European Association of Cognitive Ergonomics

ECCE-5

FIFTH EUROPEAN CONFERENCE
ON COGNITIVE ERGONOMICS

PROCEEDINGS

Urbino, Italy, September 3 - 6, 1990

Università degli Studi di Siena
C.N.R. - Istituto di Psicologia
ENI - OLIVETTI

GOLEM

European Association of Cognitive Ergonomics

ECCE-5

FIFTH EUROPEAN CONFERENCE ON COGNITIVE ERGONOMICS

PROCEEDINGS

Urbino, Italy, September 3 - 6, 1990

Organized by:
Università degli Studi di Siena
C.N.R. - Istituto di Psicologia
ENI - OLIVETTI - GOLEM

ECCE 1990
URBINO

CONTENTS

Task Analysis

Gilmore, D.
Models of debugging 3

Johnson, H., Johnson, P.
Identifying and meeting designers' requirements for task analysis tools 17

Kallela, J.
Task analysis applied to control system design 29

Cognitive Approaches to Process Control

Cacciabue, P.C., Decortis, F., Drozdowicz, B.
Simulation of problem solving deviations by a cognitive model 41

Hoc, J.M., Samurcay, R.
Modelling process operator knowledge for the design of computer support to decision making 55

Mestre, D., Savoyant, Péruch, P., Pailhous, J.
In search of realistic visualization for human-machine interaction in spatially oriented process control 67

Rizzo, A., Bagnara, S.
Collaborative interface: A PDP approach 79

Analysis of Learning, Thinking and Social Processes in HCI

Briggs, P., Lambourne, A.
Menu trees as memory aids 95

Brown, G.M.
The use of frequency and the development of knowledge 107

Dewier, A., Karnas, G.
Type of command used during a semantic categorization task on PC: The role of subjects' level of experience and task characteristics 119

Linde, L., Bergstrom, M.
Is ability to understand a query language facilitated by training pertaining to the "deep structure" of the search task? 129

Slack, J.M., Conati, C.
A framework for exploring exploration 147

Sutcliffe, A., Maiden, N.
Cognitive studies in software engineering 159

Visciola, M., Armando, A., Bagnara, S.
On-line assistance in flight simulation 173

User Models

Allwood, C.M., Bjorhag, C.G.
Training of Pascal novices' error handling ability 187

Booth, P.A., Gray, J.
Approaches to error analysis in human-computer interaction 199

Neal, L.
Predictive tasks to build user models 215

Sasse, M.A.
Users' models of computer systems 229

Computer Supported Cooperation

Bignoli, C., De Michelis, G.
Coordination through multiple media 243

Dell'Olmo, P., Nardelli, E., Talamo, M., Vocca, P.
A model of visual language for cooperative problem solving 255

Esgate, A., Whitefield, A., Life, A.
Developing usability integration principles for the design of IBCN systems 267

Rogalski, J.
Operative communication and decision making in emergency management 279

Schal, T., Zeller, B.
A methodological approach to computer supported cooperative work 291

HCI Design and Evaluation Methods

Allamanno, N., Perotto, P.P.
A knowledge based approach to support the product offering process 307

Carenini, G., Ponzi, M., Stock, O.
Combining natural language and hypermedia as new means for information access 315

Darses, F.
An assessment of the constraint satisfaction approach for design: A psychological investigation 325

Delogu, C., Paoloni, P., Pocci, P.
Quality evaluation experiments of Italian speech synthesizers 337

Kuutti, K.
The activity theory as an alternative framework for HCI research and design 347

Lim, K.Y., Long, J.B., Silcock, N.
Motivation, research management and a conception for structured integration of human factors with system development methods: An illustration using the Jackson system development method 359

Parlangeli, O.
Software usability: A user-centered process of evaluation 375

Petre, M., Green, T.R.G.
Is graphical notation really superior to text, or just different? Some claims by logic designers about graphics in notation 387

Pitariu, H., Valcea, C., Rapauzu, C., Lapadatu, L.
Using CLG and ETAG in designing software systems possible extensions -Steps toward dynamic interfaces- 409

Poitrenaud, S., Richard, J.F., Tijus, C.A.
An object-oriented semantic description of procedures for evaluation of interfaces 431

Sutcliffe, A.
Computational models for visual communication in multi-media discourse 441

van der Veer, G.C., Brazier, F.M.T.
How adaptive should a user interface be? 461

Task Analysis

Models of Debugging

David J Gilmore
Psychology Dept
Nottingham University
Nottingham, NG7 2RD, UK
Email: dg@psyc.nott.ac.uk

Abstract

This paper proposes a view of computer program debugging, which tackles some of the simplifying short-comings of existing models. The paper begins by reviewing some of the existing models of debugging and their assumptions, before looking in more detail at one of the dominant paradigms, that of predicting bug detection success. A re-analysis of the bug detection data from Gilmore and Green (1988) provides evidence that the assumptions of these models are not valid. The important part of this result is the realisation that these assumptions have been derived from a view of debugging as fault diagnosis, rather than as a critical component of design. In conclusion, the paper describes the important features of debugging as a design activity, before outlining some predictions and implications which can be derived from the model.

Introduction

To many it may seem surprising to ask the question 'why study debugging?', since it is obviously a central skill of computer programming, especially if maintenance is included within the definition of debugging. Yet, despite the fact that debugging is a skilled combination of both program understanding and program coding, it has been remarkably ignored by those studying psychological factors in programming. An examination of Curtis's comprehensive review of different paradigms within the psychology of programming (Curtis, 1988) reveals not a single mention of debugging.

Although it could be argued that debugging is simply a combination of comprehension and coding and, therefore, does not deserve a separate research effort, this argument is not persuasive for those concerned with software reliability and the provision of debugging tools for professional programmers. Improving comprehension and coding tools may be important to improving debugging performance, but it cannot be the sole answer.

A number of papers on debugging do exist, as do a few models of debugging processes. However, many of these only provide an analysis of the act of bug location, without discussing the wider task context. Given the importance of debugging activities, it is important to clarify the validity of the conclusions drawn in these studies, since they could easily lead to the development of debugging tools which do not help, or which do not address the real problem. Also, some studies have used debugging simply as a tool to investigate comprehension, without actually discussing the nature of the relationship between debugging and comprehension (eg. Gilmore and Green, 1988).

Figure 1 shows, schematically, a common approach to the context in which bug location is assumed to occur. Bug location is assumed to be a process of inspecting a mental representation of the program code, which has been derived from the fixed entities - the code, the

problem statement and real-world knowledge. Surprisingly, none of these is considered to have a direct impact on the bug location process. Figure 2 gives an example of a particular version of this model, from Katz and Anderson (1988). The schematic model, which can be termed the *fault diagnosis* model, embodies three assumptions, each of which will be discussed briefly in the following sections:-

a. Comprehension always precedes debugging
b. There is a single bug location process which is to observed.
c. The problem is not a problem.

Comprehension precedes Debugging

The task of debugging has been used to provide a window on the nature of the programmer's mental representation of the code. In such situations the conclusions drawn are only valid if one assumes that the comprehension process is complete before debugging occurs. An example of a study which makes this assumption, albeit implicitly, is that by Gugerty and Olson (1986), who conducted observational studies of novices and experts debugging programs written both in LOGO and in Pascal.

They found that experts were quicker and more successful than novices, and that they tested fewer hypotheses about the code, whilst novices frequently added more bugs to the program. This difference was attributed to the experts comprehending the program more successfully than did the novices. However, Gugerty and Olson observed that prior to forming any hypotheses, novices and experts engaged in similar proportions of activities such as reading the code and reading the problem statement (though experts were faster). From this they conclude that experts and novices use the same comprehension processes, but that experts are more successful and acquire a better mental representation of the code. The success of the experts at debugging is not attributed to better debugging skills, but to better comprehension, which is then followed by the application of the same debugging processes, but in relation to a better knowledge base for the experts than for the novices.

Thus, although Gugerty & Olson conclude that the differences in debugging success are due to differences in comprehension ability of two groups this depends upon the assumption that the process of comprehension is the same for both groups and that this process precedes debugging.

An explicit example of the assumption that comprehension is separate and precedes debugging is the study of LISP debugging by Katz & Anderson (1988), who write that

> "when debugging, one must
>
> a) test the program .....
> b) locate the erroneous line ....
> c) rewrite the buggy code.
>
> Also, if the person did not write the code originally (or wrote it a long time ago), that person will probably need to first comprehend the program." (ibid, p353).

A Single Debugging Process

Alongside the assumption that comprehension is separate and comes first, it is also commonly assumed that there is only one debugging process. Given this assumption we can infer that when we observe differences in debugging success, we can conclude that they are due to differences elsewhere - either in comprehension or knowledge structures for instance.

An interesting example of work which assumes that there ought only to be one debugging process is Kessler and Anderson (1986), who provide a four-stage model of novice LISP debugging; the stages being code comprehension, bug detection, bug localization and bug re-

pair. Kessler and Anderson collected protocols from novice LISP programmers and claim a good fit between their model of debugging and the protocols. However, this ignores the one subject who generated the correct code mentally and compared this with the actual code. Kessler and Anderson disregard this subject arguing that this strategy could only work for very simple problems. The fact that their model only allowed one debugging process prevented them appreciating the importance of this non-comprehension-based exception.

Another example of work assuming a single debugging process is that of Iris Vessey. Vessey (1985) proposed a model which assumed that novices and experts engage in the same process, but with different knowledge structures available to them. Her analysis of debugging protocols showed novices flitting between different stages in the model of debugging, whilst experts moved between stages quite rarely. Thus, the experts performed smoothly whilst the novices were erratic and error-prone. However, this conclusion depends upon the assumption that the novices were trying to perform the same process as the experts. If they were actually using a quite different debugging strategy then they might appear to be behaving erratically, when in fact they were competently pursuing an inefficient and error-prone strategy.

Vessey (1989) adopts the model of debugging used by Jeffries et al (1981) in one of the first studies of debugging. This model is based on a Kintsch's theory of text comprehension, based on an analysis of the propositions, During comprehension the propositions are chained together by links in the operands to form the program's microstructure. The meaning can then be extracted by re-forming this into a macrostructure, which represents the semantic structure of the program. The mental representation of a program is assumed to be both the micro- and macro- structures. Both structures are arranged hierarchically.

Such a model predicts that processing information which is high in the hierarchy will be easier than processing that which is deep down in the structure, except during the comprehension process, when the lower details may be processed first. Thus, Vessey predicted that bugs which are high in the propositional structure should be detected more easily than bugs deep in the hierarchy. Note that this argument requires assumption (a): that the comprehension process is complete before debugging occurs.

These arguments do not inform us how the bugs are detected, simply which bugs should be detected most easily. Without a process model it is difficult to be certain, but Vessey's arguments would seem to imply that the debugging process is the same for all contexts, since depth in the propositional structure is presented as the key factor affecting the detection rate.

However, Vessey's data did provide support for her model and it is necessary, therefore, to see whether other interpretations are possible. The main weakness in the conclusions is that they are based on an analysis of only one bug, occurring in different places. Not only does this lead to problems of generalization, but it is not clear that the same bugs occurring in different locations can really be called 'the same'. Control-flow bugs may seem to be easily equated, but semantic and plan-bugs cannot be equivalent unless at the same point in the program. For this reason it is not possible to conclude that the level of the bug in the two structures studied by Vessey was the only difference between the programs used. One large difference, mentioned but not discussed by Vessey is the fact that the location of the bug had different effects on the output. This difference was so great that even after considerable study, some subjects in one of the four conditions denied there was any difference between the actual output and that desired.

The potential for external factors such as the output, or maybe the problem statement to have an effect at the debugging level is something which is omitted from the schematic view described above. But there is some evidence which suggests that the programmer's understanding of the actual program be important.

The Problem Statement

Katz and Anderson (1988) describe a large number of LISP errors and look at the frequency with which novices repeat similar errors across a variety of problems. Regardless of any problems associated with identifying the 'same' errors across different contexts, their conclusions critically depend upon the error generation and detection processes being independent of the actual problem.

Thus, because the 'same operations' are performed erroneously in some contexts, but correctly in others, they conclude that the novices are only making 'slips' rather than applying misconceptions about how LISP works. However, the 'same operations' are defined according to the correct implementation of the program and, thus, students who misunderstand the problem will generate an error, which is neither a slip nor a misconception about LISP. This can be seen in some of the example errors provided by Katz and Anderson.

The distinction made by Wertz (1982) between *conceptual* and *teleological* bugs is useful here, since it clarifies the role of the problem in understanding bug generation and detection. Conceptual bugs are bugs arising from discrepancies between actual behaviour and required behaviour (cf. specification), whereas teleological bugs are discrepancies between actual behaviour and that intended by the programmer. In the former case the program contains a *traditional* error, but in the latter the program may be syntactically and semantically correct, the error arising because the program solves the wrong problem.

If programming behaviour reliably generated *either* conceptual *or* teleological bugs then understanding debugging might not be too complicated, but the crux of the problem is the fact that they can co-occur. A simple conceptual error, compounded by a teleological error may lead to a program whose behaviour is extremely complex and whose repair is very difficult. A further important point made by Waddington and Henry (1990) is that, for programmers who didn't write the original program, all bugs are conceptual, unless one draws inferences about the intentions of the original programmer - a process usually omitted from models of program comprehension and debugging.

Summary

The models of debugging commonly used are all simplistic and avoid the distinction between comprehension and debugging. In general the research takes a *fault-diagnosis* approach to debugging, in which one simply detects a discrepancy between *what should be* and *what is*, before correcting it. To enable the study of debugging separately from comprehension and coding it is necessary to make these three assumptions about the nature and ordering of comprehension and debugging.

A Re-analysis of Old Data

Introduction

With these considerations in mind, and given some of the problems in the data collected by Vessey, I re-examined the data from Gilmore and Green (1988), in which 40 subjects debugged 10 versions of each of two programs, with each program containing 2 bugs. In the analysis reported be Gilmore and Green the emphasis was on experimental manipulations of presentation format and no attention was paid to the bugs themselves. In the analysis to be reported here, the opposite is true, since here I shall analyse the data by bugs, rather than by subject groups. The analysis to be described will apply multiple regression techniques, using bug detection rate as the dependent variable.

If debugging can be characterised as fault diagnosis, with the programmer first understanding the program and then searching for discrepancies, then we would expect to be able to predict detection rates on the basis of *depth* within the comprehension structure. If, however, debugging is a design activity then we would expect bug detection rates to be specific to their particular context and the programmer's chosen strategy.

The Programs and Errors

The programs used in the study were all attempts at the classic 'rainfall' problem used in numerous Yale studies of programming skills. The bugs introduced into these programs were adapted from the Yale Bug Catalogue (Johnson et al, 1983), but were classified into four types (giving rise to three independent variables in the regression:-

surface: typing errors and suchlike

control: errors in the control structure, which could be termed syntactic (omissions or ill-formed statements)

plan: errors in the plan structure of the program, where the surrounding control structures are correct (omitted update of count variable)

interactions: errors which arise from a combination of control errors and plan errors (eg. Initialisations in the wrong place)

This classification was chosen for its correspondence with the presentation manipulation and for a behavioural, rather than symptomatic description (see Gilmore and Green for more information). Further experiments (Davies, 1990) have supported the usefulness of this classification, though I would not wish to claim that this is the only possible classification of these errors into different types* .

Measures of depth

For the reanalysis, a measure of the depth of each bug was necessary. Taking Vessey's results into account, and the results of Gilmore and Green (1988) I adopted three measures of depth, or location (ie. three more independent variables), each representing depth within a different structure:-

1) **serial structure**: As the name implies this was simply the serial position of the bug in the code. It was measured by taking the line number of the bug. Since the two programs were of comparable length, no adjustment was made for program length (ie. the proportional position in the code).
2) **propositional structure:** This term is used in the sense used by Vessey, where it referred simply to the depth of the bug in the control structure of the program. It should be noted that this is not analogous to Kintsch's propositional structure which captures the importance of each proposition to the meaning of the text. Propositional depth was measured by counting +1 for each new control structure encountered and -1 for the end of each control structure.
3) **goal structure:** This is closer to Kintsch's propositional analysis and is based on the Yale analysis of programming goals and plan structures. The analysis was performed with reference to the Yale goal structures for comparable problems, but it was not exactly equivalent. There are probably numerous different goal structures for these problems, and the different programs used in this study might reflect some of these

---

* Other differences between these bugs, such as their effect on the output of the program (eg. runtime errors or simply an incorrect result) or the distinctiveness of the resulting statement (eg omissions or the use of the wrong operator in an assignment) have not yet been addressed, even though they may be major contributors to bug detection rate.

differences. However, I believe that these differences are small and that it is not, therefore, worth describing the goal structures for each buggy problem.

In order to ensure that these are three different structures, the correlations between the depth of each errors in each pair of structures was calculated. The largest correlation occurred, somewhat surprisingly between depth in the serial structure and depth in the propositional structure ($r = 0.24$).

Task and Method

In the original experiment subjects were given 10 buggy versions of programs for each of two problems. Each buggy program contained 2 bugs, giving a total of 40 bugs overall, ten of each of the four types. The subjects (40 experienced Pascal programmers) were given a sample correct program for each problem, in an attempt to ensure that they fully understood the problem which the buggy programs were supposed to solve. They were told that their task was to mark and briefly describe any errors which they detected in each of the 10 programs for each problem, but that they were not to try and correct the program. They were given limited time to study each program and the analysis here only examines detection rate (ie the number of subjects who detected the bug), and not the time to detection.

Results

Multiple regression was performed on the data, using detection rate as the dependent variable and using bug type and bug depth as 6 independent variables. This complete model (Model a) accounted for 36% of the variance ($F_{6,33} = 4.69$, $p = 0.001$). When the variables coding bug type were dropped from the model (Model b), this changed to 5% of the variance (a significant change, $F_{3,36} = 6.77$, $p = 0.001$). By contrast, when the depth measures were omitted, the model (Model c) accounted for 37% of the variance (a non-significant change). This analysis is summarised in Table 1, from which one can infer that bug type is the most important factor influencing detection, with the location of the bug in the program making no contribution.

Table 1: Summary of regression analysis without interactions.

Model	R^2	Adj R^2	F	p	R^2 Ch. from a	F ch.	p ch.
a : 6 IVs - serial depth, propnl depth, goal depth, bug type (3 dummy variables)	.46	.36	4.69	.001			
b : 3 IVs - serial depth, propnl depth, goal depth	.13	.05	1.76	.172	-.33	6.77	.001
c : 3 IVs - bug type (3 dummy variables)	.42	.37	8.62	.000	-.04	.86	.47

However, given our original results concerning the improved detection of bugs when perceptual cues highlighted the relevant structure it seemed sensible to look at the interaction terms in this analysis (ie. the product of location and bug type for each location and each bug type). Thus, a second analysis was performed for the purpose of clarifying the role of depth for each bug type separately. For this analysis, there were 12 predictors in all - the three original measures of depth, plus the nine interaction terms. This model (Model a) was able to account for 49% of the variance ($F_{12,27} = 4.1$, $p=0.001$). From this model the three critical interactions (goal depth x plan bugs, propositional depth x interaction bugs and serial position x control flow bugs) were omitted, producing a model (b) which accounted for 23% of the

variance, which is a significant change ($F_{3,30} = 5.98$, $p < 0.005$). Although Model b produced a significant regression equation, none of the individual coefficients had a significance less than 0.1. When the other 6 interaction terms were omitted instead, the model (c) was still able to account for 44% of the variance (a non-significant change - $F_{6,33} = 1.5$, $p = 0.2$). These results are summarised in Table 2.

In Model c the coefficients for each of the three interaction terms were positive, indicating that the bugs were easier to find when they were deeper in the structures, which is in contradiction with the usual predictions in such experiments.

Table 2: Summary of regression analysis including interactions.

Model	R^2	Adj R^2	F	p	R^2 Ch. from a	F ch.	p ch.
a : 12 IVs - serial depth, propnl depth, goal depth, interaction terms (9 variables).	.65	.49	4.1	.001			
b : 9 IVs - serial depth, propnl depth, goal depth, 6 interaction terms (excl. goal x plan, prop x intern, serial x control).	.41	.23	2.32	.04	-.23	5.98	.003
c : 6 IVs - serial depth, propnl depth, goal depth, 3 interaction terms (goal x plan, prop x intern, serial x control).	.52	.44	6.06	.000	-.12	1.54	.2

Discussion

These regressions results provide a clear indication of the importance of bug type on debugging success. The results of the second regression, with the interactions included, are striking in that they suggest that the detection rate for different bugs may differ according to their depth in one particular structure of the program. If the psychological theorizing about the role that depth plays is correct then this implies that programmers (as a group, at least) must be capable of mentally representing these different structures.

The fact that bugs were more easily found when they were deeper in the structures contradicts the traditional interpretation of propositional analyses of comprehension. However, it is consistent with the idea that debugging occurs as part of an ongoing comprehension process, rather than on the basis of a constructed mental representation. Taken together these two results suggest that there is a critical role in debugging for a comprehension process which is capable of inspecting different structures within the code, either within individuals or, at least, between individuals.

Summary

In interpreting these results we should remember that the data were collected for a different purpose and that the task context for debugging was "desk checking", rather than programming in a real environment. However, the following important points can be made:-

1 - predicting detection rates is complex, but type of bug seems to be the main predictor (accounting for 36% of the variance).

2 - programmers seem to be able to target their attention towards different structures separately, leading to the different detection rates for different types of bugs.

Both of these points challenge the assumptions about debugging which were described earlier. First, debugging seems to be occurring as a part of comprehension; second, there seem to be different debugging (or comprehension) processes, focussing on different structures; and third, although bug type is a critical factor, the programmer's knowledge of the problem can only have an impact on some of the types of bug.

A Model of Debugging

What the above analyses show most clearly is the complexity of bug detection, but in particular they illustrate the fact that debugging success is a product of both comprehension processes and the actual meaning of the task, program and bug. In order to accommodate these factors we need a more flexible, strategic model of debugging behaviour. This will have three components.

1 - A flexible, incomplete comprehension process that can choose to represent different program structures according to task demands, tools available and skill level.
2 - A mental representation of a correct program for the problem as well as of the buggy program* .
3 - A process for detecting mismatches across these two representations of the problem and program.

