed.

We shall consider man at work, carrying out different tasks and we shall consider the learning process (languages) related to that. We will use and further elaborate the classification by Järvinen (1980).

Vihmalo (1982) has compared various approaches to psychological work analysis. She found that Miller (1967), Annett et al. (1967, 1971) and Hacker (1980) emphasize the role of goals in work analysis. They also share the theoretical concept of an action program, which is necessary for achieving the goal. The action program directs and controls human behaviour. All the researchers above have derived the term `action program` from the concept of plan by Miller, Galanter and Pribram (1960), who define it in the following way: "A plan is any hierarchical process in the organism that can control the order in which a sequence of operations is to be performaed". The practical utility to be derived from diagnosing a plan is evident in work instruction and training: by improving the plan work performance is improved

Plans are studied most throughly by Hacker (1980), who

distinguishes three kinds of plan: movement-pattern, schema and strategy. The highest level plans are strategies, which are created before the practical accomplishment of a task. Strategies quide the intellectual regulation needed in problem-solving and planning tasks.

When considering a certain difficult task we can distinquish two problems: the object problem and the processing problem (cf. Eloranta 1974). The former means the operations necessary to perform the task, and the latter either building up a plan or applying a plan built earlier. The plan contains a set of rules for solving an object problem. According to the ideas presented above, a certain task can be classified in one of three categories: (I) a problem-solving task or, (II) an application task or, (II) a routine task, depending on whether it requires (I) building a plan, (II) applying a ready-made plan, or (III) a minimal psychical effort, to perform a task.

In large our research object is man carrying out tasks of different categories (I,II and III). Especially we are interested in languages and user learning related to computer systems use and development. The critical question is wether the emphasis is on "carrying out a specific task" or on "learning a specific language" or on "man as a whole".

If the emphasis in research is put on the former aspect then research methodology certainly can be based on causality concept of controlled experiments and assumptions (a1) and (a2) (see p. 3) are correct. Even the abstract process of solving a set of problem solving tasks is an adaptable system. But if the emphasis is put on the later aspect (man as a whole - man as a learning problem solver) we are dealing with self-steering systems in our research. Assumptions (a1) and (a2) are not correct and we have to choose a methodology of type B.

In our research cases we have choosen a methodology of type B (see p. 2). We think that in real life in work the change process caused by the use of modern computer technology is not typically a simple adaptation process from the users` point of view. It is a large problem setting and it is related to the general life situation of the user. For example former experiences of radical changes in work, economical background - home, fears of loosing the job, educational background, fears of being not able to learn the new needed skills related to the technology interfere heavily to the strict adaptation process of learning the new needed skills. (Hiltunen, 1986)

We think that the aspect of carrying out tasks can be emphasized for category III in a way that assumptions (a1) and (a2) hold. This is based on the belief that an experienced user always performs a certain routine task nearly in the same way or at least the result (the final state) and the prerequisites (the initial state) are the same. The same state of the interactive system (routine task and the user performing the task) can be repeated and the final state controlled by the initial state. In this kind of research the assumption on the role behaviour of man has been made. Man has the role of carrying out routine tasks excluding his intellectual process as a whole. The approach can be criticied by claiming that ral human beings do not obey such

strict roles. However, for methodological reasons the assumption of role behaviour is very useful and there certainly are adaptive roles according to which people are behaving, in our context of research we are not denying the assumption of role behaviour with respect to routine tasks.

The situation is quite different when task category I is concerned. The solution of the processing problem as such is an adaptable system but the solver of the problem does not have clear ideas about the plan for problem solution (prerequisites for solution and the `algorithm` giving the solution according to proper initial state), and even the goal (proper solution of the problem) can be unclear. As stated earlier many aspects of life are interfering with this kind of situation. Because of this man as whole especially the intellectual process interferes with the situation when the problem solver is searching for the solution. The assumption of role behaviour is incorrect and assumptions (a1) and (a2) don`t hold. It can be claimed with good reasons that the less there is role behaviour the more probable is the solution of the problem solving task. The system composed of a human problem solver and a problem solving task (especially in complex and new environment as in use and development of new computer applications) is self-steering although the goal of the original problem solution is constant. The action research approach instead of the ideals of controlled experiment might be used for category I, because its basic assumptions are in concordance both with the interpretation B and with the concept of self-steering systems (cf. Tyllilä, 1981).