Figure 3 illustrate the schematic view of the proposed model, in which comprehension clearly plays a central role.

Comprehension

The important characteristic of comprehension as presented here is an emphasis on both the program's behaviour and the code: ie both *what* it does and *how* it achieves it. In order to comprehend a program fully the programmer should have knowledge of both *what* and *how*, but in order to perform numerous programming tasks (including much maintenance and debugging) only knowledge of one of these is really necessary. A variety of factors will determine which structure the programmer focuses on. For example, the notation or the comprehension environment may provide better facilities for observing the behaviour of the program (*what*), than it does for teasing apart how the program actually works. BASIC is probably a good example of a language where programmers may focus on comprehending *what it does* information.

A complexity here arises from the recursivity of the programming process, in which one can treat parts of the program as programs themselves† . This leads to the characteristic that the mental representation can contain many different levels. How many levels will depend upon the size of the programming problem, the skill level of the programmer (novices may have difficulty comprehending even single statements), the familiarity of the problem and the type of mental representation being formed. In this way certain parts of the program can be represented by 'what' knowledge and other parts by 'how' knowledge.

---

* This is seen in our experiences of teaching programming to undergraduate psychologists, where the frequent stumbling block is not understanding the language and its syntax, but understanding the problem which is to be solved.

† This is true for any model of programming, but the implications are very different across the many models of programming.

In most programming environments it is *what* information which is most easily obtained, but the difference between 'what' and 'how' knowledge is affected by the language paradigm. For a declarative language (eg Prolog) the *how* information is predominantly procedural, whilst for BASIC, such procedural information would represent *what* knowledge. In this way one can see why tracing facilities are critical in a language like Prolog, since they provide the easiest access to *how* information about a Prolog program, whereas Pascal and BASIC programmers would seem to prefer breakpointing and variable tracing in order to acquire *how* information.

Debugging

Most programmers may feel satisfied with their comprehension when they understand either the *what* or the *how* information. However, when debugging the programmer is not completely free to choose which information to gather, since they need to be able to compare their comprehension model with their model of what or how the correct program should be.

We can assume that there can be as much complexity surrounding the comprehension of the problem as there is for the buggy program, but there is no reason (especially when debugging someone else's program) for the two comprehension models to be of the same form. However, when searching for a mismatch between problem and program (ie. debugging), it is essential for there to be some overlap between the models.

There are three important features of this theory of debugging:-

1 - the programmer must first reconcile their understanding of both program and problem.
2 - mismatch detection can take place in terms of either *what* or *how* information, dependent upon the nature of the programmer's knowledge of program and problem, and the support provided to them by the environment.
3 - mismatch correction can be performed in relation to both *what* and *how* information.

Thus, if a programmer comprehends a buggy program in terms of *what* it does, then debugging should be easiest if their understanding of the problem is in similar terms.

One task which is not well described within this framework is that of mismatch detecting - the crux of debugging behaviour. However, given this approach to debugging we are now in a much better position to conduct research into how differences between the problem and the program are detected.

Implications

The first important implication of this theory of debugging is that it characterises debugging as design, rather than as fault diagnosis. The model outlined in Figure 3 describes programming behaviour in general as well as it does debugging behaviour. The only difference is the quantity of information derived from the program code and the problem statement. One can, thus, view programming as a continuum from early design and coding (all information from problem statement) through debugging one's own program (equal information from program code and mental representation of problem) through to debugging someone else's code (equal information from program code and problem statement).

Although I have previously written about the importance of a distinction between coding and comprehension and, indeed between different types of comprehension, preferring to distance myself from the 'comprehension is the core of all programming' school of thought (eg. Gilmore and Smith, 1984), the view being described here does not contradict this. The danger of describing comprehension as the core programming process is that it is commonly as-

sumed to be a simple unitary process, rather than a complex strategic one. In the model presented here comprehension is still the central process, but it can take many different forms.

The task of understanding *how* a program works can be very demanding, especially for novice programmers (eg. due to working memory limitations), who find it easier to focus on *what* knowledge. This leads them to continually introduce new bugs into programs, since they are simply attempting to alter the behaviour of the program, rather than trying to correct the way in which the program works.

As well as providing this explanation of why novices introduce many new bugs to their programs, the model also provides a new perspective on the results of Davies (1990). He showed that BASIC programmers without education in program design skills did not use programming plans to aid their comprehension and debugging processes, whilst BASIC programmers who had been taught about program design performed like Pascal programmers, being guided by the plan structure of the program. Gilmore and Green (1988) had observed the behaviour of the first group and explained this in terms of notational differences (using the concept of *role-expressiveness*) which gave rise to different knowledge structures. Davies rejected this notational difference, claiming that design skills were the cause of the Pascal - BASIC differences observed by Gilmore and Green. But, within the new framework for debugging presented here, it is possible to see how both the notation and the existence of design skills could give rise to strategic differences amongst programmers, leading some groups to focus on *how* information, comparable to programming plans.

The importance of congruence between the programmer's models of the program and the problem may also explain the results of Petre and Winder (1988) who found that, despite frequent arguments that different languages are suited to different kinds of problems, skilled programmers used the same algorithm for the same problem across a wide range of programming languages. This finding does not refute the claim that different languages are suited to different problems, particularly since only half the experts used the same language first for all the presented problems, but it does indicate that the cognitive cost of re-constructing one's understanding of the problem is expensive relative to the cost of translating it into a different language.

From the system designer's perspective an important implication in these results and this model of debugging is that there is a clear, critical role for the programming environment in debugging performance, something which is remarkably lacking from earlier models. Environments which provide 'good' sequential trace information for procedural languages may be reinforcing a *what-it-does* comprehension strategy, when *how-it-works* knowledge might be more important to the programmer whose understanding of the problem is of this form. Paradoxically tools which support the programmer in finding an alternative perspective on the code may be viewed as irrelevant by a programmer who does not realise that an alternative exists and whose model of the problem is congruent with their preferred comprehension strategy.

Summary

From the review of existing models of debugging and from the regression analyses on the detection rates of various errors, I have developed a model of debugging which places it amidst normal programming activities (ie. as a part of design), rather than separating it out as a different activity (ie. fault diagnosis). One of the model's main features is the important role given to the mental representation of the problem to be programmed, as well as the mental representation of the program itself. The act of debugging is one of reducing the difference between the actual code and that required, through a comparison of these two mental representations.

References

Curtis, W (1988) Five paradigms in the psychology of programming. In *M Helander (Ed) Handbook of Human Computer Interaction*. Amsterdam: North-Holland Press.

Davies, S (1990). The nature and development of programming plans. *International Journal of Man-Machine Studies, 26.*

Gilmore, D J & Smith, H T (1984). An investigation of the utility of flowcharts during computer program debugging. *International Journal of Man-Machine Studies, 20*, 357 - 372.

Gilmore, D J and Green, T R G (1984). Comprehension and recall of miniature programming languages. *International Journal of Man-Machine Studies.*

Gilmore, D J and Green, T R G (1988). Programming plans and programming expertise. *Quarterly Journal of Experimental Psychology, 40A*, 423-442.

Gugerty, L & Olson, G (1986). Comprehension differences in debugging by skilled and novice programmers. K E Soloway and S Iyengar (Eds). *Empirical Studies of Programmers.* Norwood, New Jersey: Ablex Publishing.

Jeffries, R, Turner, A, Polson, P and Atwood, M (1981). The processes involved in designing software. In *J R Anderson (Ed) Cognitive Skills and their Acquisition.* Hillsdale, NJ: Lawrence Erlbaum Assoc.

Johnson, W L, Soloway, E, Cutler, B and Draper, S (1983). *Bug Catalogue: I.* Research Report No. 286, Computer Science Dept, Yale University, Newhaven, Connecticut.

Katz, I & Anderson J R (1988). Debugging: An analysis of bug location strategies. *Human-Computer Interaction,* 3, 351-400.

Kessler, C M & Anderson, J R (1986). A model of novice debugging in LISP. In E Soloway and S Iyengar (Eds). *Empirical Studies of Programmers*. Norwood, New Jersey: Ablex Publishing.

Petre, M & Winder, R (1988). Issues governing the suitability of programming languages for programming tasks. In D Jones and R Winder (Eds). *People and Computers IV.* Cambridge: Cambridge University Press.

Vessey, I (1985). Expertise in debugging computer programs: A process analysis. *International Journal of Man-Machine Studies, 23*, 459-494.

Vessey, I (1989). Toward a theory of computer program bugs: an empirical test. *International Journal of Man Machine studies, 30*, 23-46.

Waddington, R & Henry R (1990). Expert programmers re-establish intentions when debugging another programmer's program. In D Diaper, D Gilmore, G Cockton & B Shackel (Eds). *Proceedings of Interact '90.* Amsterdam: Elsevier North Holland.

Wertz, H (1982). Stereotyped program debugging: An aid for novice programmers. *International Journal of Man-Machine Studies, 16*, 379 - 392.

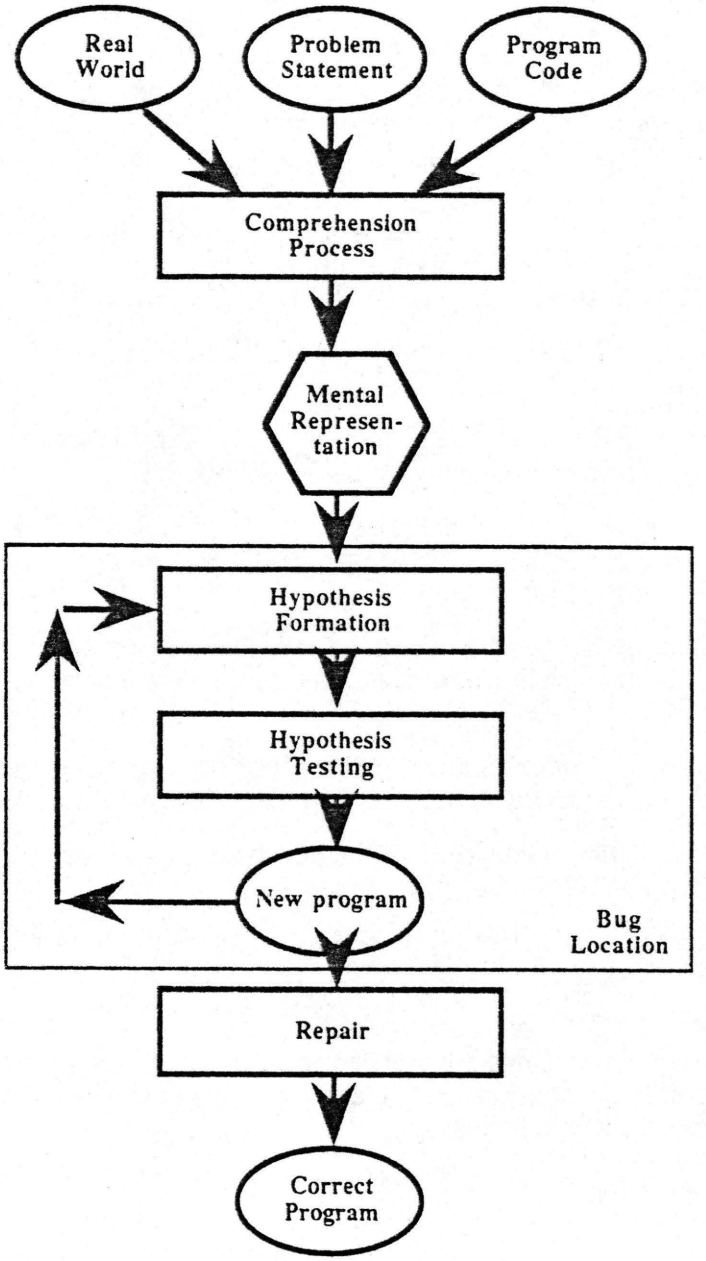


Fig. 1: A schematic view of many models of debugging.

Key

Cognitive processes

Real-World Entities

Cognitive States

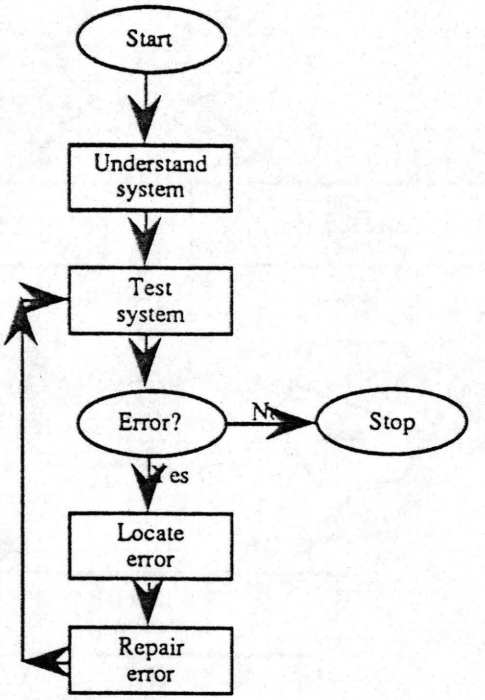


Figure 2: A model of debugging (from Katz and Anderson, 1988, p. 353).

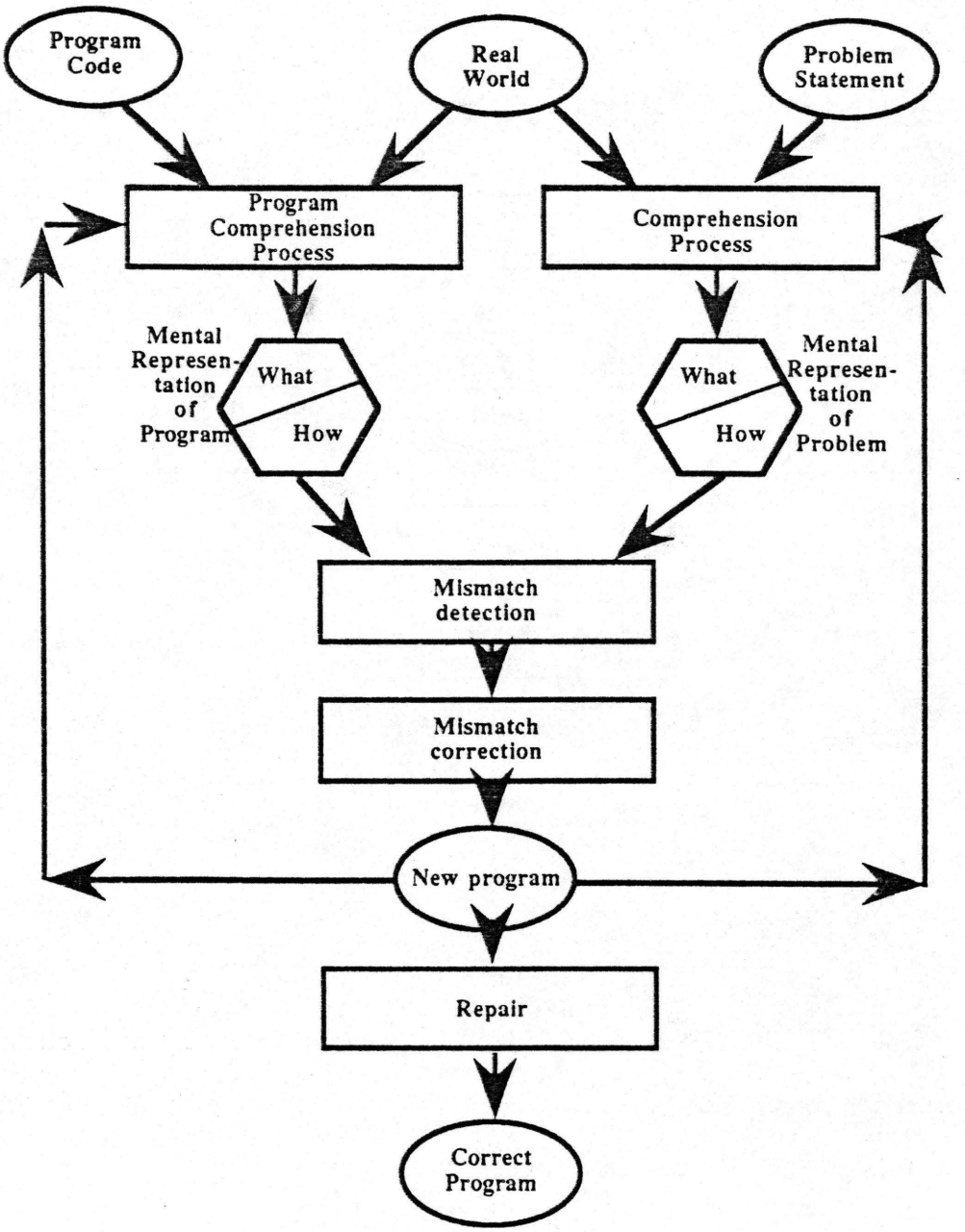


Fig. 3: A schematic view of debugging.

Key

Cognitive processes

Real-World Entities

Cognitive States

Identifying and meeting designers' requirements for task analysis tools.

Hilary Johnson & Peter Johnson

Department of Computer Science
Queen Mary and Westfield College
University of London.
Mile End Road
London
E1 4NS
hilaryj@uk.ac.qmw.cs
071-975-5246
pete@uk.ac.qmw.cs
071-975-5224
Fax081-980-6533

Abstract

The authors have developed an approach to task analysis (TA) known as Knowledge Analysis of Tasks (KAT). KAT comprises a theoretical framework and supporting methodology for carrying out task analyses. The ultimate objective in producing KAT is to integrate TA and task models into system design by providing information about users and tasks at appropriate stages in design. To facilitate the integration of TA into system design, a survey of designers was conducted to identify where, when and how task data can be represented to aid integration into system design. We were also keen to know what designers needed in the way of tools to support TA and its integration into system design. In this paper we discuss the major tenets of the theory and then summarise the results from the survey. Designers' requirements for TA tools are then outlined, followed by a discussion as to how these requirements might be met. Finally, brief details of design specifications for the proposed KAT tools are presented.

Introduction.

The study of human computer interaction HCI has an overriding goal, that is to assist software designers to construct computer systems which people find useful and usable. One way to approach this goal is to assume that knowing something about how users approach and carry out tasks will aid software designers when making design decisions which will ultimately affect computer system usefulness and usability. Rosson et al (1988) have argued that designers consider obtaining information about users and tasks as a major contributor to the generation of design ideas. They distinguish between design idea generation and design development and suggest that design idea generation is facilitated by task/user analysis tools and techniques, while design development is facilitated through the use of specification and

prototyping tools. By carrying out task analyses, users and tasks will be taken into account in the formation of the design idea. If this is the case then interface and system design will take place from an informed position on the part of the designer, resulting in the need for less iterations.

Approaches to task analysis (TA) typically investigate what people do when they carry out one or more tasks. Information about how users plan and carry out goal-directed behaviour is one aspect of users and tasks that can be identified and analysed. Probably the most often cited approach to TA in HCI is the GOMS approach of Card et al (1983). The underlying philosophy of GOMS is that tasks can be thought of in terms of goal structures, operations, methods and selection rules. Kieras and Polson (1985) extend the GOMS approach by arguing that production rules can be used to model goals, operations, methods and selection rules, and furthermore that these production rules bear a close relationship to the way in which people structure their knowledge of the task. However, neither Card et al nor Kieras and Polson explicitly claim that people actually possess task knowledge structures.

Recently, the authors and colleagues (Johnson et al 1988a; Johnson et al 1988b; Johnson and Johnson, in press) have outlined an approach to task analysis (TA) which comprises techniques for gathering and analysing information about users and their tasks, and then subsequently building task models in the form of "task knowledge structures", (TKS). Frame-based representations have been used to represent appropriate aspects of system design in terms of those task knowledge structures. In developing task knowledge structures we have taken account of a number of theoretical assumptions which we have made on the basis of existing theoretical and empirical work in cognitive psychology.

Theoretical Basis for TKS.

i) *Task knowledge as conceptual knowledge.* Task knowledge is represented in conceptual or general knowledge structures in long term memory. This notion corresponds to the theoretical position taken by Schank (1982) in assuming that knowledge of frequently occurring events is structured into meaningful units in memory.

Empirical support for our assumption that people possess something akin to TKSs can be found in the work of Galambos (1986). Galambos conducted a series of experiments which show that people recognise and use structures of events, such as the order, the sequence and the importance of activities within the event sequence to understand, explain and make predictions about those events. Further support for our view that task knowledge is represented in long term memory comes from the work on text and story comprehension

by Graesser and Clark (1985) in which general knowledge structures relating goals to causal and enabling states, plans for achieving goals, intermediate states and alternative solutions or paths are all represented in a conceptual knowledge structure for the events of the story.

We assume that all the knowledge a person possesses about a task is contained within the task knowledge structure and that the TKS is activated in association with task performance.

ii) *Structure in tasks.* Task knowledge and therefore TKSs would be "unstructured" if the knowledge people possessed about the task world (or domain) was such that all possible elements of the knowledge used in those tasks could be associated, and was associated with equal probability to other knowledge elements. This structure is not just an imposed structure but is a reflection of the structure found in tasks in the real world. All possible elements and attributes of a task do not co-occur with equal probability. Our view here is similar to that of Garner (1974) in describing how people represent physical objects based on the structure of those objects in the real world.

We argue that task behaviours do not occur independently of one another. Some pairs and n-tuples of task behaviours are quite probable while others which might be logically or physically possible, never occur together in reality. Within tasks, some behaviours are carried out together, precede or follow other behaviours, and in some cases prime, or are primed by other behaviours. One form of this structuring is in a PLAN which provides a feasible and acceptable structuring on the task behaviours. Empirical evidence for the representation of PLANS in long term memory can be found in the work on programming tasks (Green et al, 1987). Further evidence for structuring in tasks comes from the work of Byrne (1977) on tasks in domains such as cookery and meal planning.

iii) *Representativeness of task elements.* A TKS is composed of a number of task elements which differ in representativeness. This is similar to the notion of representativeness used by Rosch (1985) and colleagues (Rosch et al 1976) to describe the relations between objects and their categorical representation in memory.