Task category II is not self-evident as category I. The problem solver is still learning when he is applying the plan. The situation is open for his intellectual process as a whole. The assumption of strict role behaviour is not correct. Hence we can follow a similar reasoning as for category I.

1.2. Theoretical conclusions based on the sociocybernetic analysis

The analysis of categories I, II and III was based on consideration of human action as a self-steering system. Aulin (1982) has analyzed self-steering systems and suitable research approaches for them as follows: Self- steering governs the action of man as a whole, as an active and creative being capable of expanding state-development producing unique states that did not previously exist. The core of human self-steering is the intellectual process . This process cannot be adequately studied by experimental methods because of the manipulation of states inherent in all the so-called behavioral research involving a stimulus-responce technique. It is a notable property of self-steering that the same state never returns in a self-steering system. This makes all that is essential in self-steering inaccessible to experimental methods, and to all theoretical models based on an analogy and thus requiring for their verification a direct confrontation with the real, modelled object.

In conclusion, a research methodology based on the methodological ideals of controlled experiments is not suitable in a research situation where in the development and use of computer based

systems in practice the users are learning new languages for understanding the system and for mastering their new work situation. Only some aspects in this kind of situation can be studied by experimental methods but the situation as a whole is suitable only for action research or for related methods.

The title of section was "theoretical aspects" but we have discussed mainly about methodological points. There is a clear reasoning behind this kind of approach. We have denied the use of traditional experimental methods in research situations as our cases, at the same time we have denied the possibility of explanative theories based on experimental methods. When our research objects are considered to be unique in the sense that they produce always new states we can not generalize in the traditional sense from the experiences of our research cases. This is the epistemological problem in action research. However, if we think daFinetti`s epistemological theorem about the cumulation of empirical knowledge (Aulin, 1977, 1982, Savage, 1971), there are no reasons against the cumulation of empirical knowledge by using action research, it is only different. General empirical knowledge proceeds from case to case and the context of produced knowledge is very central because it helps the users of the knowledge to evaluate its (general) value in a new problem setting. Because of this epistemological difference of experimental methods and action research it is a typical claim that action research is a proper methodology in new and complex research situations and that after some initial results produced by action research it is possible to proceed to experimental methods and build up explanative theories. We think this is only a partial truth, if we consider that man as a whole is a self-steering system then experimental methods can produce knowledge only about some aspects of man. Our experiences suggest that the use of action research is very useful to "catch" the whole in a concrete specific organizational problem setting.

2. FOUR EMPIRICAL CASES

More detailed information about the cases can be found in the references given in each subsection.

2.1. First case: adjustment of a ready made software in a bureau of social care

The first case concerns a computing system supporting some functions in a department of social care (Järvinen et.al.,1982). The system was developed in Helsinki and was taken into use in the city of Tampere. The supplier gave several days of instruction during the change-over period. During this training the programs were introduced to professionals (users/end users) by presenting them as JSP diagrams (Jackson 1975) and using the language and terminology of systems and computing experts where top down decomposition has a central role. The learning situation was organized as the professionals could be steered from outside. Learning results were poor and training evoked fears against the system. In programs for data entry the fill-in-the-blank technique was used. To this end the programs were inflexible and they totally determined the order of data input.

However, in the middle of the change over period the system was tested by taking 100 actual 'records' (data referring to 100 families which were clients of the department). The professionals then learned fast to understand and to use the system. When the actual records were fed into the system (it was done by the professionals and by some auxiliary clerks) it was necessary to start to use the language of the specific functions which were supported by the system. We think that in this phase the known conceptual content and a concrete learning situation were the success factors which helped the professionals to proceed and master their learning.

2.2. Second case: a programming course for users/end users

The second case (Järvinen 1981) is direct continuation of the first one in the same bureau. We as researchers arranged a short programming course for professionals (seven lessons, i.e. Friday afternoon every second week). The goal was to teach the programming language (restricted COBOL) used in the system of the professionals. We applied the bottom-up idea proposed by Holt et. al. (1977). In addition we prepared two manuals during the course. One was COBOL manual written in Finnish and the other contained principles of computation, program design and testing, and well-documented examples. Already during the first two lessons the participants programmed and run simple programs. The approach was very concrete.

However, after the first three lessons the motivation to learning decreased. We discussed the problem. The professionals gave us an idea: "Please teach us to understand the programs which we have in daily use". In the rest of the course this idea was followed. Their motivation to learning again increased. Six of the eight professionals passed a program comprehension test. In the daily work of the professionals the course implied better communication between the professionals and computing specialists responsible for the system development and maintenance. Again the known conceptual content as a starting point was a success factor in learning. The poor syntax of the used programming language could not overrule this. Another point was that our teaching situation was organized in an interactional way, we together with the professionals could almost immediately evaluate the given feedback and change the planned content of the course.

2.3. Third case: a participative introductory systems development course for users/end users

The third case was an introductory systems development course for officials and clerks of the metalworkers' union (Järvinen and Tyllilä, 1983). The group was very heterogenious. - The problem of the union was: how to utilize computing technology in administration of the union, or how to develop information systems for their purposes. The union wanted to proceed relatively independent in this question. Therefore we organized the systems development course for them. The initial plans for the course were developed in an interactive way with the representatives of the union.

Two principles were applied in course development: (1) The first steps in systems development: feasibility study, information

analysis, and the purchase of software and equipment, become important objects of learning because the organization had decided not to hire own computing personnel; (2) Familiarity of subjects to be taught should be practiced.

There were three main areas in the final course structure: first, a very abstract lecture describing the overall technological perspectives; second, analysis of own work and third, software purchasing process.

The first area was picked up because we thought that union officials were very interested about the overall possibilities and future consequences of information technology and we did not get disappointed. The clerks were interested from the second area. For the third part we had an offer of computing equipment for the purposes of the union as a teaching material. Again it was clear that the every day language (terminology and concepts) of computer professionals is a notable obstacle for user learning. At the end of three days lessons and exercises the course was evaluated by a semi-structured questionnaire. Learning results were remarkable. Later as an implication of this course a form for analysis of own job was planned in the headquarters of the union. The relations between us and the union become permanent.

This case shows that both abstract lectures (technological perspectives) and concrete practicing (analysis of own work) can give positive results when the participants are properly familiar with the conceptual content to be taught.

2.4. Fourth case: an introductory course in office automation for university clerks

The course consisted of three main areas: first, basic concepts of computing (computer, program, programming, work station and communication); second, office automation applications (demonstrated in practice) and third, experiences from office automation in the organization, and changes in work. The main structure was planned in a working group with reprsentatives of clerks, computing experts, researchers and administration. The content of the third part was planned in a more interactive way in a small group. The main result of this small group (1-2 social scientists, the author and one typist) was a two hours video describing different word processing systems which are used in the university. No one had former experience of producing video tapes, the typist was a user of a word processing system. It took about ten hous from the group to produce the video. Only in the final editing professional help was used.

The first and the third part were two extremes from the viewpoint of user language and learning. The first part consisted of lectures with 'technical', 'computer scientific' content and the third part consisted of both the video and lectures concerning office work and automation. The course was carefully evaluated with a questionnaire. The third part was very positively accepted while the first part was widely criticized.

Again the used language had an important role. The language and concepts used in the video and in the discussions dealing with

the problems and questions demonstrated in the video was known to the course participants. It offered a motivating starting point for learning. The lectures given by the adp professional were too far from the thinking structures of the participants. In the evaluation responses the participants still used those thinking structures which were commonly used in their work and not those given by the lecturer although it was allowed to use the lecture notes when ansvering.

2.5. Summary of the results

Partially by chance and partially consciously we have followed priciples which also have been evaluated to be succesful in adult learning in other studies. For example Jenny Rogers (1983) emphasizes the following principles of adult learning: first, working from the known to the unknown (in our cases for example from own work to computing); second, learning through success; and third learning by doing (practical excercices which have a strong relation to jobs of the users).