We assume that a TKS includes knowledge about objects and their associated actions and that these task elements differ in how representative they are of the TKS of which they are a part. One implication of this is that the procedures containing these representative objects and their associated actions are more central to the TKS than other procedures. Empirical

evidence for the notions of procedural centrality and action/object representativeness in task behaviour can be found in Leddo and Abelson (1986) who found that for tasks such as borrowing a book from a library there were particular segments which were more central to the particular instantiation of going to the library and more representative of the class of similar tasks. Frequency of occurrence alone was not a reliable indicator of object and associated action representativeness nor of procedural centrality.

A theory of TKS.

A TKS is a summary representation of the different types of knowledge that are recruited and used in task behaviour. A TKS is related to other TKSs by a number of possible relations.

One form of relation between TKSs (within role relations) is in terms of their association with a given role. A role is assumed to be defined by the particular set of tasks that an individual is responsible for performing as part of their duty in a particular social context (a "job" is one form of social context). A person may take on a number of roles, for example "author", "referee", "teacher" etc. There are tasks associated with each of these roles. There will be a TKS for each task that a role may be required to perform. Those tasks that are related because they are performed by the same role will have the role relation property associating their respective TKSs.

A second form of relation between TKSs (between role relations) is in terms of the similarity between tasks across roles. This occurs when a person is performing a similar task under different roles. For example, a person may carry out the task of "arranging meeting(s)" while assuming the role of "chairman", "client", "husband". Each task may be performed differently in one or other respect. However, a single person assuming all these roles would have a knowledge structure for each task and also knowledge (not necessarily explicit) of the relations between these tasks. Thus the TKSs for each of the "meeting arranging" tasks would have relational links to other "meeting arranging" tasks performed by that person when assuming a different role.

Within each TKS there are other knowledge representations. The goal structure identifies the goals and subgoals contained within the TKS. A goal is assumed to be the state of affairs that a particular task can produce and forms part of a person's conceptualisation of their task world while carrying out a given role. A subgoal is a lower level goal. The goal structure also includes the enabling and conditional states that must prevail if a goal or subgoal is to be achieved. In this way the goal structure represents the plan for carrying out the complete task. A plan is the particular formulation and possible ordering of subtasks that are undertaken to achieve a particular goal. The plan defined in the goal structure is carried out through a procedural structure

which contains alternative procedures for achieving a particular subgoal. A procedure is a particular element of behaviour that forms part of a subtask. As there may be alternative sets of procedures for achieving a particular subgoal there are also conditional and contextual groupings of these procedures. In this way strategies are represented. Strategies are particular sequences of procedures. The procedures rely on other knowledge of the objects and actions which when combined constitute a given procedure statement. Actions and objects are the lowest level of elements of tasks.

Action and object knowledge is represented in a taxonomic substructure. Within the taxonomic substructure information about the properties of the action(s) and object(s) are represented. This includes, the representativeness of the object, the class membership and other attributes such as the procedures in which it is commonally used, its relation to other objects and actions, its features (such as what the object can do or possess) and its typicality rating. These are described in more detail in Johnson et al (1988a) along with a number of techniques for identifying and modelling TKS components. Johnson et al (1988b) provide a method for applying these task models to design using examples from real tasks that have been analysed and real designs in complex domains that have been proposed.

Task analysis and system design.

In order to facilitate the introduction and use of task analysis in system design we need to know what software design practice entails, and also how to provide tools to support designers in carrying out task analyses. A current view of design processes and practices in system design is provided by Johnson and Johnson (1989). However, we also need a view of what designers consider to be the potential contribution of task analysis to various stages of system design. Additionally, if we are to construct TA tools then we need to identify designers tool requirements. A survey was undertaken to identify this information. Below we summarise some of the survey findings. (Full details of the survey can be obtained from Johnson and Johnson, 1989).

The designers who participated in the survey thought that the contibution of TA might be:

At the feasibility/initial planning stage:

*Identifying and documenting any new functions/new tasks the computer may support.
*Identify potential functionality of the system from user perspective.
*Identify user population and characteristics of that population.
*Identify characteristics of interface to be developed.
*Allocation of function between user and system.

*Assess scope/degree of larger-scale TA to be undertaken later in development lifecycle.

At the requirement/analysis stage:
*Identify and document user/UI requirements comprising details about;
*hierarchical structure of tasks (goals and subgoals)
*how users achieve goals and subgoals
*listing and ordering of undertaking task procedures
*frequency with which particular procedures were carried out by users
*reasons why and circumstances under which one procedure was used in preference to another
*inputs and outputs from each procedure
*events, data used, actions, objects
*standard set of properties relating to objects and actions, e.g. frequency, time taken, etc
*expectations the user entertains about the system after user has carried out an operation
*division between user and system.

At the Design stage/ User interface Development/ Dialogue design:
*Provide initial input to guide dialogue and screen design, comprising;
*details of what users expect to have available to them at any one time;
*the structure and sequence of their usage of system facilities;
*the names and form of representation to be given to screen-presented objects and events;
*information that should be available in given contexts,(i.e. design of screens);
*structure between contexts, (i.e. mapping between screens)
*how much to put on the screen at once with reference to number of commands
* what information to go on screens and the grouping of that information
*what commands are needed to support user operations and what those commands will be
*User testing.

At the prototyping stage:
*Guide initial format and presentation of prototype by indicating what the screens should look like
*Identify data that has to be displayed.
*Identify operations and sequencing of procedures
*Ensure dialogue specification is represented in a format that can be understood and verified with end users and to carry this out.

At the validation stage :
*User testing

At the update and maintenance stage:
*Identifying, documenting and cataloguing user problems.

Designers' tool requirements.
The designers were also asked about the desired characteristics and criteria they thought that the proposed TA tools must satisfy; the results are summarised below:

*easy to use
*on-line
*can be used on a small-scale at first then used more extensively later
*must have a facility for registering and defining objects and actions
*facility to list characteristics and attributes of objects
*facility to help in constructing procedures and task hierarchy
*facility to show task hierarchy and procedures graphically - and allow procedures to be expanded into detailed operations by some diagrammatic convention
*assist in process of transforming task model into representations used by designers.

Meeting designers' TA tool requirements.
The first two requirements that the tools should be easy to use and also on-line were more general requirements than the more specific facilities that the designers would like to be made available to them.

We have developed a set of design specifications for tools to support designers in using KAT and integrating the resulting task models/data into system design. (A brief outline of the tool design specifications is given in the next section; for a fuller account, see Johnson and Johnson, 1990). These specifications are for on-line tools thus satisfying one of the designers' more general requirements. The other more general requirement was that the tools must be easy for designers to use. The specifications for the tools have been put together from an informed and principled position, i.e. by undertaking surveys into system designers tool requirements and then keeping in mind the results of these surveys when designing the tools. However, the resulting tools built from the specifications would have to be tested extensively on the designers in order to assess how easy they were to use. A further series of tests would then be necessary to establish whether the resulting systems, built after the designers had successfully used the tools in those designs, were easier for the end-user to interact with. We have not as

yet implemented the TA tools. The TA tool specifications incorporate elaborate help facilities and introductions to task analysis to aid the system designers. The implemented tools will support the construction of a task model but this can be undertaken to a lesser or greater extent thereby making the tool appropriate for small scale use at first and more extensive use later.

An object/action database or dictionary will allow TA tool users to register, define and characterize attributes of objects and actions and their interrelationships to each other and amongst themselves. A goal-oriented structure to be completed by designers should, with the addition of appropriate help facilities, aid in the construction and graphical representation of task hierarchies and procedures.

A non-uniform view of design practices and minimal end-user involvement (see Johnson and Johnson 1989), leads us to conclude that the proposed tools must be versatile and flexible, capable of integration with different approaches to, and stages in design, should not impede design stages and should be capable of integration into both structured design methods and other design lifecycles.

The content and function of the task analysis tools should provide end-user/task information and users' interface requirements, relate identified requirements back to end-users in a non-technical fashion, and lessen the need for designers to have to rely only on their intuition. The TA tools should also provide the basis for getting user feedback, decrease the number of end-user problems associated with using the system in undertaking tasks and provide a basis for including end-users in quality assurance procedures.

In addition to the proposed TA tools being general purpose we decided that an independent subset of these tools should be specifically tailored to one of several potential structured design methods. SSADM has been chosen, for a number of reasons, largely beyond our control, as the structured design method for our tool subset.

Design specifications for the KAT toolset.

In this section we outline some of the features which characterize the proposed tools. The tool specifications have been developed taking into account the designers' requirements discussed in the previous section. In the full report (Johnson and Johnson 1990) the authors indicate how the tool screens might look, (i.e. the user interface to the TA tool), the menu labels and content, and also describe the functions that will be available to the users, and/or will be carried out automatically by the implemented TA tools. Here we give brief details of the screen contents and functions.

The format of the TA tool screens will be as follows: First, screen titles will identify and refer to particular screens. A screen title will appear at the top of the screen. Below the screen title, there will be shaded menu headers which will be visible at all times when the screen has been selected. Finally, under each of the menu headers are pop-up menus with menu items which are activated when the menu header is selected. Therefore all the menu items are available to the user at any time that the user chooses, on selection of the appropriate menu header. Some of the menus also have within them cascading menus. On the user selecting a menu item the pop-up menu disappears and the appropriate text will appear on the screen. Text on the screen will be presented as on-line scrollable text. Below we give an indication of the contents of each of the screens.

Screen 1 Knowledge Analysis of Tasks (KAT) Screen. Screen 1, will be always present when the application is started. It will be mainly an information screen giving general details about KAT and the KAT tool. This screen will provide an introduction to KAT activities. Screen 1 will contain:-
i) an introduction to KAT
ii) a help facility
iii) introduction to task analysis,
iv) general guidelines for conducting task analyses
v) a glossary of TA terms
vi) details about the scope of TA,
vii) an outline of the types of knowledge components that comprise KAT
viii) introductions to "collecting data", "analysing data" and "modelling tasks".
When the user wants to actually undertake KAT activities, other screens will be activated automatically as a result of the user selecting either "Collecting data", "Analysing data" or "Modelling tasks" menu items. In addition Screen 1 will provide the user with a short-cut to the TKS screen (described later).

Data collection and analysis screen. This screenwill contain:-
i) a help facility
ii) introduction to data collection
iii) introduction to knowledge components
iv) identifying and registering actions and objects in the action and object databases.
v) identifying procedures and plans
vi) identifying goals and subgoals
vii) knowledge sources
viii) data collection techniques with an introduction to each technique, their advantages, disadvantages, appropriateness, and a proforma questionnaire.
ix) introduction to data analysis
x) Generating generic (common) task elements

xi) generic actions and objects
xii) generic plans, subgoals and procedures

In addition to the information above, there will be windows associated with the data collection and analysis screen, some of which will have computer supported functions.
*Computer supported generification procedures will be supported by a generification window which when activated will identify generic actions and objects from those entered by the user, dependent on a user-defined threshold. As a result the window will display a comprehensive list of generic objects and actions, listed alphabetically and with the corresponding frequency also quoted. This list can then be transferred to the taxonomic substructure.
*The object database window and the action database window will contain scrollable lists of actions and objects.
*The goal structure window will support the input of plans, procedures, goals and subgoals, which it will present to the user in both graphical and textual (including rule-based) form.

Task Knowledge Structures (TKS) screen. This screen will display the empty slots of the TKS which have to be completed by the designer. Extensive help facilities will be available at this screen to indicate how the slots can be filled and referring to each of the other screens and windows for techniques and methods by which the TKS might be completed.

System integration screen. Once the designer is satisfied with the partially complete or fully completed TKS, the task model can then be transformed into one of the various representational formats which will make up the System integration screen. This screen will comprise the following features:-
i) Introduction to system integration
ii) Potential integration of TA into system design
iii) KAT's contribution to design at the presentation, dialogue/interaction and functional levels.
iv) Task modelling representational formats:-
Introduction,
English,
Diagrammatic/graphical representation
Production Rules
SSADM:-
Introduction
Feasibility
Systems analysis
System design

This completes the details of the proposed KAT screens and windows. As remarked earlier each of the features requested by the designers in our survey data have been taken into account in the design specifications of the KAT task analysis tools. In the next stage of the research the KAT tool designs will be implemented.

Summary.
At the beginning of this paper we argued that one way of assisting designers to build usable and useful computer systems was to provide them with information about how users plan and carry out goal directed behaviour. We then discussed the uses of task analysis in system design, followed by a description of the theoretical background to TKS. The results of a survey concerned with identifying what designers wanted, needed and expected in the way of tools to support task analyses and their integration into system design, were presented. The designers' requirements are discussed along with ideas about how to meet those requirements. Finally, we provided a very brief description of a set of design specifications for proposed KAT tools to support designers in carrying out task analyses and their integration into system design.

Acknowledgements.
We are grateful to ICL URC for funding the project "The development of task analysis as a design tool".

References.

Byrne, R., 1977 Planning meals: Problem solving on a real data-base. Cognition 5, 287-332.

Card, S.K., Moran,T.P. and A. Newell, 1983 The psychology of human-computer interaction. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Galambos, J.A., 1986 Knowledge structures for common activities. In J.A. Galambos, R.P. Abelson, and J.B. Black (eds) Knowledge structures. Hillsdale, New Jersey: LEA.

Garner, W.R., 1974 The processing of information and structure. New York:Wiley.

Graesser, A.C. and L.F. Clark, 1985 Structures and procedures of implicit knowledge. Norwood, New Jersey. Ablex Publishing Corpn.

Green, T.R.G. Bellamy, R.K.E. and J.M. Parker, 1987 Parsing and gnisrap: a model of device use. In H.J. Bullinger, and B. Shackel, (eds) INTERACT 87

Proceedings of the Second IFIP Conference on Human-Computer Interaction, 1-4 September. Elsevier:North Holland.

Johnson, P. Johnson, H and F. Russell, 1988 Collecting and generalising knowledge descriptions from task analysis data. ICL Technical Journal, 6, 137-155.

Johnson, P., Johnson H., Waddington, R. and A. Shouls, 1988 Task related knowledge structures: Analysis, modelling and application. In D.M. Jones and R.Winder (eds), People and Computers: From research to implementation. Cambridge: Cambridge University Press, 35-62.

Johnson, H and P. Johnson, 1989 Integrating task analysis into system design: Surveying designers' needs. Ergonomics special issue.

Johnson, H and P. Johnson, 1990 The development of task analysis as a design tool. Design specifications for tools to support task analyses. Report to ICL, January.

Johnson, P and H. Johnson, in press Knowledge Analysis of Tasks: Task analysis and specification for human-computer systems. In A. Downton (ed) Engineering the human-computer interface. London: McGraw Hill.

Kieras, D and P.G. Polson, 1985 An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22, 365-394.

Leddo J. and R.P. Abelson, 1986 The Nature of explanations. In J.A. Galambos, R.P. Abelson, and J.B. Black (eds) Knowledge structures. Hillsdale, New Jersey: LEA.

Rosch, E. 1985 Prototype classification and logical classification: The two systems. In E.K. Scholnick (ed) New trends in conceptual representation: Challenges to Piaget's theory? New Jersey: Lawrence Erlbaum.

Rosch, E., Mervis, C., Gray, W., Johnson, D., and P. Boyes-Braem, 1976 Basic objects in natural categories. Cognitive Psychology, 8, 382-439.

Rosson, M.B., Maass, S., and W.A. Kellogg, 1988 The designer as user: Building requirements for design tools from design practice. Communications of the ACM, 31, 1288-1298.

Schank, R.C. 1982 Dynamic Memory: A theory of reminding and learning in computers and people. New York: Cambridge University press.

TASK ANALYSIS APPLIED TO CONTROL SYSTEM DESIGN

Jari Kallela
Technical Research Centre of Finland
Laboratory of Electrical and Automation Engineering
Otakaari 7 B
02150 Espoo, Finland
fax: +358 0 4356 7014
tel: +358 0 456 6457

ABSTRACT

Currently the process industry has only few practical methods for designing the human subsystem; i.e. the control room and operator's work. On the other hand, many problems associated with complex systems have been attributed to the human operators. Improving man-machine system design will contribute to decreasing these problems.

In this paper, task analysis is used to aid the design of the control system. This approach enables designers to consider users prior to detailed design. The purpose is to import new features to design practice in order to make the production more efficient and tasks more suitable for the operators. The ideas are applied to the control system design of a paper machine.

INTRODUCTION

Industrial processes are becoming more complex and integrated. Flexibility, efficiency and availability are essential for plant performance. In addition, authorities urge safety and low environmental load. These demands can be handled at least to some extend by increasing the level of automation. This has had a great impact on operator's job. The operator has been removed from the control loop and his work consists mainly of supervision and emergency management.

Although human operator and his work is crucial for successful production practically no methods are used for work design. The control system in particular modifies operator's work. In this paper, some new methods for the design and the analysis of control tasks are presented.

The main topics of this paper are illustrated in Fig 1. First, the process model which describes the control tasks and task structure is build for a paper machine. The automation decisions which are made in control strategy meetings can benefit from this task structure. It is also used for constructing the CRT displays.

The engineering view of the design process normally considers the design as a flow of consecutive phase and it relies on top-down decomposition of the design problem. This

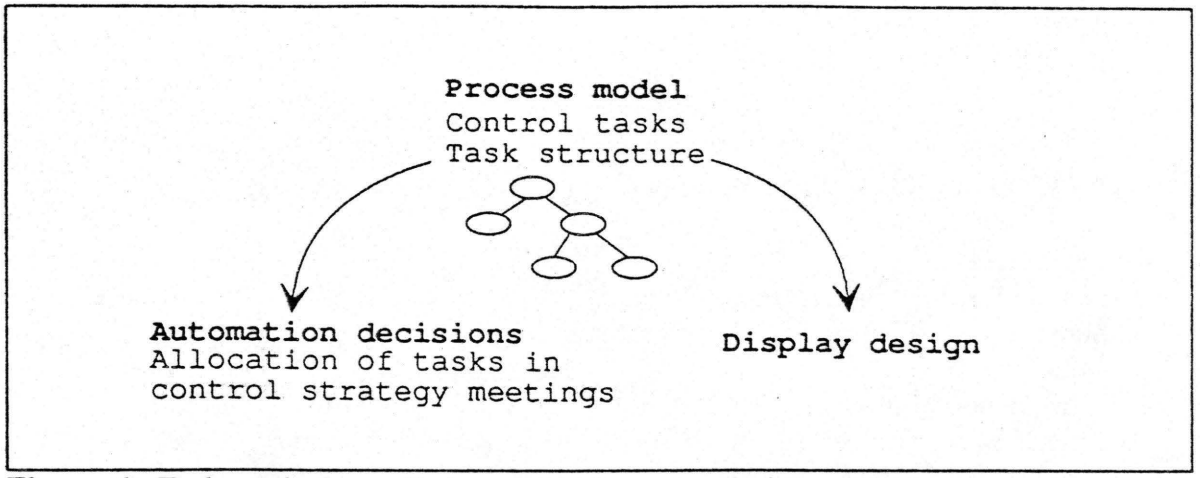


Figure 1. Task analysis applied to control system design.

is a useful approach when project management is concerned. In the current design practices the life cycle of a process automation system is divided into phases like preliminary design, basic design, detailed design, start-up and operation (Ranta 1985).

The design can be also seen as an iterative process where the design is followed by describing and analyzing the results. The actual design is the creative part where solutions are found to the problem. Documenting and describing the solutions properly enables the designer to communicate and get comments about the solution. The analysis is then used to evaluate and improve the solution and the analysis gives initiative to the next design-description-analysis loop. These phases can not always be separated; the design may look like drawing descriptions while the constraints are considered at the same time.

PROCESS MODEL

The functional model of the process consists of two types of functions: process and control. Process function transfers the raw materials and energy into products and waste. Physical and chemical reactions characterize the process functions and mathematics is used for modelling. Control functions, in turn, are based on information processing, input and output data and they can be modelled in terms derived from software engineering.

Naturally, the process function is controlled by the control function. Goals are important when proper control is determined. Control functions receive their input information from the process itself, from raw materials and products (Fiq 2).

Paper machines are large and complex systems and in Finland, they are highly automated. In a Foudranier-type paper machine the paper stock is distributed onto a moving wire which forms the fibers into a sheet. The additional water is removed by presses and heated cylinders. In the end, the paper is wound into a reel (Smook 1986). Paper making is a continuous process and typically four men are working on a machine in three shifts.

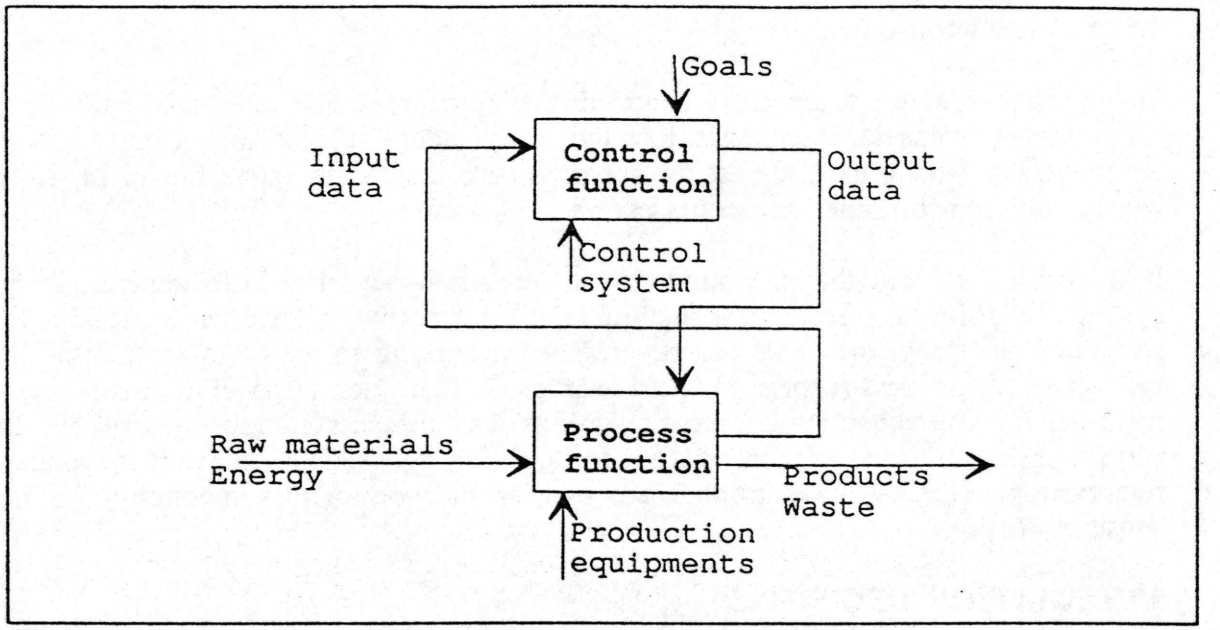


Figure 2. The connections between process and control function.

The control functions demanded by the paper machine can be organized in a hierarchical way. These "generic" control tasks are closely related to the paper making process, the subprocesses, the production phases, the states of the machine or the state transitions. The hierarchy is illustrated in Fig 3.

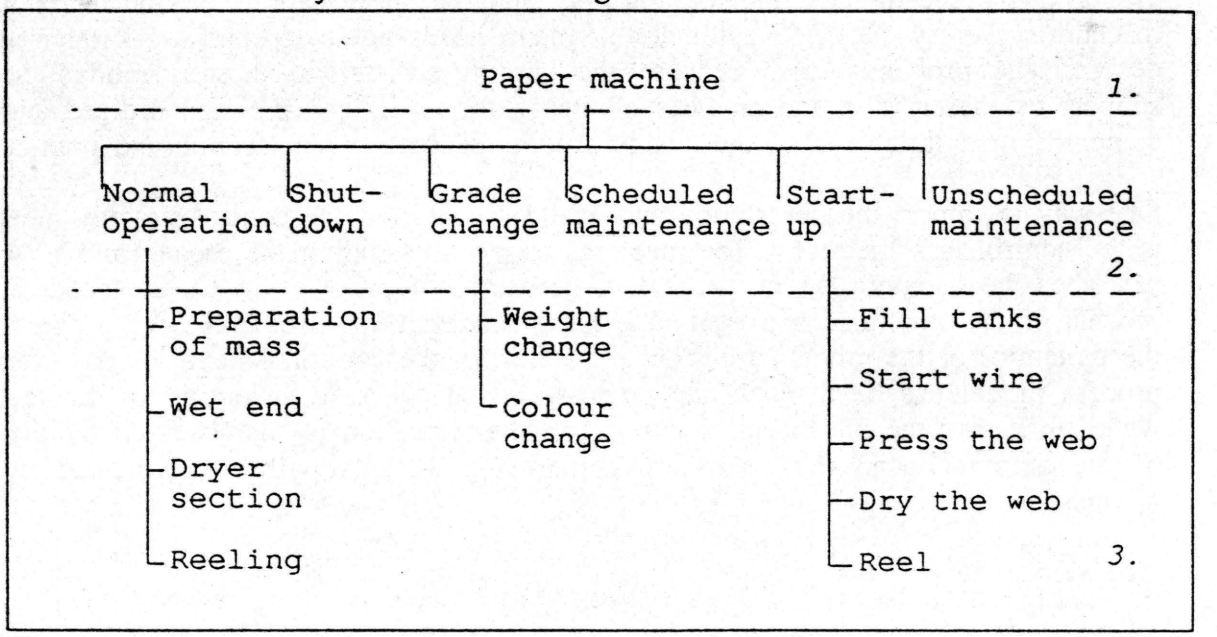


Figure 3. Hierarchical structure of control tasks in a paper machine.

The top level of the hierarchy consists of tasks concerning production planning, R&D and maintenance scheduling. The second level is made of the different states in which the paper machine can be. The operation is regarded as normal when the paper machine is producing acceptable paper. Typically, the normal operation covers over 80% of

the total operation time.

In the third level there are tasks controlling subprocesses like the tasks under "normal operation" or steps in a sequence like the tasks under "start-up". It seems logical that all the tasks which have the same superior task belong to the same class, in other words they control either subprocesses, phases or states.

It is not self-evident that this kind of hierarchy can be formed. In general, there are two grounds for a hierarchy: structural and constructive (Harré et al 1985). In the structural hierarchy, the goals can be arranged according to the tasks so that the higher task gives goals or subgoals to the lower ones, like "normal operation-subprocesses" relationship. The constructive hierarchy means that the parts compose the whole. In the control tasks this kind of constructive features can be seen in "start-up -> sequences" relationship. Both of these grounds are used in the control task hierarchy which may cause confusions.

AUTOMATION DECISIONS

Throughout the design of human/machine systems, design decisions are made, explicitly or implicitly, that define the level of automation and also the role of the human. However, the functions of the control system are specified mostly in control strategy meetings at the end of basic design. Users, process designers and automation designers take part to these meetings. Currently, there are many problems in these meetings: the discussions proceed in an inconsistent way and the specifications produced are partly inaccurate. Poorly specified automation system needs a lot of extra work later in the design. The problems arise because the current practice in these meetings do not support the description and analysis of the solutions. If they are standardized and documented in a suitable way, the design process become more accurate and predictable.

One way to import the description and analysis phases to the control strategy meetings is the control task hierarchy. The meetings start with building the process model which describes the behavior of the process. Other kinds of process models are needed, too: models of the structures and equipments and models how to evaluate the process. But the dynamics of the process are most important for the control system design. After the process model has been build, the process can be analyzed according to the features which influence the allocation of the tasks. The allocation is characterized by the role of the automation in the control task, i.e. what actions will be performed by the automation.

Analyzing the process

Now the analysis concentrates on the process. The analysis reveals factors which are difficult or expensive to automate and which demand special attention. The features of the process which influence the automation are for example:

- Type of the process. Is it a continuous/batch process?
- Extent. How large is the process?
- Dynamics. How often are the control tasks repeated, how long do they last?
- Uncertainty. Can all important factors be observed, are measurements reliable?

How dangerous it is?
- Connections. How are the important factors connected to each other and to the environment?
- Disturbances. What are the typical problems in this kind of process?

After the analysis, the control tasks are divided between operator and automation. The problem is that a control task can be automated in many ways and only few rules can be applied (Frey, Rouse 1990). The human operator and his abilities are limitations from the designer's point of view and it is impossible to find one "optimal" allocation suitable for all operators. Therefore, the changes and improvements should be possible later, even after the design phase.

Describing the allocation

During the allocation, the role of the automation is decided and described. This description is needed later when the automation system will be configurated. Sometimes the role of the operator has to be documented, too.

The control task gives a natural basis for this description. Different subtasks can be found in a control task. Components of a sequential completeness of a task structure are preparation, execution and checking and organization of the task (Hacker, Iwanowa, Richter 1983). Another approach emphasizes the work of the process operator: control, special procedures and drills, routine maintenance, recording and reporting (Crossman 1974). Concerning the level of automation, a convenient set of subtasks in a control task is:

1. Preparation. Planning, predicting future developments of the process, optimizing the process.
2. Input information. Measurements.
3. Information processing. Phases of decision making: diagnose, goal setting, decision and action planning (Goodstein, Andersen, Olsen 1988)
4. Execution. Continuous and sequential control.
5. Checking. Monitoring.
6. Communication. Reports.
7. Disturbance management.

Each one of these subtasks can be, at least in principle, automated to a different degree, so the decision have to be made for every subtask. The level of automation includes the role of the automation in all of these subtasks.

Explicit example: Grade change

Paper machine produces typically about 20 different paper grades which differ from each other by the basis weight, color, thickness, smoothness or other properties. The goal is to change the grade fast (not to lose production time) and to make unacceptable paper as little as possible.

The grade change can be analyzed according to the features of the process:

Type

The grade change is batch-type, because it has a clear start and end.

Extent and dynamics

Grades are changed on an average 2-3 times a day but variations are remarkable. It takes 5-10 minutes if only the basis weight is changed so the change of basis weight is reasonably fast and simple. When color is changed, it takes considerably longer because there are so much water circulating in the machine.

Uncertainty

On-line basis weight measurement is good enough but on-line color measurement is no reliable. Laboratory results can be trusted but they are slow.

Connections

The productivity of the paper machine depends on the successful grade changes. The amount of broke produced in the grade changes can make the normal operation difficult.

Disturbances

Breaks are common during the grade change

Fig 4. illustrates how the subtasks of the grade change are normally automated.

	Automation	Operator
Planning		selects orders
Input	measures parameters	
Information processing		gives setpoints and color dosing
Execution	automatic control loops	
Checking		monitors
Communication	transmits lab results	gets orders, reports
Disturbance management	alarms when limits are exceeded	

Figure 4. Level of automation in grade change.

The present situation can be improved if on-line color measurement becomes trustworthy. Then the execution of the grade change can be thoroughly automated. The operator decides when and what grades will be changed.

DISPLAY DESIGN

The field of CRT based display design, both in the area of applicable guidelines and research dedicated to the effectiveness of information design with particular emphasis

on industrial process control, is a relatively new area (Wagner 1988). The control task hierarchy is useful in the display design, too. During the detailed design, the displays of the automation system can be structured according to the tasks. This gives natural surroundings to the operator. With a proper display hierarchy the need for change between displays is minimized.

The control tasks have to be analyzed in detail. During the allocation, an overall view is suitable for the definition of WHAT the necessary functions are. But when concrete displays are concerned, the question is: HOW the functions should be implemented? For useful displays, at least following factors should be analyzed in each control task:

- Goals. Why this task is to be performed and what results are expected from this task?
- Input information. What measurements or data is needed for recognizing the state of the process?
- Control actions. What control actions should be performed in order to achieve the goals?
- Algorithms. How the control actions are concluded from the input information?
- Connections. How the task is connected to other tasks and processes?

The content of the displays can not be decided without the operators because the information content of a task is not evident to the design engineer. The operators should take part to the design early enough in order to have real influence on the displays. Users should not be seen merely as a reference of design but as an intentional subject who is able and willing to improve the production process (Launis 1990).

A copy of an overview display is included in appendix 1. The display was done for DAMATIC XD control system by VALMET Automation Inc. Co-operation between automation engineers and paper machine operators was close when these displays were designed. These kinds of general information displays are quite rare partly because of technical constraints of earlier process control systems, partly because of designer's inadequate experience on the operation in practice. Extensive use of trend curves and missing many pipe lines makes this display different from the normal ones.

CONCLUSIONS

The hierarchical task structure supports the design process. The same model can be used in different design phases and during the design the model becomes more accurate and concrete. The model of the whole guarantees that the information exchange between different designers, between design phases and between the design and the operation is efficient. The previous solutions form a good base for future design when they are documented in a standardized way. Automation design will profit from exact specifications and organizational design and training will be easier when tasks are specified properly.

The design practice presented in this paper is clearly "top-down" (Rouse, Cody 1988). The design starts by defining objectives which are decomposed into functional requirements. These requirements are then converted into description of tasks and the responsibilities for which are then allocated among humans and machines. But user

participation is essential for good design results. Therefore the "bottom-up" processes (Norros, Toikka, Hyötyläinen 1989) are included and the users part in the design is emphasized.

The hierarchical task structure has, however, a drawback. It can easily become the basis for the organizational structure of the production plant. The higher tasks are allocated to the managers and production planners and the lower ones to the operators. Strict organization is not very suitable, for example, for flexible production.

ACKNOWLEDGEMENT

The preparation of this paper has been a part of the project "TAOS - Task Analysis in Control System Design" (1988..1990) which has been financed by VTT, Technology Development Centre and private companies. At VTT there are several projects going on which are developing methods for the automation system design. The practical modelling was done in two Finnish paper mills.

REFERENCES

Crossman E R F W. 1974. Automation and skill. Edwards E, Lees F (ed). The human operator in process control. Taylor & Francis.

Frey P, Rouse W. 1990. Knowledge-based tools for automation decisions. American Nuclear Society's Topical Meeting, June 10-14 1990, Nashville, USA.

Goodstein L P, Andersen H B, Olsen S E. 1988. Task, errors and mental models. Taylor&Francis.

Hacker W, Iwanowa A, Richter P. 1983. TätigkeitsBewertungssystem, TBS. Humboldt-Universität zu Berlin, Ag-Nr:605/48/83.

Harré et al. 1985. Motives and mechanisms.

Kallela J, Tommila T. 1990. Task analysis in control system design. American Nuclear Society's Topical Meeting, June 10-14 1990, Nashville, USA.

Launis M. 1990. Designing of workplaces in the current design processes. NIVA course on participation in the designing of workplaces, Feb 5-9 1990, Mariehamn, Finland.

Norros L, Toikka K, Hyötyläinen R. 1989. Constructing skill based FMS - lessons for design and implementation. IFAC/IFIP/IMACS symposium on Skill based automated production, Nov 15-17 1989, Vienna, Austria.

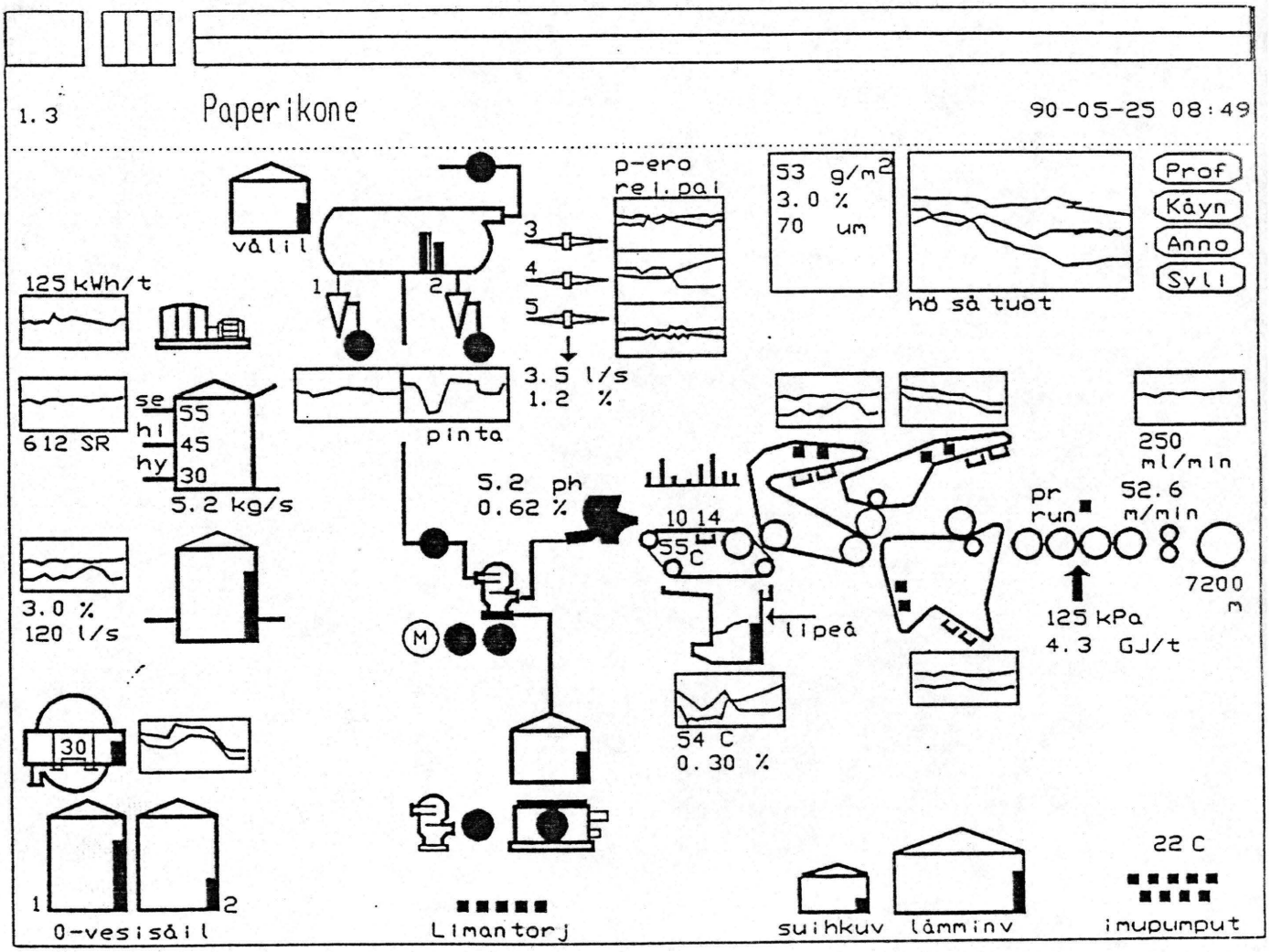
Ranta J. 1985. Automation systems and control room design; problems and methods. Technical Research Centre of Finland, research Notes 517.

Rouse W B, Cody W J. 1988. On the design of Man-Machine Systems: Principles, Practices and Prospects. Automatica, Vol 24, No 2, pp. 227-238.

Smook G A. 1986. Handbook for pulp and paper technologists. TAPPI, USA.

Wagner E. 1988. The computer Display Designer's Handbook. Studenlitteratur, Lund, Sweden.

Appendix 1.


1.3
Paperikone
90-05-25 08:49
välil
p-ero
rei.pai
53 g/m2
3.0 %
70 um
Prof
Kåyn
Anno
Syli
hö så tuot
125 kWh/t
612 SR
se
hi
hy
55
45
30
5.2 kg/s
3.5 l/s
1.2 %
pinta
5.2 ph
0.62 %
10 14
55 C
250
ml/min
pr
run
52.6
m/min
7200
m
125 kPa
4.3 GJ/t
liipeä
3.0 %
120 l/s
54 C
0.30 %
30
0-vesisäil
Limantorj
suihkuv
lämminv
22 C
imupumput

Cognitive Approaches to Process Control

SIMULATION OF PROBLEM SOLVING DEVIATIONS BY A COGNITIVE MODEL

P.C. Cacciabue, F. Decortis, B. Drozdowicz

Commission of the European Communities
Joint Research Centre Ispra
VA 21020
Italy

Abstract

In this paper, several cases of deviations of human operator problem solving strategies are analysed in the conceptual framework provided by **COSIMO** (COgnitive SImulation MOdel), a computer-based model of an operator.

These deviations of operator problem solving are simulated by modifying COSIMO parameters. *Cognitive lockup* when an operator becomes fixed on a diagnosis. He/she tries to take into account all known explanation, without being much addressed to the current situation. *Cognitive inertia* when an operator fails to interpret a change in the plant configuration linked with an incident. *Unadapted change* when an operator changes his current line of reasoning on the basis of the most salient event and fails to assess contingent past events upon the change in view.

1. Introduction

A problem tackled by the research in the field of cognitive ergonomics is the development of models of operators who control complex dynamic systems (Hollnagel, Mancini and Woods, 1986). On the one hand, the simulation allows to examine the hypotheses of the model and to test their validity in order to progress in the description, the explanation and the prediction of the operator problem solving and behaviour. On the other hand, such a model, coupled with an intelligent decision support system (IDSS), should point to the areas of operator's weakness and supply appropriate aids in varying circumstances.

The use of cognitive models linked to man-machine interactions problems is recent. Their originalities are in the synergism between Intelligence Artificial formalisms to express some psychological hypothesis (Bonnet, Hoc and Tiberghien, 1986; Cacciabue, et al. 1989). This enables to model as closely as possible some aspects of operator's knowledge, cognitive mechanisms and strategies and in particular deviations of problem solving when operators stress is high. Stress can alter in a significant way the decision making, but deviations are hard to predict because they are often the result of multiple factors (Cooper and Payne, 1988; Goldberger and Breznitz, 1988).

When controlling a complex system, an operator performs several activities (Edwards and Lees, 1974; Bainbridge, 1981; De Keyser et al, 1987): he or she supervises, detects and diagnoses some perturbations, he/she plans actions and executes them in order to keep the system in normal conditions. These activities are very demanding regarding cognition. The operator recognizes some patterns, uses his memory, makes reference to situations which he has previously encountered, predicts future events and anticipates the evolution of specific situations. In emergency or critical situations, operators may be under great stress and have to find an appropriate solution in a very limited time. When the level of stress is very high, the operator is likely to show premature closure (Janis, 1983), that is terminating the problem solving without generating all the alternatives and without seeking all the information available. A high level of stress may reduces the operator problem solving capacities, especially when the problem comprises numerous competing values. The operator's

attention and perception may be degraded, resulting in narrowing the range of perceived alternatives, overlooking long-term consequences, ineffective searching for information, erroneous assessing of expected outcome and using oversimplified decision that fail to take into account some values implicated by the hypothesis chosen.

In this paper by deviations of problem solving, we will intend a progressive de-coupling between the plant evolving in time and the operator ability to perceive and interpret an incident, to integrate new events that modify dynamically the problem to be solved and then to act appropriately on the system. Deviations frequently observed in dynamic and complex environments are fixations, that is failure to revise a situation assessment as new evidence comes in (De Keyser and Woods, 1989, Woods, 1984). Here we shall look at two aspects of fixations : failure to shift from explanations when a change in the plant configuration has occurred and failure to shift from explanations even after an action has been performed.

These fixations are difficult to identify when they occur. Sometimes, some actions performed by the operator or some verbal exchanges with others operators may indicate that they are currently locked at one level of explanation. Often they become manifest lately in the evolution of the accident when evidence indicates that operators have not succeed in updating their view of the world (Masson, 1989).

This type of simulation of operator problem solving deviations raises several questions which are far from being solved. However, we will try in this paper to answer some of these questions by focusing on the problems of :

- the description of some deviations of operator's behaviour and how they seem to be produced,
- the conditions under which the operator model starts to deviate from an expected behaviour hat is the parameters which need to be modified within the cognitive model in order to simulate these deviations.

Various parameters are responsible of constraining and tuning the behaviour of main parts of the model, allowing it to reproduce subtle variations in the simulation of the overall model performance. Some of them will be explained in the paper.

The first part of this paper will be dedicated to the description of the cognitive model. In particular, the main parameters which will be modified by deviations of problem solving will be discussed. Simulation of problem solving deviations will be presented in the second part as well as the results obtained.

2. Description of COSIMO

COSIMO is a cognitive simulation model of a human operator controlling a complex environment. The description of COSIMO will be divided in a structural description, with the main components and in a functional description that is how the model works.

2.1 Structural description

The operator model presumes a dual information processing architecture comprising a limited, serial but computationally powerful *Working Memory (WM)* and a virtually limitless *Knowledge Base (KB)*, containing both declarative and procedural knowledge structures (Reason, 1987). These knowledge structures have to capture the experience gained by the operator during both his work and formal and practical learning.

The basic assumption of this cognitive model concerns the "cognitive primitives" or mechanisms by which stored knowledge units are selected and brought into the working memory for processing. Two primitives have been identified :

- the Similarity Matching (SM) primitive matches the perceived cues coming from the work environment with corresponding diagnostic cues described in the knowledge base. In other words, the SM compares perceived data coming from the environment (external cues) and cues provided by prior processing (internal cues) with previously experienced cues contained in the knowledge base.
- the Frequency Gambling (FG) primitive solves possible conflicts between partially matched hypothesis selected by the SM in favour of the more frequent, already encountered and well known incident scenario (Reason, 1986).

These primitives act so as to minimise cognitive strain and to maximise the chances of immediate information processing, which is often recognised as the most common problem solving strategy of operators in emergency situations (Reason, 1986). The problem solving based on the SM-FG primitives is therefore defined as an immediate information processing by contrast with a deep reasoning. Switching from immediate information processing to deep reasoning is assumed to be possible only when sufficient time and attentional resources are available and when the stress is low.

For our current problematic we are looking at deviations of immediate information processing. In fact in critical situations the operator may be extremely stressed and has to find an appropriate solution in a very limited time. We thus assume that in such a situation no sophisticated reasoning is performed and the operator resorts to previous experience on similar cases.

It is argued therefore that the role of the WM is : (1) to assert the external and internal cues to match with similarities in the KB; (2) to decide whether the hypothesis activated for explanation provided by the SM and FG primitives is appropriate and if not (3) to reiterate the search with revised cues.

Knowledge Base

Operator knowledge is represented by means of frames. The frames currently represented in the model are called rules-based-frames (RBF). A RBF is a packet of cues configurations and instructions for diagnosing and recovering from a particular situation. Each RBF contains the following elements :

- an index label identifying the type of accident, (loss of power or steam generator tube rupture) ;
- a subjective frequency weight according to the relative number of times the operator has encountered this situation in the past ;
- the elapsed time since the start of the event;
- a set of attributes and attribute values corresponding to expectations about associated symptoms appearing on the control panel, e.g. "the pressure of water in the first steam generator" (attribute), "is high and increasing" (attribute value) ;
- a plan of actions to resolve the problem under development. They are the recovery actions that are appropriate for the situation.

Dynamic representation of knowledge

A dynamic representation of knowledge has been defined in order to meet the evolution of the plant. The experimental plant with which COSIMO interacts is the Auxiliary Feedwater Systems (AFWS)

which is a system operating in nuclear power plants for supplying feedwater to the steam generator during accidents and conditions of plant shut-down (Cacciabue et al., 1990).

This dynamic representation of knowledge contain information at several time instants, i.e. :

- the evolution of accident;
- values of symptoms at current time, and which hold over time;
- new symptoms which were not yet been observed;
- values of symptoms that have changed over time;
- actions at current time.

2. 2. Functional description

COSIMO uses a set of cognitive activities which could be applied in different ways during the problem solving evolution.

Cognitive filter

The communication between the plant and the cognitive model is done through a cognitive filter which selects among the large number of data produced by the environment those which are currently perceived. This perception is guided by a salience criteria which is a function of two parameters, physical salience and cognitive salience, namely their rating on physical dimensions and their context subjective significance.

Physical salience reflects physical properties of cues : type of indicator, position in the control panel, light or sound intensity of alarms.

Cognitive salience expresses the subjective familiarity and significance of a cue. For instance, a particular indicator may be checked by the operator because he knows that it is particularly relevant for the current task and the possible perturbation.

Numerical data coming from the simulated plant are translated in semantical ones by the interpreter using reference distributions.

Diagnosis

Once the cues have been filtered, they are matched with the knowledge base through the Similarity Matching agent. The number, form and quality of these cues determine the matching, i.e. the more pertinent the calling conditions, the less numerous and the better supporter will be the hypothesis selected from the knowledge base. Not all the knowledge base is accessed. Once a hypothesis is activated in the working memory, it will govern a hypothesis driven information processing, i.e. all cues are priority matched in the light of this hypothesis.

In case of ambiguity, either in the interpreted data or in the symptoms represented in the KB, more than one explicating hypothesis may be brought to mind by the Similarity Matching. The Frequency Gambling primitive has to solve conflict between partially matched hypotheses in favour of the more frequent one encountered in the past.

Matching and frequency values are combined to deliver a support to each hypothesis. This combination may take various forms according to the level of expertise of the operator at his current cognitive state. For instance an expert operator may always favour the frequency value while a novice may give higher priority to the similarities between situations.

The hypothesis selected is the one obtaining the greatest support from the combination of similarity weight and frequency weight.

The diagnosis is intended as a dynamic process which may be carried over during an extended period of time, which can be interrupted for some alternative plan or new diagnosis and possibly restarted at a later stage.

Hypothesis evaluation

An evaluation stage aims at deciding if an hypothetical explanation can be trusted or if it has to be rejected. When the hypothesis is rejected, a new matching is initiated between the already perceived cues from the environment and all the others known cues from the knowledge base.

Execution

Once an explanation has been selected, the model may execute the corresponding recovery actions compiled in the rule base frame. The diagnostic symptoms and recovery actions are listed together in the frame .

Short-cuts and opportunistic activities

Several researches have shown that operator activities - filtering, diagnosis, planning - are not necessarily linearly concatenated. On the contrary, depending of their level of expertise, operators are able of short-cuts (Rasmussen, 1986; Leplat, 1985; Reason, 1987). In particular, operators may operate forward and backward motion between the several stages. They may fix on a first diagnosis, do some actions and if not satisfied of the effects obtained they may modify their initial diagnosis. Acting on the system may be guided to delay the system evolution in order to gain some times for thinking or to test the system, helping then the interpretation. Cognitive activities cannot then be strictly ordered on the temporal axis but should intervene in an opportunistic way.

A blackboard architecture (BB) has thus been implemented in order to facilitate this opportunistic characteristic of cognitive activities. A description of the implementation of COSIMO within the BB architecture can be found in Cacciabue et al. (1989).

3. Description of parameters

Table 1 shows parameters which should be modified to simulate deviations.

insert table 1 about here

Filter threshold

This parameter allows to modify the set of data to be taken into account by the model. Decreasing the filter threshold extends the field of attention. More cues are then collected. This does not mean that only pertinent cues are taken into account. The operator can be for instance attached to unimportant details. Increasing the filter threshold means to narrows the field of attention which can be positive when only salient cues are taking into account but it can be negative when salient cues are discarded.

Cognitive vs physical salience

Greater importance can be given to the physical or to the cognitive salience. We assume that in situation of stress, the operator will give priority to the external appearance of an information rather to its pertinence for solving the current problem.

New and changed cues

New and changed cues traduce events evolution on time. In COSIMO, different weights can be given to newly appeared cues or to changes that have occurred from one step to the next during the problem solving process. Using these parameters weights the model enables to modify the amount of information to be interpreted.

Cognitive inhibition.

According to a high confidence in some process evolution, component reliability or current component status, the model may decide not to consider a specific set of data. Cognitive inhibition can be a very economic process because it allows to neglect some unimportant or not pertinent events and to limit the information search at certain moment. But it can be source of errors too. For instance, the operator under stress can be induced to ignore a part of the system which is currently in a failure condition. When very stressed, this can be amplified as the operator will be ever less prone to revise his conviction about inhibited events.

Interpretation threshold

By this parameter is possible to modify the model capacity to interpret variations of process data. Increasing the interpretation threshold may signify that little variations are judged as stable states. Moreover the variation may or not be considered on time. In the relative threshold, the variation can be interpreted in time. An increasing of temperature may signify something completely different if the operator knows that this variation has occurred in three minutes or in two hours. In the absolute threshold the variation can be interpreted independently of time.

Stress can modify these factors by ignoring significant time periods of variation of specific quantity.

Similarity vs frequency

Differential weights can be given to the similarity or to the frequency. It is the case when an increased pertinence is given to the similarities between two situations rather to the occurrence of one situation in the past. In this way the result of the diagnosis can be modified.

Extended and temporal search

The search in the dynamic knowledge base can be done in two directions. Search can be limited to current times according to the evolution of the plant, or it can be very extend searching for frames at all times periods. On the other hand, search can be limited at one line of reasoning already activated then limiting itself at the current scenario or it can be extended at all possible scenarios.

Evaluation threshold

Allow to modify the confidence in the current hypothesis. As the hypothesis selected has the highest support, this latter can however be too low to give the model any confidence in this explanation. An evaluation stage aims at deciding if a potential explanation can be trusted or if it must be rejected.

4. Simulation of problem solving deviations

Three main aspects of deviations of problem solving were examined in the framework of COSIMO. The first one is cognitive lockup, the second is unadapted change and the third is cognitive inertia.

4.1 Definitions

Cognitive lockup

An operator may choose an hypothesis for diagnosis and accords a high degree of confidence. New symptoms are accounted for in the light of the hypothesis activated. Congruent data are privileged and contradictory are discarded. This fixation may occur at any time in the course of the operator control of the plant. It may be very difficult to detect that operators are currently entering in a fixation and then to give them appropriate aids.

Unadapted change

In this case operators may uncritically change his current line of reasoning on the basis of low salient cues and the most encountered in the past or the more recommended without trying to confirm his hypothesis and without making contingency plans. He may adopts new actions which are incompatible with the current state of the plant.

Cognitive inertia

In case of cognitive inertia, it is assumed that operators faced with an incident continue on their initial line of reasoning (then on the first activated hypothesis), ignoring salient information. Salient information can be represented by new or changed cues. This means that the operator may collect many information, having thus a very extended span of attention but being disturbed with many details.

New and changed cues are accounted for in the light of the first hypothesis activated. Recovering actions may correspond to this hypothesis and being executed when the hypothetical explanation is obsolete regarding the evolution of the system. De-coupling between the operator understanding and the evolution of the system may be always greater.

4.2. Description of COSIMO problem solving deviations

Four cases have been simulated with COSIMO as showed by fig. 1 and 2.

Case A: it describes the expected behaviour of the model when no particular incident occurs at the AFWS.

Case B: describes the case of a cognitive lockup with a change of parameters which occur during the simulation of the cognitive model.

Case C: describes an unadapted change whereby the model changes view after a new plant failure, but does not take into account past events related with current evolution.

Case D: describes the case of a cognitive inertia which causes the persistence of the first explanation even after a new plant failure.

In figures 1 and 2 the various agents describing the cognitive processes acting at different times are represented. Figures 3 and 4 indicate respectively the distribution of filtered variables and the similarity matching values during the four simulations.

These cases and the related figures will now be described in some details.

insert fig. 1, 2, 3 and 4 about here

Expected behaviour : case A

No human error or plant failure occurs in this case, the simulation indicates the following characteristics:

- the model takes into account an expected set of indicators either logical (configuration of the AFWS, position of valves, etc) or physical (values of quantities like temperature, pressure, etc) (fig. 3);
- the value of similarity matching throughout the simulation is equal to 1 (fig. 4);
- corresponding actions are executed according to an optimum solution of the control procedure.

Cognitive lockup : case B

In case of cognitive lockup, the model becomes fixed on a diagnosis (fig.1). The following observations can be made :

- when the model is entering in a stage of fixation, more cues are filtered (fig.3), among others those which may not appear related to the current problem. In this way, the fuzziness of the model understanding results in a search for a greater amount of information. When the model becomes locked-up, it does not interpret any new or changed cue which could allow the shifting in view;
- when the model enters in a stage of fixation, it tries to take into account all possible known explanations, without being much addressed to the current situation, after a while, an arbitrary explanation is chosen and it will not succeed at revising the situation assessment as new evidence comes in ;
- the matching is always lower than the expectation (fig. 4), the number of agents called into action becomes drastically smaller and they are repeated without a possibility of a real resolution of the problem (fig. 1).

In COSIMO this deviation was obtained by :

- expanding the model field of attention, i.e. decreasing the filter threshold;
- limiting the model consideration for new and changed events, i.e decreasing weights of new and changed cues;
- decreasing cognitive salience weight and increasing physical saliencweight;
- reducing the interpretation of small variations by increasing the interpretation threshold;

- increasing the weight of the frequency relative to the matching;
- increasing the evaluation threshold.

Unadapted change : case C

At the beginning of this simulated case, COSIMO succeeds in identifying the correct explanation through the SM and FG primitives. It has then to execute the corresponding actions at the right moments. In this case, after 900 seconds, the driver introduced a failure in the system by causing a leakage on the inlet of one of the four steam generators. COSIMO again succeeds in identifying this incident. However, the new situation is tackled without considering the history of the sequence and, in particular, considering the fact that for the first period the accident was of different nature. As far as the sequence of actions is concerned it results that the activity related to the control are well performed but those linked to the state of the plant show an unsatisfactory outcome. In particular, it can be noticed (fig. 4) that the matching for the period following the new event is rather poor while the set of filtered variables (fig. 3) remains almost constant because the model has a clear understanding of the situation.

Cognitive inertia : case D

This is the same case as before up to 900 seconds, but at this time COSIMO does not succeed in noticing the change from the indications from the AFWS. Even if the abnormal behaviour of AFWS is demanding a greater cognitive effect of variables collection and interpretation (fig. 3), the high satisfaction of the model about its current hypothesis, does not allow any attempt to update the reasoning in the light of the disconforming cues. This leads to a poor matching of the selected cues with the expectation. Although with a very low similarity matching value, the model remains confident in its explanation and does not succeed at revising its situation assessment. The model continues acting on the system in the light of the obsolete explanation and a progressive de-coupling between the model understanding of the plant and the evolution of the plant is then observed.

In COSIMO this deviation was obtained by :

- expanding the model field of attention by decreasing its filter threshold;
- decreasing the new and changed symptoms weight;
- decreasing the weight of cognitive salience;
- increasing the weight of frequency relative to the matching ;
- decreasing the evaluation threshold and thus making the model less cautious about mismatches between hypothesis expectations and relate cues.

5. Conclusions

In this paper we have presented a COgnitive Simulation MOdel (COSIMO) of human behaviour during the management of a plant. COSIMO represents a real step forward with respect to cognitive theories and modelling as it contains already the transformation of a psychology theory in a computerised framework, i.e. the gap between modelling and simulation has been bridged. Indeed by COSIMO the response of a cognitive behaviour to a variety of situations can simply be obtained by allowing a dynamic interaction of plant and human model, characterised only by their input data. Modifying these input it is thus possible to simulate a vast variety of biases, deviations as well as errors of the human cognitive and behavioural responses during the management of a plant. In particular, the most significant deviations of problems solving strategies under stress conditions have been studied in

this work, with quantitative information concerning the amount of cognitive activity and workload evaluated by the operator model during the control sequence.

The evaluation of the validity of the hypotheses of the model and the availability of real data are amongst the open problems for this kind of research. We are aware of such problems and we are currently organising a research plan in this direction. We are, however, confident that the theoretical work of modelling so far performed and its computerised implementation represent a promising achievement in the direction of the formulation of better tools for the accurate consideration of the human behaviour in the control loop of working environments.

Acknowledgement

The authors would like to thank Michel Masson for his essential collaboration at COSIMO development.

References

Bainbridge, L. (1981). Le contrleur de processus. *Bulletin de Psychologie,* 34, 813-832.

Bonnet, C., Hoc, J.M. et Tiberghien, G. (Eds) (1986). *Psychologie, Intelligence Artificielle et Automatique.* Bruxelles, Mardaga.

Cacciabue, P.C., Codazzi, A., Decortis, F. (1990). The control of AuxiliaryFeedwater System for simulating man-machine interaction. MOHAWC Workshop. Lige. Mai 1990.

Cacciabue, P.C., Decortis, F., Mancini, G., Masson, M., Nordvik, J.P. (1989). A cognitive model in a blackboard architecture : synergism of AI and psychology. *2nd European Conference on Cognitive Science Approaches to Process Control.* Siena, Italie, 24-27 octobre.

De Keyser, V., Woods, D.D. (1989). Fixation erros in dynamic and complex systems. In *Advanced Systems Reliability Modelling*, Colombo, A.G. and Micenta, R. (Eds) Kluwer Academic Publishers, Dordrechts.

De Keyser, V., Decortis, F., Housiaux, A., Van Daele, A. (1987). Les communications hommes-machines dans les systmes complexes, Action Fast, Bruxelles, Politique Scientifique.

Edwards, E., Lees, F. (1974). *The human operator in process control.* London,Taylor & Francis

Hollnagel, E., Mancini, G. et Woods, D.D. (1986). *Intelligent Decision Support in Process Environments.* Nato Asi Series, Vol. 21. Springer Verlag.

Janis, I.L. (1983). Decision making under stress. In L. Goldberger, S. Breznitz (Eds). *Handbook of stress. Theoretical and Clinical Aspects.* New York, Free Press.

Leplat, J., (1985). *Erreur humaine, fiabilit humaine dans le travail.* Paris, Armand Collin.

Masson, M. (1989). Fixation errors : failures to revise situation assessment in dynamic hazardous systems. A modelling approach. Presented at "Towards methods for registration and analysis of near-misses". 6-8th september, Eindhoven, The Netherlands.

Rasmussen, J. (1986). A framework for cognitive task analysis in systems design. In E. Hollnagel, G. Mancini and D.D. Woods (Eds). *Intelligent Decision Support in Process Environments.* Nato Asi Series, Vol. 21. Springer Verlag.

Reason, J. (1986). Recurrent errors in process environments: some implications for the design of Intelligent Decision Support Systems. In E. Hollnagel, G. Mancini and D. D. Woods (Eds.), *Intelligent Decision Support in Process Environments,* NATO ASI Series, Springer-Verlag, Berlin.

Reason, J. (1987). Cognitive aids in process environments : prostheses or tools ? *International Journal of Man-Machine Studies*, 27, 5&6, 463-471. London, Academic Press.

Woods, D.D. (1984). Some results on operator performance in emergency events. D. Whitfield ed. Ergonomic Problems in Process Operations. Inst Chem. Eng. Symp. Ser.90.

Parameters	Definition	Default value
Filter threshold	Salience value is compared to this threshold in order to define the variables which pass the filter	0.75
Physical salience weight Cognitive salience weight	Modify the influence of the cognitive or the physical salience in the computation of the cue salience	2 1
New cue weight Changed cue weight Symptom weight	Weights used to qualify either new cues firtsly appeared, changed cues from a previous state	1 0.9 1
Cognitive inhibition	Set of quantities to be inhibited in the cognitive salience assignment	nil
Interpretation threshold	Defines sensitivity (relative vs absolute) in accounting for the variation of quantities to be interpreted	10/60
Similarity weight Frequency weight	Modify the influence of similarity or frequency in the selection of the hypothesis. Used to compute the final combined support value.	1 1
Extended search Times search	Indicate part of KB and times extends which have to asserted to select the set of hypotheses	All cases of EAM KB 0
Evaluation threshold	Degree of confidence for one hypothesis	0.7

Table 1 : COSIMO paramaters and default values

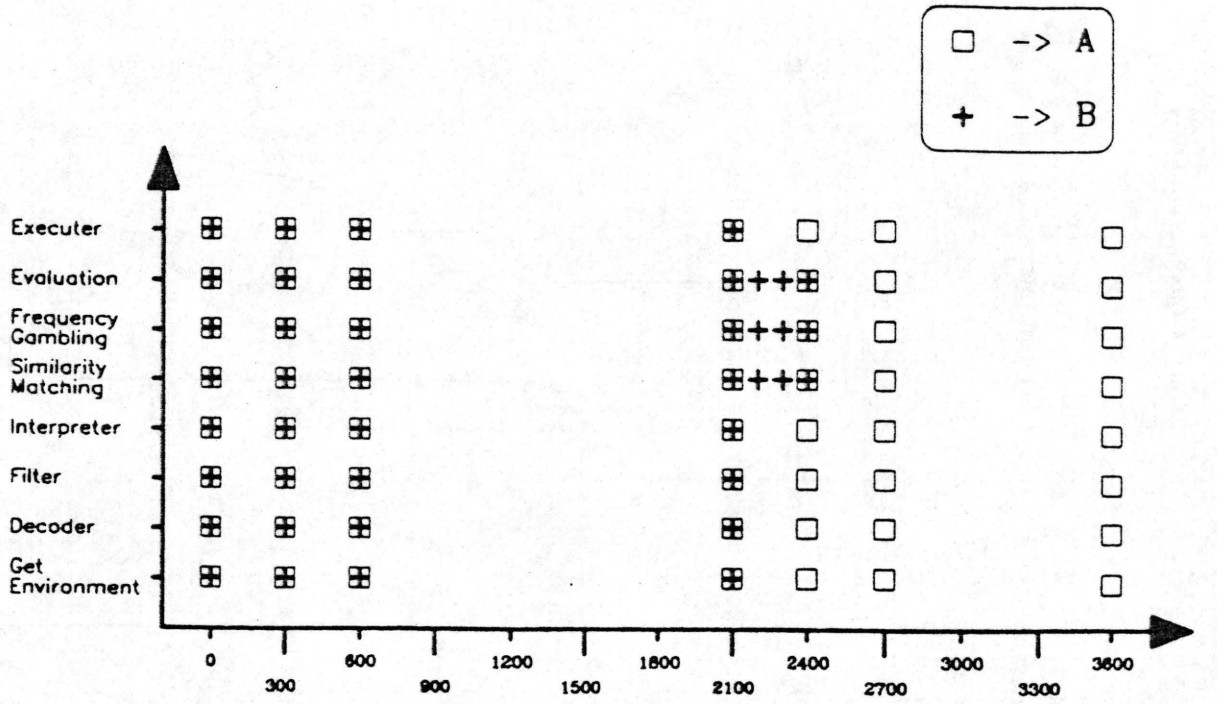


Fig. 1 : Agents describing the cognitive processes in cases A-B.

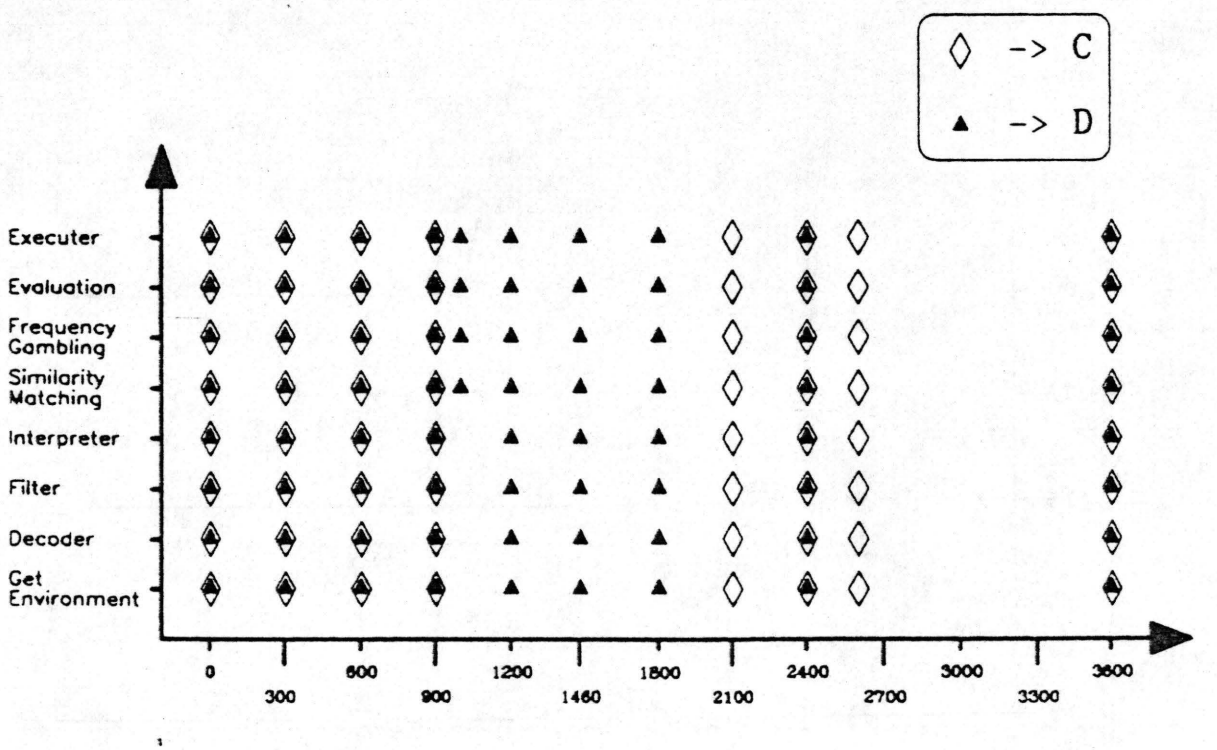


Fig. 2 : Agents describing the cognitive processes in cases C-D.

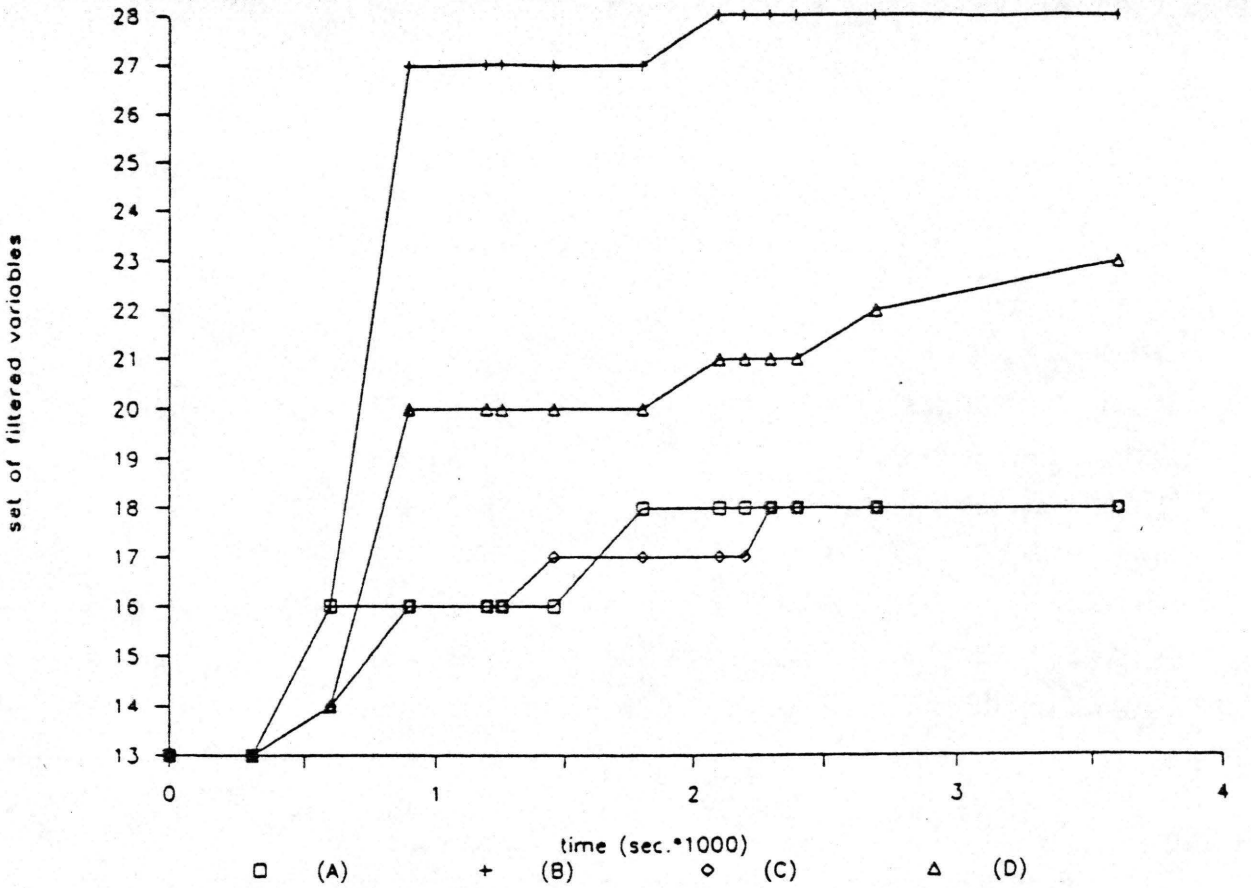


Fig. 3 : Distribution of filtered variables for test cases A-D with COSIMO

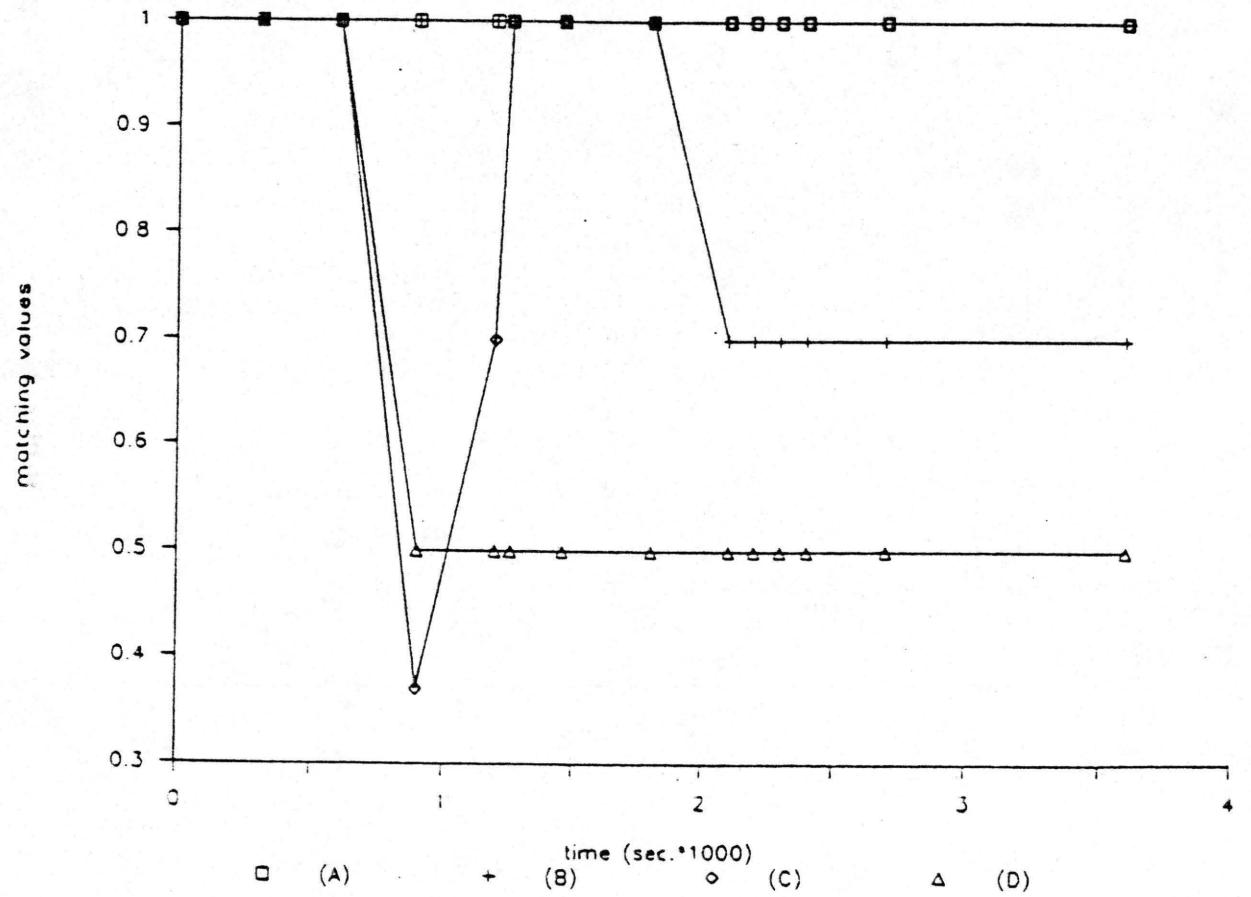


Fig. 4 : Distribution of matching values for test cases A-D with COSIMO

MODELLING PROCESS OPERATOR KNOWLEDGE FOR THE DESIGN OF COMPUTER SUPPORT TO DECISION MAKING

Jean-Michel HOC & Renan SAMURCAY
Researchers at CNRS

CNRS - UNIVERSITE DE PARIS 8
URA 1297 "Psychologie Cognitive du Traitement de l'Information Symbolique"
Equipe de Psychologie Cognitive Ergonomique
2, rue de la liberté - F-93526 SAINT-DENIS CEDEX 2

EXTENDED ABSTRACT
ECCE5. Urbino (Italy), September 3-6, 1990

ABSTRACT

This paper describes work carried out within the framework of a research program dealing with computer support needs specifications in industrial process supervision (blast furnace). The study presented here was aimed at defining the structure and content of operator process knowledge representations that can be used in enhancing human-computer interaction. The Representation and Processing System (RPS) used by the operators was inferred from off-the-job interviews. RPS implementation was observed in a process simulation with a thinking-aloud technique. In blast furnace supervision where the process is continuous and the latencies very long, the RPS is mainly causal, but integrates functional and topological features. The components of the RPS can be modelled as schemas, corresponding to schematic representations of unobservable phenomena inside the plant. These representations proved to be insufficient however to support planning activities (especially prognosis), which are crucial in this kind of situation. A theoretical model of the process was elaborated in collaboration with the technical services of the factory as an enrichment of the RPS to improve planning strategies. Model-RPS comparison shows the RPS to be a deformed version of the model which is mainly influenced by operator repertory of interventions on the process, hence restricting actual diagnosis and prognosis activities. Implications on the design of training sessions and computer support to diagnosis/ prognosis activity are discussed.

This research work was supported by the European Coal & Steel Community (ECSC), the French National Research Institute for Safety & Health (INRS), the French Ministry of Research, and the SOLLAC-Dunkirk Iron & Steel Works.

INTRODUCTION

This paper surveys results from a three-year research program, designed to identify process operator needs for purposes of computer support to decision making and training. It presents one aspect of this program - operator knowledge representations of process operations - as related to the design of a common language usable in human-computer, as well as human-human, cooperation. The study deals with the supervision of a complex and continuous process with long latencies (a blast furnace), where planning, schematizing, diagnosing, and prognosing are crucial activities (Hoc, 1988a). More generally, this study addresses issues raised by the supervision of a dynamic environment where state-transformation models are not applicable, since operators must deal with continuous evolutions (Hoc, 1989a).

In contrast with most work on Human-Computer Interaction (HCI), which aim at improving available software tools, the present study is located upstream in the design of computer support to supervision. The primary focus is how to help operators in their main activity, which is not using software, rather than how to help them to use software.

In continuous processes with long latencies, cognitive analysis mainly pinpoints issues concerning operator knowledge representation. Diagnosis, decision making, and planning strategies in process environments have received thorough descriptions (Bainbridge, 1984; Rasmussen, 1986; Amendola et al., 1988). More attention has been devoted here to modelling knowledge bases underlying the strategies than these strategies *per se*. Knowledge representation is crucial to the

design of human-computer cooperative dialogues. To enable each partner to comprehend his/her partner's reasoning, a common basis for communication needs to be established between them (Woods, 1986; Roth et al., 1988).

A stepwise approach to operator knowledge representation was adopted here, because it provides better access to the operational representations that operators actually use (Leplat & Hoc, 1981). As a first step, the Representation and Processing System (RPS: see Hoc, 1988a) used by the operators was inferred from an interview technique (Hoc, 1989b). Second, the implementation of this RPS was observed during a simulation of the process, by a thinking-aloud technique (Hoc, 1989c). From an ergonomical standpoint, criteria were needed to assess the efficiency of the various types of representations used by different operators, in order to design support for the most efficient ones. However, even the best RPS appeared insufficient to support prognosis and planning activities. This led to the third step in which the RPS was enriched and transformed into a theoretical model with the help of the technical services of the factory. The RPS was then compared to this model, in order to identify training needs and design an assessment experiment.

1. THEORETICAL FRAMEWORK

A minimal description of the process and the task is necessary here. More details can be found in Hoc (1989b or c).

A blast furnace (BF) is a reactor which reduces iron oxide (ore), by means of carbon monoxide (CO) at an appropriate temperature (about 500°: solid state). Alternated layers of ore and coke (providing carbon) are automatically charged into the top of the BF (like a vertical cylinder). On the bottom, hot air (about 1200°) is blown and cast iron collected (about 1500°). CO and heat are produced by coke combustion, part of the heat being obtained by coal powder and oxygen injections. Apart from iron, a BF manufactures secondary products such as gas and slag (unreduced oxides like alkaline). The presence of two opposing phenomena - the slow descent of charge (4 m/h) and the rapid ascent of gas (1 m/s) - creates delayed feedback in causal relations: transformations in the top affect transformations in the bottom, and *vice versa* (with different time lags).

The study only deals with one type of blast furnace operator - the conductor - who is in charge of the process inside the BF, but is also responsible for the overall plant, in a control room. Operators use about 60 parameters (mainly pressures, temperatures, times, mechanical and chemical analyses of charges and products). Certain parameters, especially predictors, are provided by a computerized model of the process (only valid during normal operations). They can make about 10 kinds of one-action interventions (such as proportion of coke in the charge, blast air flow, injections, etc.).

In terms of the cognitive typology Hoc (1989a) developed to classify process control situations, a BF situation exhibits the most extreme contrast with static state-transformation situations traditionally studied in Psychology and Artificial Intelligence, for the reasons given below:

- the control of the process - the effect of operator action upon the crucial variables of the process - is indirect and delayed, and hence planning is useful (for process operation planning rather than for operator action planning); e.g.: a change in the proportion of coke in the charge produces an effect upon iron temperature 6 to 8 hours later;

- process information - the actual values of the crucial variables - is furnished exclusively by remote variables which are the indicators; inference is thus necessary to diagnose the state or evolution of the process; most of the sensors are peripheral (due to temperature conditions inside the BF);

- the process is continuous - i.e. the process variables are continuous - and there is a straightforward way to discretize it to control the process; thus reasoning mechanisms deal

primarily with evolutions and seldom with states; the charge layers are not identified in the bottom of the BF.

Research has shown that, when dealing with technical devices, operators use several kinds of RPS (Brajnik et al., 1989; Baudet & Denhière, in press), to move from deep to surface knowledge: causal (graph of relationships between variables), functional (goal structure of the plant), transformational (goal implementation), topological (physical structure of the plant). In continuous processes with remote control and indirect access to information, little is known about the kinds of RPS actually used by operators. This investigation of operator knowledge representation is essential for the design of computer support to diagnosis and training. There is a need for a common language in Human-Computer diagnosis, to achieve true cooperation between both humans and computers, not only at the level of outcomes, but also during implementation of the diagnosis strategy itself.

In line with Rasmussen (1986), diagnosis is investigated, in this process control context, as it is related to its goal - decision making. This contrasts with experimental studies which deal with diagnosis itself. There should be considerable interaction between diagnosis and decision making, which cannot be analysed if the two components are studied separately, as is the case, for example, in certain investigations of medical diagnosis (Groen & Patel, 1989). Diagnosis is treated here as a comprehension activity, on the basis of the assumption that level of comprehension attained in work contexts is directly related to the decision making requirements. Certainly one of the aims of our work is to contribute to basic research in Psychology, but another is the fulfilment of the ecological validity criterion.

2. ACCESSING AND MODELLING OPERATOR KNOWLEDGE REPRESENTATION OF THE TECHNICAL PROCESS

The first step in the job analysis was aimed at describing operator activities and their relative importance. Observation of on-the-job behaviour was conducted simultaneously with interviews on the use of tools (especially decision making rules) in the control room (Hoc, 1989b). The data show that the operator activity mainly consists of reasoning mechanisms (hypothesis elaboration and testing, comprehension of the situation via hypotheses, etc...). This prompted us to make a finer analysis of operator reasoning, i.e. diagnosis and decision making.

In order to identify the BF RPS the 7 operators use, each was interviewed on the relationships between observable parameters in the process. The technique consisted of an exhaustive scanning of the parameters, by means of photographs of the tools presenting these parameters in the control room (documents, recorders, computer support, etc...). This technique enabled the operators to access their RPS in a concrete situation (e.g.: the situation in which the record had been photographed) and to report on the relations between the parameters in this concrete situation, before generalizing to other situations. In this way, we were able to circumvent an abstract task, which could increase difficulty of access to the knowledge base, and in particular the operational RPS actually used on the job.

The results reveal the importance of unobservable variables, which act as intermediary entities in expressing causal relationships between observable parameters. 20% to 70% of the relations reported by the operators (43% on the average) concerned these unobservable variables which did not appear in the photographs (Fig. 1 presents an example of an operator simplified influence graph). These variables, termed descriptors, refer to unobservable phenomena of various kinds (thermodynamical, mechanical, chemical, etc...) inside the BF. The values of these descriptors are inferred from observable parameters.

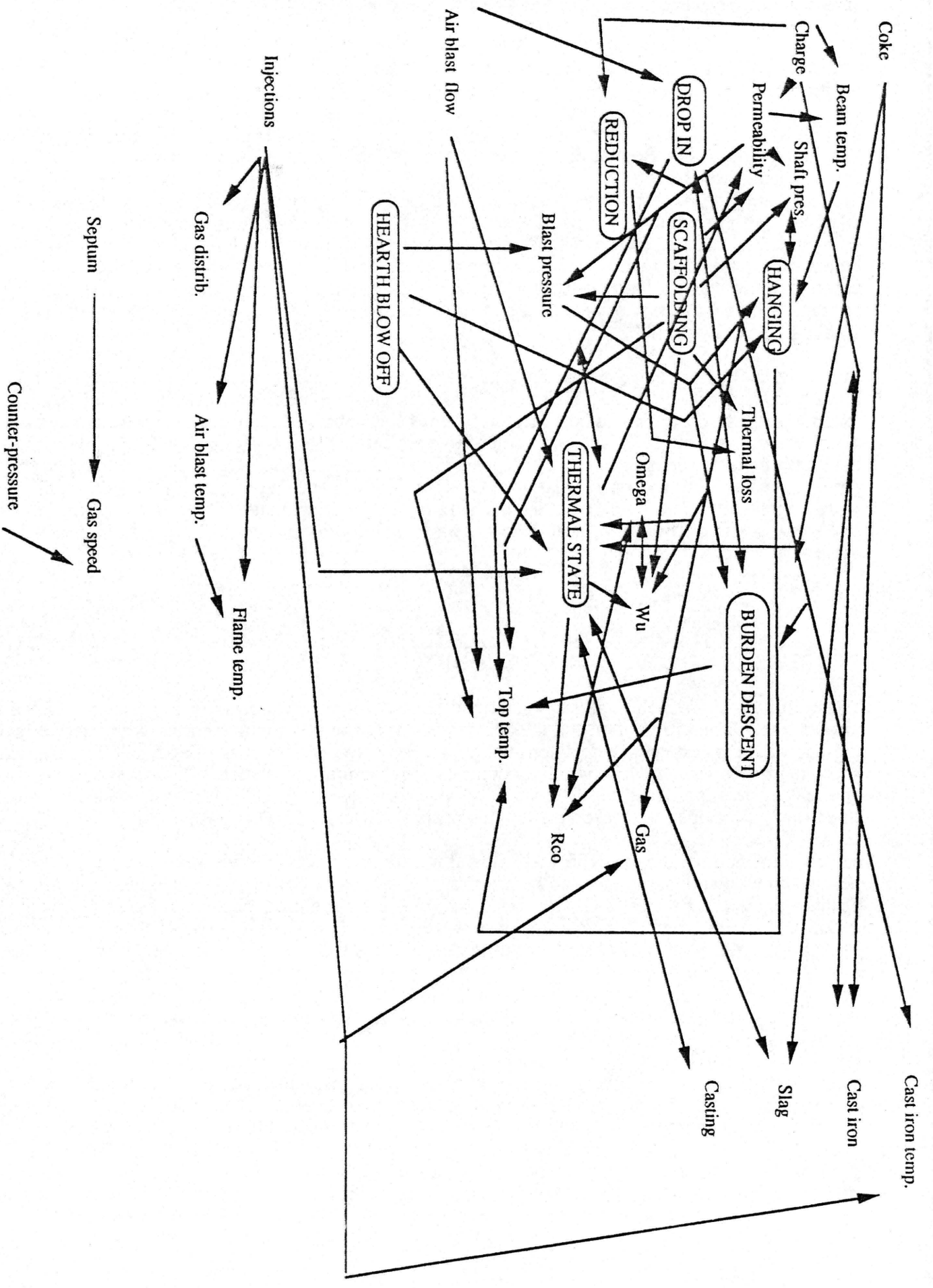


FIGURE 1 : Influence graph of a BF operator. Circled entities are descriptors, the other entities are parameters (after Hoc, 1989b)

Finer analysis of the influence graphs and the reports on decision making rules served to define basic structures involving these descriptors. Each descriptor in fact defines a schema structure (Fig. 2) grouping several kinds of entities:

- determinants: parameters or descriptors having a causal effect upon the descriptor;

- caused entities: parameters or descriptors caused by the descriptor; in order to reach a goal expressed in terms of the caused entities (e.g.: reduction improvement), a subgoal representing a modification of the descriptor can be defined (e.g.: satisfy the symmetry of the burden descent);

- indicators: parameters which serve to evaluate the descriptor (e.g.: leads installed around the BF to provide information on the stock level); even when the descriptor has a causal effect upon the indicator, no operator goal is expressed in terms of modifying the descriptor to modify the indicator (a physician does not mainly treat measles to get rid of the rash);

- actions (under operator or engineer responsibility).

In a context where process operators use a variety of RPS, which need to be associated for efficient computer support (Goodstein, 1983), the hybrid features of the BF operators' RPS is a critical factor. This RPS is mainly causal, but also:

- topological: most of the phenomena represented by the descriptors take place in a defined area of the BF;

- functional: the distinction between caused entities and indicators is linked to the decomposition of goals into subgoals; the phenomena are associated with different functions of the process (heating, reducing, producing, etc...).

This economy of operational knowledge representation needs to be preserved in specifications of computer support to supervision activity.

The implementation of this type of RPS was studied through thinking-aloud protocols (6 operators), in a process simulation which reproduced most of the main process malfunctions (Hoc, 1989c). Efficiency criteria were elaborated on an internal basis, after analysing all the protocols (e.g.: hypothesis formation, testing etc. at the right time). Comparing the most efficient control strategies against the least efficient ones shows that the efficient strategies:

- made more use of the descriptors (24% to 32% of the reported representations vs 11% to 16%);

- expressed more hypotheses in terms of descriptors (62% to 89% vs 32% to 65%);

- based decisions more often on descriptors (27 to 50 vs 0 to 20);

- and processed interactions between descriptors more easily (e.g.: simultaneous hypothesis handling: 6 to 18 vs 3 to 5).

However a number of obstacles to planning and handling of interactions emerged from performance of the group of operators, which on the whole was not very efficient. These obstacles showed where improvements could be made in the RPS inferred from observations of these operators. The empirical data showed that descriptors are important to activity, but the representations were too schematic to be used to diagnose or anticipate evolutions in multiple malfunction situations.

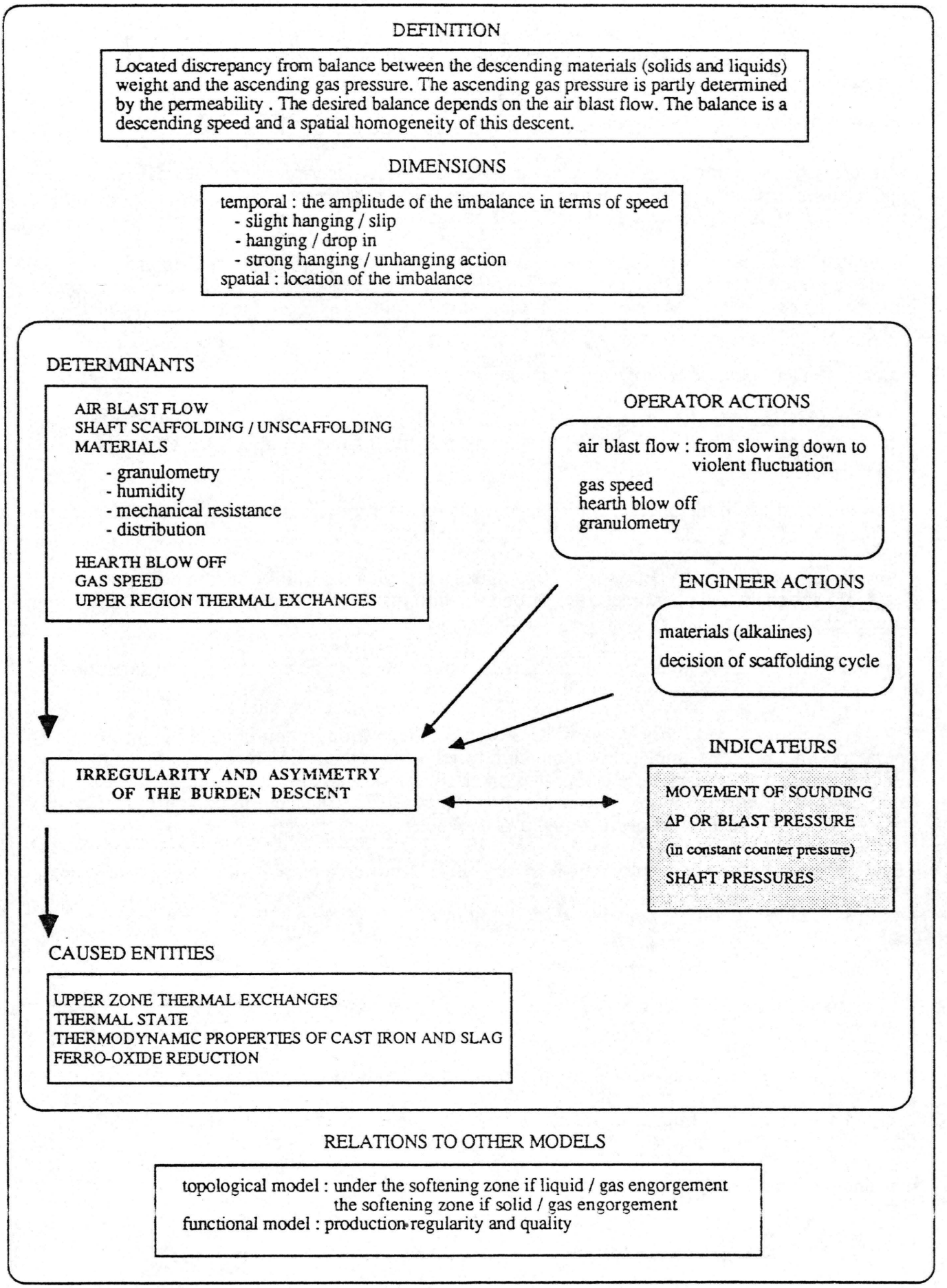


FIGURE 2 : Example of a descriptor schema: Burden Descent (after, Hoc&Samurçay, 1989)

3. ELABORATION AND FIRST ASSESSMENT OF A THEORETICAL TECHNICAL PROCESS MODEL (Hoc & Samurçay, 1989)

A theoretical study conducted in conjunction with the technical service of the factory enabled us to obtain a finer grain decomposition of the phenomena. The resulting overall structure of this theoretical model remains compatible with the empirical ones.

Enhancements in the theoretical model, in comparison with the operator RPS, include:

- better definition of all descriptors in terms of well-identified phenomena, which served to improve the semantics of the descriptors;

- decomposition of composite descriptors into elementary phenomena;

- grouping of different descriptors corresponding to the same phenomena;

- a systematic inventory of relations between descriptors, parameters and actions.

The theoretical model was constructed in two phases. In phase one, only one engineer worked with researchers to improve the accuracy of the model while, at the same time, maintaining its similarity to the operator RPS. In phase two, the model was presented to other engineers for discussion and criticism. Discussion about the model showed that the engineers had some difficulty reaching a consensus. Analysis of these difficulties (in progress) should enable us to generate hypotheses about the potential difficulties operators may encounter during training.

Before using this theoretical model for training purposes and in experiments on computer support to diagnosis/ prognosis, it was assessed against the RPS inferred from interviews, and the implementation of RPS in the simulation. This assessment was based on the analysis of:

- the relative importance of each descriptor in the three constructs (percentages of entities reported in relationship to each descriptor in the model, in the interviews, and in the simulation);

- "short-cuts" (reported relations which should have referred to a descriptor, but bypassed it)

a) relative importance of the descriptors (Fig. 3)

The distribution of relationships in the model (reference distribution) ascribes equal importance to each of the 8 descriptors (about 15% of the relationships).

The empirical distribution obtained from the interviews (interview distribution) is almost similar to the reference one, except for:

- a marked over-representation of 2 descriptors: Thermal State and Burden Descent,

- a marked under-representation of 2 descriptors: Scaffolding and Hearth Blow-Off.

Although all the BF phenomena were involved in the malfunctions, the empirical distribution obtained from the simulation (simulation distribution) differs considerably from the reference distribution:

- the over-representations observed in the interview distribution were also present: Thermal State (here more pronounced) and Burden Descent;

- 2 other descriptors are over-represented: Scaffolding (clearly) and Hearth Blow-Off (slightly); these phenomena were important in the simulation;

- the descriptors under-represented in the simulation were only used by one operator.

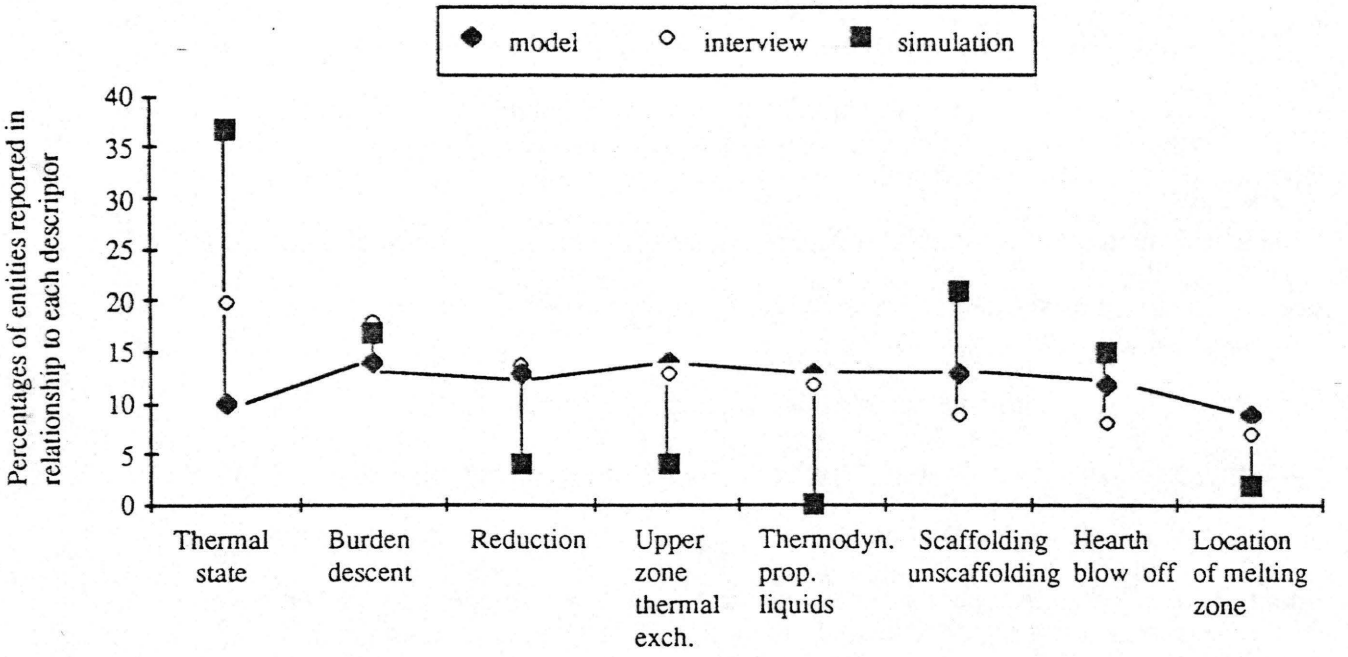


Figure 3: relative importance of the descriptors in the model (black lozenges), the interviews (white lozenges), and the simulation (black squares) (After Hoc & Samurçay, 1989).

Inspecting the properties of the descriptors in relation to their relative importance shows that:

- the over-represented descriptors (in the interviews or the simulation) are the ones which can be handled through direct actions in a number of different ways by the operators to control phenomena (in particular Thermal State which is the most over-represented descriptor);

- rarely used descriptors (in the simulation) correspond to phenomena which operators cannot handle directly (or certain actions are possible and well known by the operators without being under their responsibility).

These properties are indicative of the strength of association between knowledge representation and action. One implication for computer support to diagnosis is that neglected descriptors need to be put back into the action loop to enable the operators to use these descriptors in the case of interactions.

b) short-cuts

The empirical data show that two kinds of short-cuts are used in reasoning:

- Omission of intermediary descriptors.

The over-represented descriptors directly receive causal links which would have passed through the under-represented descriptors. When intermediary descriptors interact, the processing of interactions is thus jeopardized.

- Substitution of an indicator for a descriptor.

When an indicator is related to the overall phenomenon, replacement either can simply be an abbreviation, without serious impediment to evolution representation, or may indicate impoverishment of the representation to numerical variables, without clear representation of phenomena. In the latter case, the sole numerical representation deprives operators of the predictive

power of representations of phenomena, especially in abnormal conditions where the BF mathematical model is not valid. When the indicator is only related to part of a phenomenon, the consequences are greater.

A detailed analysis of these short-cuts is in progress. Analysis is made more difficult in studies mainly relying on verbal reports. An objective analysis has been conducted on words operators use, but the links between words and their actual representations calls for inference on the part of the experimenter. Other types of techniques could yield possible bases for interpretation.

CONCLUSION

This work illustrates a stepwise methodology for defining and testing specifications for advanced computer support to process control before designing a prototype. The challenge is to contribute to the design of a computer support which could actually be used by process operators, in a type of situation where almost no expert systems are usable, because of the lack of analysis of operator activity and needs, prior to the design stage.

When attempting to optimize a man-machine system, technical devices as well as operator knowledge representations need to be enhanced. The methodology we used enabled us to maintain the operational properties of the operator RPS (especially the link between diagnosis and action) and, at the same time, improve it.

Before investigating the usefulness of such a model in supervision, operators must be trained to use the model as a guide in their reasoning. Training operators in the conceptual understanding of the model is not feasible in the short term. Besides, relatively long term investigation is required to describe the dynamics of the learning processes in this type of training.

For the purposes of the ongoing research program, a less ambitious training programme has been planned, which only deals with two aspects of the support system: interface understanding and model familiarization. For both, a set of BF situations will be used to put operators in an active position. The prediction is that operators will modify their RPS when using a diagnostic tool based on the model. This tool should help them restructure local and instantaneous information at a more abstract and schematic level.

Analysis of the results of a new simulation with operators who will take part in the training sessions is now in progress to compare their outputs with the data obtained from these sessions, and test for the changes in strategies during learning. The focus is on learning mechanisms such as learning-by-doing, which should help specify the value of mechanisms of this type in this context.

A ® Hypercard environment, using the descriptor schemas, is being developed to support diagnosis/ prognosis activity at the start of new shifts to aid in:

- interpretation of parameter evolutions,

- access to historical information,

- continuation of previous shift reasoning,

- exhaustive analysis of the situation.

Comparisons, reproducing a representative sample of BF malfunctions, will be carried out on three conditions, defined on the basis of anticipation richness:

(1) a control situation with the tools in use in the control room, where the descriptors are not the rationale for information display;

(2) access to information through the RPS inferred from empirical data,

(3) access through the theoretical model.

This experiment will serve as computer support assessment, and at should provide more details on diagnosis strategies, which are underspecified when the analysis is mainly based on verbal protocols. The experimental technique should enable us to better control information access and protocol interpretation. Moreover, difficulties or errors observed in using computer support could be used as accurate indicators of strategies, as was the case in a Hoc's previous study on expert computer programming strategies (1988b).

REFERENCES

Amendola, A., Bersini, H., Cacciabue, P.C. & Mancini, G. (1988). Modelling operators in accident conditions: advances and perspectives on a cognitive model. In Hollnagel et al., 145-158.

Bainbridge, L. (1984). Diagnostic skill in process operation. **International Conference on Occupational Ergonomics**. Toronto.

Baudet, S. & Denhière, G. (in press). Mental models and acquisition of knowledge from text: representation and acquisition of functional systems. In G. Denhière & J.P. Rossi (Eds.) **Text and text processing**. Amsterdam, North-Holland.

Brajnik, G., Chittaro, L., Guida, G., Tasso, C. & Topano, E. (1989). The use of many diverse models of an artifact in the design of cognitive aids. **2nd European Meeting on Cognitive Science Approaches to Process Control**. Siena (Italy).

Goodstein, L.P. (1983). An integrated display set for process operators. In G. Johannsen & J.E. Rijnsdorp (Eds.). **IFAC**. Oxford, Pergamon.

Groen, G.J. & Patel, V.L. (1989). The relationship between comprehension and reasoning in medical diagnosis. In M. Chi, R. Glaser & M. Farr (Eds.). **The nature of expertise**. Hillsdale (N.J.), Erlbaum.

Hoc, J.M. (1988a). **Cognitive psychology of planning**. London, Academic Press.

Hoc, J.M. (1988b). Towards effective computer aids to planning in computer programming: theoretical concern and empirical evidence drawn from assessment of a prototype. In G.C. van der Veer, T.R.G. Green, J.M. Hoc, D. Murray (Eds.). **Working with computers: theory versus outcome**. London, Academic Press, 215-247.

Hoc, J.M. (1989a). Cognitive approaches to process control. In G. Tiberghien (Ed.). **Advances in cognitive science : theory and application**. Chichester, Horwood, Ch. 9, 178-202.

Hoc, J.M. (1989b). La conduite d'un processus à longs délais de réponse: une activité de diagnostic. **Le Travail Humain, 52**, 289-316.

Hoc, J.M. (1989c). Strategies in controlling a continuous process with long response latencies: needs for computer support to diagnosis. **International Journal of Man-Machine Studies, 30**, 47-67.

Hoc, J.M. & Samurçay, R. (1989). An ergonomic approach to knowledge representation. **2nd European Meeting on Cognitive Science Approaches to Process Control**. Siena (Italy).

Hollnagel, E., Mancini, G. & Woods, D.D. (Eds.) (1988). **Cognitive engineering in complex dynamic worlds**. London, Academic Press.

Leplat, J. & Hoc, J.M. (1981). Subsequent verbalization in the study of cognitive processes. **Ergonomics, 24**, 743-755.

Rasmussen, J. (1986). **Information processing and human-machine interaction**. Amsterdam, North-Holland.

Roth, E.M., Bennett, K.B. & Woods, D.D. (1988). Human interaction with an "intelligent" machine. In Hollnagel et al., 23-69.

Woods (1986). Paradigms for intelligent decision support. In E. Hollnagel, G. Mancini & D.D. Woods (Eds.). **Intelligent decision support in process environments**. Berlin, Springer-Verlag, 153-173.

IN SEARCH OF REALISTIC VISUALIZATION FOR HUMAN-MACHINE INTERACTION IN SPATIALLY ORIENTED PROCESS CONTROL.

D. Mestre, A. Savoyant, P. Péruch & J. Pailhous
Cognition & Mouvement
CNRS & University Aix-Marseille
Marseille, France.

ABSTRACT

Operators are more and more confronted with control processes which present spatial characteristics (remote vehicle guidance, robotics and telemanipulation, ...). A major issue is thus to define the information which should be provided to operators in order that they have a sufficient basis to make safe and efficient decisions. More precisely, an important problem is to define the appropriate content of the video terminal through which the operator has access to the work environment and to the relationships that exist between the tool or vehicle he/she is controlling and the environment.

From the analysis of localization and vehicle control tasks, and more generally from the study of the visual bases of locomotion control, we present the general idea that realism cannot be defined as an a priori given, and that the tentative reproduction on a video screen of the "real" environment does not necessarily lead to good performance. For instance, in studying ship controlling tasks, we came to the conclusion that the control task was crucially dependent on simple visual cues, such as the border lines of the navigation channel and the visualization of the vehicle itself. Also visual motion perception plays a key role in the control of spatial behavior. From this type of study we argue that a fundamental study of the visual bases of displacement control, as well as a better understanding of the nature of spatial frames of reference involved in the control of spatial behaviors are a necessary stage in the definition of more efficient human-machine interfaces.

1. THE PROBLEM

In the short and middle term evolution of the development of teleoperated devices and mobile robots, it is necessary to relativize their autonomy (Uttal, 1990). Human assistance in the use of these new technological tools will continue to play a significant role. Hence, taking into account the problems associated with human-machine-workspace interfacing remains a fundamental question throughout their conception. More precisely, since we are confronted with teleoperated devices, their displacements and manipulations are controlled by a (relatively) static human operator. The definition of the content and format of the available spatial information is thus a major issue. We want to demonstrate here that the realism of an interface cannot be defined as an a priori given. It must be defined in relation to the nature of the control task and in accordance with human capacities.

One important constraint of teleoperation tasks comes from the fact that the device under control is out of the operator's direct sight. Nevertheless, visual control remains the dominant mode of control in most cases, since "direct" visual perception is replaced by images produced by video cameras on-board the mobile device. In some cases though, cartographic information about the mobile's location in the workspace can be added. The choices that have to be made concerning the information to be given to the operator necessarily modify the specific conditions in which the visual control of the mobile's displacement is executed. These modifications as well as the analysis of the difficulties they imply on the operator's part raise fundamental questions. Among them are the nature and perceptual bases of the control of displacement.

It is assumed (Allen, Lunefeld and Alexander, 1971; McRuer, Allen and Weir, 1977) that the overall control task includes a navigation phase (planning of the displacement at long term), a guidance phase (deciding maneuvers to be carried out in relation to the environmental layout) and a control phase (carrying out the desired maneuvers). Concerning the displacement's planning, the location of the teleoperated mobile is a major piece of information that has to be made available to the operator. In this instance, the environment may be known or unknown by the operator (he/she can use a map or may have to build a cognitive representation from the successive displacements in an unknown space). In any case, the information about the mobile's location in space cannot be reduced to simple information about the mobile's position (that is a point-like place on a map). The spatial information must also deal with the mobile's orientation. The orientation can be defined in reference to the mobile's direction

of locomotion (forward-backward, rightward-leftward, upward-downward) or to the direction of gaze of the on-board video camera. In cases where the camera is orientable, these two direction will not always be identical. Moreover, when the mobile is out of sight, the operator tends to code the mobile's position in reference to his/her own spatial frame of reference. He/she knows first whether the mobile is approaching or receding from him/her. This is what came out from experiments carried out around the "CENTAURE" project (developed at Commissariat pour l'Energie Atomique in France): Operators had difficulties when they had to take the robot back to its starting position (Dupont, 1988). When spatial frames of reference are defined alternatively with regard to the operator or to the robot, this may cause a conflict which could result in a right-left inversion.

Concerning the perception of the mobile device's displacement, the static operator lacks kinesthetic and labyrinthine information which is usually associated with self-motion. This renders difficult the mobile's orientation and movement control. These difficulties arise in part from the fact that, in these conditions, depth and distance perception are distorted. Other problems may come from missing or partial ground perception or from the impossibility of perceiving a part of the mobile itself on the video terminal. The video camera itself can introduce problems arising from optical deformations and a limited field of view (in particular missing peripheral vision). It is important also not to dissociate these perceptual problems from "action" problems, in particular at the level of the constraints imposed on the visual exploration when it is mediated by a video system.

Finally, the operator's behavior is usually decisive, especially when "temporal emergency" dominates the situation (as in hostile environments). In situations where rapidity is the key factor, the human-device interface must be compatible with fast corrections (eventually reflex actions) of the operator: the efficiency of the interface will be really evaluated in emergency situations, more than in usual conditions. It appears thus that teleoperation tasks are not "natural" situations. It is equally true that when operators are confronted with them, they certainly use their own experience and skills about displacements through space (such as locomotion or vehicle driving). It is one objective of designers and engineers to render these situations as "natural" (or "ecological") as possible (Mestre, 1988a; Flach, 1990). From that perspective the analysis of the processes involved in the control of displacement (Pailhous and Mestre, 1989) is of fundamental importance for approaching teleoperation problems.

2. SOME RELEVANT STUDIES IN THIS CONTEXT.

Our laboratory does research on perceptive and cognitive processes involved in the control of displacement in humans. Two main fundamental aspects can be distinguished there. On one hand is planning the displacement, including the knowledge of our present location in space, that of our present orientation, that of the goal of the displacement, and the choice of a path to get there. On the other hand is the operational level, the execution of the displacement itself, which concerns both carrying out the planned trajectory and controlling the displacement (for instance keeping the vehicle inside defined boundaries on the appropriate trajectory). The planning and general guidance of the trajectory refer mainly to a "spatial representation" context. The control of displacement will be approached from the point of view of its perceptual (visual) aspects.

2.1. LOCALIZATION PROCESSES.

Originally, applied research (localization in maritime space when approaching harbor areas) has raised theoretical questions concerning the cognitive processes involved in localization tasks. These consist in relating successive viewpoints of a moving observer to his/her knowledge of the environment he/she is moving in (if I see this, I am here; Levine, 1982; Levine, Marchon and Hanley, 1984). In other words, localizing oneself is building connections between visual scenes which depend upon our position/orientation (an egocentric view of space) and a set of spatial relations among environmental elements which are independent of our current position and orientation (an exocentric system of reference, as a map).

Starting from this articulation between egocentric and exocentric frames of reference, experiments have shown some characteristics of the location process. The processing methods observed by measuring various parameters used to evaluate the spatial relationships showed that subjects could use a variety of spatial tools (like cardinal, ordinal and metric relationships among elements) to locate themselves on a map. The results showed how easily the subjects could draw on this repertoire, according to the tasks requirements (Péruch, Pailhous and Deutsch, 1986). At this time, a computerized method of analysis of localization processes is being developed using an interactive graphic device (see figure 1). The subject's task is to discover on a map the position and orientation of a "target-viewpoint". This apparatus enables us to manipulate the visual information he/she can use and to analyze the characteristics of the preferred displacements used to do the task (Péruch and Savoyant, 1990).

Moreover, this task presents some aspects of teleoperation tasks, since a static subject controls the displacement of a mobile on a graphic interface.

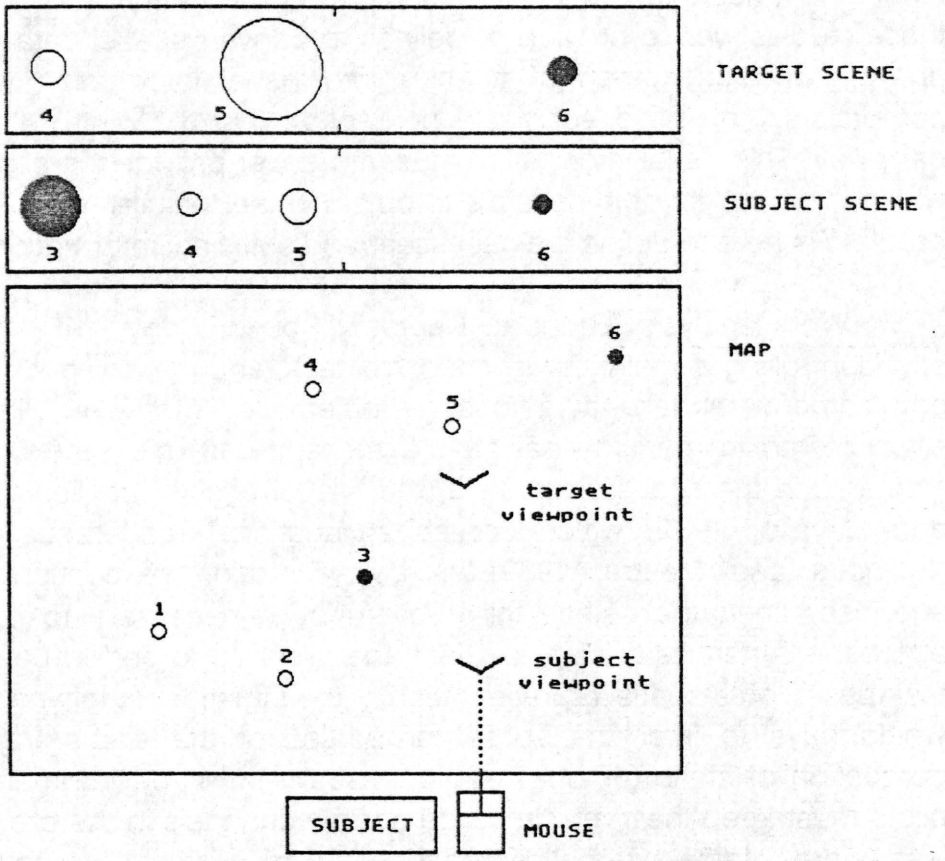


Figure 1. The experimental software creates "maps" (bottom), "target scenes" (top) and "subject scenes". The target scene represents what a subject would see from a "target viewpoint" on the map, with "egocentric" information about the distance and orientation of environmental elements. The task is to find this unknown target viewpoint on the map. The actual "subject viewpoint" is materialized by a graphic cursor on the map, the displacements of which are controlled with a mouse device. The subject viewpoint corresponds to successive positions and orientations of this cursor on the map.

2.2. THE VISUAL CONTROL OF DISPLACEMENT.

Research has also been carried out in relation to the integration and use of visual information for knowing and controlling spatial characteristics of a vehicle's trajectory. We first applied these studies the definition of visual computerized aids

for piloting large ships in harbor areas. Indeed, the development of radio-localization systems, in parallel with the development of micro-computer technology, has enabled the definition of on-board visual aids. The overall brief of these studies was to provide guidelines to convert spatial data (i.e. geographical position and orientation of a ship) into visual information which has to be compatible with the direct visual perception that the pilot gets from the ship's gangway. This visual information must moreover provide the pilot with information which is difficult if not impossible to obtain in the "natural" work space. The major problem is here to define the necessary and sufficient information.

Work analysis (Brouard, Deutsch, Routin, Cuny, 1979) first showed that ship controlling is mainly a visual control task, and that the poor structuring of the environment renders it a priori problematic. Moreover, in the conditions encountered in maritime navigation, the apparent movement of environmental elements in the pilot's visual field, the "visual effects" of displacement (Pailhous and Cavallo, 1982, which act as a functional reafference of displacement (Pailhous and Mestre, 1989), is below motion perception thresholds. We supposed then that, since this movement is necessary to obtain information about the dynamics of displacement, the pilot has to perform a specific cognitive task based on a spatial representation of the situation. In this hypothesis, the pilot would have to infer the spatial properties of the ship's trajectory from the perception of static (or quasi-static) "visually poor" environment. Moreover, the pilots must keep their vehicle inside narrow channels in the presence of more or less unpredictable external disturbances (wind, currents) which modify the ship's dynamics.

Psychophysical studies have shown that, in tentatively similar conditions, subjects perceive elementary components of the overall visual motion, like radial expansion of a pattern of dots or their translatory motion in the frontal plane. But the integration of the elementary visual motions into a coherent global visual motion is incomplete (Mestre and Bonnet, 1989). From this analysis, interactive simulations of harbor navigation have been implemented on a micro-computer equipped with a high-resolution graphic terminal. With the aid of appropriate software using a dynamical model derived from recordings of real trajectories, subjects could monitor the trajectory of a simulated large ship by giving rudder orders. Every three seconds, the computer updated the variables defining the state of the vehicle (such as turn rate, heading angle, course angle, speed and spatial location). These values were used to draw a perspective view of the navigation area on the graphic display as the observer would see it at that time for his/her location in space (see figure 2).

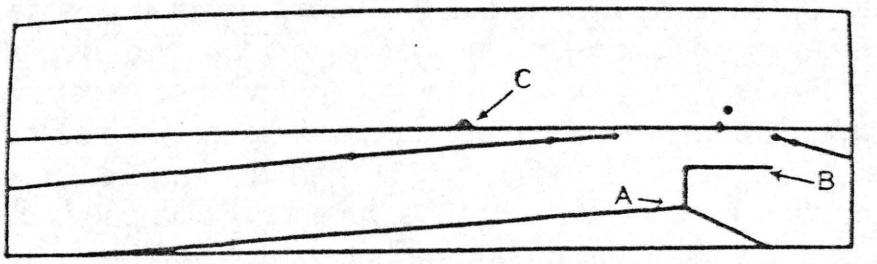


Figure 2. Perspective view of the navigation area as it appeared on the graphic terminal. It corresponds to the perspective view a pilot would get of the navigation area from the ship location calculated by the simulator. On that static picture, the channel's boundaries are represented by continuous lines joining the buoys delimiting the channel. At the bottom of the picture (A) the stem and mast of the ship are stylized. An oriented segment is drawn from the top of the mast (B). It is a turn rate indicator, whose length is proportional to the turn rate of the ship. The triangle on the horizon line (C) indicates the direction of displacement of the ship.

In the experimental work, we manipulated the visual information represented on the graphic "interface", as perspective views of the navigation area. The main results are that:

1) The control of displacement is favored by a simple structuration of the visual environment, notably by a representation of the channel borderlines (Cavallo, Mestre Peruch, Deutsch and Pailhous, 1987).
2) Representation on the perspective view of part of the vehicle (providing direct information about its spatial orientation) and dynamic visual cues indicating its rotational movements resulted in improved control performance (Mestre, 1988a; see also Schuffel, 1984).
3) Visual information about the spatial properties of the ship's trajectory also played a positive role in the overall control task (keeping the ship inside channel boundaries). In particular, graphic depiction of the ship's heading direction and an analog representation of motion in the visual scene played a positive role (Mestre, 1988a).

This work enabled us to show the fundamental role of information relative the system's dynamics for the control of displacement. It has contributed to our theoretical elaboration about the involvement of "low-level" cognitive processes in

the control of displacement (Pailhous and Mestre, 1989). It further suggested the fundamental role of visual motion perception for vehicle control, and provided guidelines for the definition of on-board visual aids for piloting large ships.

However, the processes involved in the control of displacement are far from clear. Indeed, many experimental studies have tried to compare perspective versus cartographic modes of presentation of spatial information in ship piloting (Schuffel, 1980; Bertsche and col., 1981; Perdock and Elzinga, 1984). Results from simulation experiments suggest a superiority of a perspective mode (see also Péruch, Cavallo, Deutsch and Pailhous, 1984 and figure 3). However, the experimental conditions are too different in the various works to lead to an clear conclusion. Schuffel (1984) poses the hypothesis of a necessary complementarity between the two modes of representation (the radar -map- view enabling localization and perspective views facilitating the control of displacement).

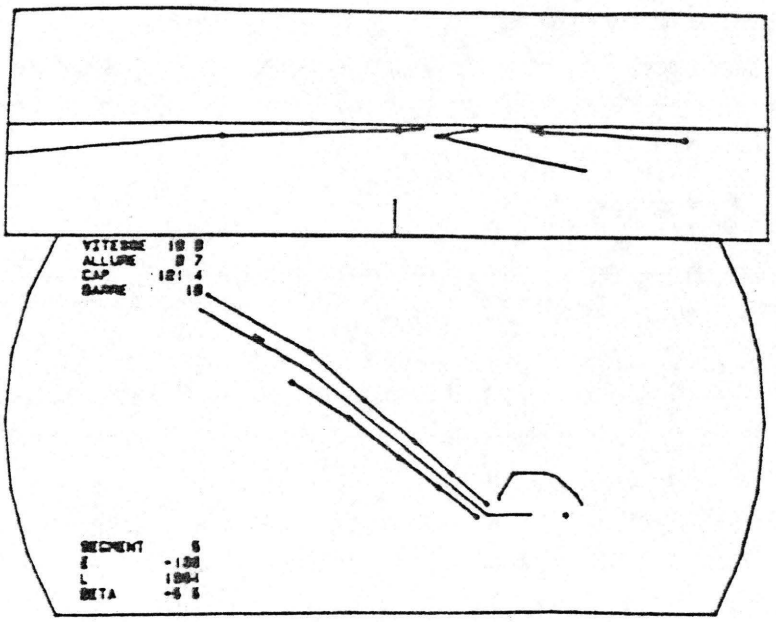


Figure 3. The computer screen with both modes of presentation of the visual information. In the lower part (cartographic mode) the limits of the access channel to Antifer (Le Havre) harbor are defined with lines joining buoys; the mid-line channel is also drawn. The vessel is represented by a small mobile segment. In the upper part of the picture, a perspective view of the channel is drawn, where the vertical line represents the stem mast of the ship. The alphanumerical information concerns the bridge parameters (speed, rate, heading, helm) and the steering parameters provided by the radio-electric aid system.

3. CONCLUSIONS AND PERSPECTIVES.

In short, designing realistic and functionally efficient human-machine interfaces in spatially-oriented control tasks remains a hard problem, and much needs to be done yet. We suggest that, in this class of tasks, a major issue is the format of visual information and that two important components of the control task are the localization of the mobile device, the control of its displacement and their interaction. Concerning the localization component, the analysis of the cognitive processes involved is in progress. We want in particular to understand, by experimental manipulation of the available information, the respective roles of cartographic (exocentric) and perspective (egocentric) information in a localization task. An important aspect is also the role of dynamic versus static visual information. Concerning the visual control of displacement, we emphasize the fundamental role of motion perception. We are already testing experimentally the ability of human observers to perceive properties of their displacement (such as the direction of locomotion) from the transformations of the visual scene (optical flow) resulting from the observer's displacements (Warren, Mestre, Blackwell and Morris, 1990). We are applying this analysis to car driving tasks (Mestre, 1988b; Berthelon, Mestre and Péruch, 1990), trying to show how fundamental it is to understand what the driver actually sees (and is able to perceive) to be able to furnish guidelines for the conception of the road environment and also to define efficient tests evaluating driving aptitudes.

In the ergonomic context, our aim is to contribute to the elaboration of useful guidelines for the conception of valuable human-machine interfaces, especially in the domain of teleoperation tasks. In order to go in this direction, our strategy is to build practical relationships with teleoperated devices designers, who have already developed or are developing prototypes. This will enable us to analyze the envisaged solutions to the above mentioned problems. Among the diversity of situations and solutions, this analysis should lead to fundamental questions which could be experimentally approached. This work supposes direct observations of working devices and the elaboration of a grid of analysis. We can mention some criteria which could be part of this grid, derived mainly from our previous studies. Concerning visual information, we will concentrate on its static or dynamic nature, which relates directly to the question of the nature of the interface (graphics and/or video images). The quality of the interface itself has also to be carefully analyzed (contrast, resolution), as well as its size (in particular its angular aperture). From what we said before, the cartographic and/or perspective format of the visual information will be questioned. The environment itself (hostile, known/unknown, convex/concave) should be of importance. We

will also pay much attention to the manipulation devices, including the control of the video camera. This approach to concrete work situations should enable us to understand better the perceptuo-motor coordination in the control of action and more concretely to provide precise recommendations for the design of human-machine interfaces. In this sense, we suggest that these interfaces have to be defined from the beginning with the general idea that they are a functional part of the human-environment interactions. They also have to represent the functional structure of the work situation in a way that matches the immediate task and the perceptuo-motor capabilities of the human operator.

REFERENCES

Allen, T.M., Lunefeld, H. and Alexander, G.J. (1971). Driver information needs. *Highway Research Record, 366,* 102-115.

Berthelon, C., Mestre, D. & Péruch, P. (1990). Perception of a moving vehicle when approaching an intersection. In A.G. Gale & al. (Eds.). *Vision in Vehicles III.* Amsterdam: North-Holland. In press.

Bertsche, W.R., Cooper, R.B., Feldman, D.A. & Schroeder, K.R. (1981). An evaluation of display formats for use with marine navigation piloting systems. Washington DC: United States Coast guards.

Brouard, J., Deutsch, C., Routin, M., & Cuny, X. (1979). Etapes préalables à l'analyse de la conduite : l'exemple du pilotage portuaire. *Bulletin de Psychologie, 344*, 263-271.

Cavallo, V., Mestre, D., Péruch, P., Deutsch, C., & Pailhous, J. (1987). Conception d'une aide informatisée au pilotage : un exemple de dialogue entre recherche et application. *Recherche, Transport, Sécurité (Revue de l'INRETS), 14*, 23-30.

Dupont, M. (1988). Etude ergonomique du poste de commande d'un robot mobile. Mémoire de recherche pour le diplôme d'Ergonomiste, CNAM.

Flach, J. (1990). Perceptual organization in ecological interfaces. In Proceedings of the First European Workshop on Ecological Psychology. Marseille, June 7-8 1990.

Levine, M. (1982). You-Are-Here maps: Psychological considerations. *Environment and Behavior, 14,* 221-237.

Levine, M., Marchon, I. & Hanley, G. (1984). The placement and misplacement of You-Are-Here maps. *Environment and Behavior, 16,* 139-157.

McRuer, D.T., Allen, R.W., Weir, D.H. and Klein, R.H. (1977). New results in driver steering control models. *Human Factors, 19,* 331-347.

Mestre, D. (1988a). Visual control of displacement at slow speeds. *Human Factors, 30,* 6, 663-676.

Mestre, D. (1988b). Vehicle control described as a tracking task: The role of visual information. Proceedings of the 7th European Conference on Human Decision Making and Manual Control. Paris, 18-20 Octobre 1988, 139-144.

Mestre, D., Bonnet, C. (1989). Le contrôle visuel du déplacement à vitesse lente. *Psychologie Française, 34, 41-46.*

Mestre, D., Cavallo, V., & Péruch, P. (1986). Definition of an on-board visual aid for piloting large ships. In A.G. Gale & al. (Eds.), *Vision in Vehicles.* Amsterdam: North-Holland Elsevier Sciences Publishers BV.

Pailhous, J. & Cavallo, V. (1982). Les effets spatiaux du déplacement: Leur rôle et leur traitement. *L'Année Psychologique, 82, 457-472.*

Pailhous, J. & Mestre, D. (1989). Interpretation spatiale des reafferences visuelles du mouvement. In J.P. Caverni, C. Bastien, P. Mendelsohn & G. Tiberghien (Eds.), *Psychologie Cognitive : Modèles et Méthodes.* Grenoble : Presses Universitaires de Grenoble.

Perdock, J., & Elzinga, T. (1984). The application of micro-simulators in port design and ship handling courses. Proceedings of the Third Conference on Marine Simulation (MARSIM 84), Rotterdam.

Péruch, P., Cavallo, V., Deutsch, C.& Pailhous, J. (1984). Real-time graphic simulation of visual effects of locomotion. In T.R.G. Green, M.J. Tauber & G.C. Van der Veer (Eds.), *Cognitive Ergonomics: Mind and Computer.* Heidelberg: Springer Verlag.

Péruch, P., Pailhous, J. & Deutsch, C. (1986). How do we locate ourselves on a map. *Acta Psychologica, 61,* 71-88.

Péruch, P. & Savoyant, A. (1990). Spatial knowledge involved in a location task: Possible conflicting frames of reference. In Proceedings of the First European Workshop on Ecological Psychology. Marseille, June 7-8 1990.

Schuffel, H. (1980). Some effects of radar and outside view on ship's controllability. In D.J. Osborne & J.A. Lewis (Eds.), *Human factors in transport research*. London: Academic Press.

Schuffel, H. (1984). The interaction of rate of movement presentation with ship's controllability. In *Proceedings of the Third Conference on MArine Simulation, MARSIM 84,* Rotterdam.

Uttal, W. (1990). Les téléopérateurs. *Pour la Science, 148,* 90-95.

Warren , W.H., Mestre, D., Blackwell, A.W. & Morris, M. (1990). Perception of circular heading from optical flow. *Journal of Experimental Psychology: Human Perception and Performance.* In press.

Collaborative