In conclusion we think that these four cases demonstrate that:

1. Abstract lectures concerning computing in office work implicate low interest and poor learning results. Especially this is true if the listeners are not familiar with the concepts used in lectures. The relation between learning new skills and knowledge, and their use at work stays unclear to listeners.

2. Concrete learning situations implicate higher interest and better learning results.

3. The syntax of the language used in communication with users is not as important as the conceptual content of the communication. For example we have got good learning results with syntactically machine oriented tools when the conceptual content has been familiar to users.

REFERENCES:

J. Annett and K.D. Duncan (1967), Task analysis and training design, Occupational Psychology 41, 211-221.
J. Annett et. al. (971), Task analysis, Training information, No 6, 1-23.
A. Aulin-Ahmavaara (1977), A general theory of acts, with application to distinction between rational and irrational `social cognition`, Zeitschrift fur allgemeine Wissenschaftstheorie 8, No 2, 191-220.
A. Aulin (1982), The cybernetic laws of social progress, Pergamon, London.
G. Barber, P. de Jong and C. Hewitt (1983), Semantic support for work in organizations (in R.E.A. Mason (ed.), Information Processing 83, North-Holland, Amsterdam), 561-566.
D.A. Buchanan (1979), The Development of Job Design Theories and Techniques, Saxon House, England.
K.T. Eloranta (1974), Heuristics and heuristicity, Academic Dissertation, University of Tampere (in Finnish)
P.M. Foster (1976), The theory and practice of action research in work organizations (in A.W. Clark, Experimenting with organizational life, action research approach, Plenum, New York),

59-76.
W. Hacker (1980), Allgemeine Arbeits- und Ingenieurpsychologie, VEB Deutscher Verlag der Wissenschaften, Berlin.
C.A. Higgins and F.R. Safayeni (1984), A critical appraisal of task taxonomies as a tool for studying office activities, ACM Transactions on Office Information Systems 2, No 4, 331-339.
R.C. Holt, D.E. Wortman, D.T. Barnard and J.R. Cordy (1977), SP/k: A system for teaching computer programming, Comm. ACM 20, No 5, 301-309.
M.A. Jackson (1975), Principles of program design, Academic Press, London.
P. Järvinen (1981), The user as a learning component in a data system during the change-over and use periods, ACM SIGCPR's (Special Interest Group of Computer Personnel Research) 18th Annual Conference at Georgetown University (Washington D.C.) 39-63.
P. Järvinen, et. al. (1982), Analysis and design of jobs in a man-computer system, Dept. of Math. Sciences, University of Tampere, Report A74.
P. Järvinen and P. Tyllilä (1983), Introductory systemeering course for users: The Murikka case (in Nurminen, M., Gaupholm, H., Report of the sixth Scandinavian research seminar on systemeering, Institute for Information Science, University of Bergen, Norway), 150-183.
G.A. Miller, E. Galanter and K.H. Pribram (1960), Plans and the structure of behavior, Holt & Co, New York.
R.B. Miller (1967), Task taxonomy: Science or technology?, Ergonomics 10, 167-176.
P. Oquist (1978), The epistemology of action research, Acta Sociologica 21, No 2, 143-163.
R.N. Rapoport (1970), Three dilemmas in action research, Human Relations 23, No 6, 499-513.
J. Rogers (1983), Can old dogs learn new trics? (in Rijnsdorp and Plomp (eds.), Training for tomorrow, IFAC/IFIP Conference in Leiden June 7-10, 1983, 28-32.
L. Savage (1971), The Foundations of Statistics, John Wiley & Sons, New York.
G.I. Susman and R.D. Evered (1978), An assessment of the scientific merits of action research, Administrative Science Quarterly 23, 582-603.
P. Tyllilä (1981), On motivation and problems of action research especially in information system development and use (in Kerola, P. and Koskela, E. (eds.), Report of the fourth Scandinavian research seminar on systemeeering, Institute of Data Processing Science, University of Oulu), 71-76.
A. Vihmalo (1982), The psychological analysis of work: A study of different approaches (in Järvinen, P. et. al., (eds.), Analysis and design of jobs in a man-computer system, Dept. of Math. Sciences, University of Tampere, Report A74), 13-23.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique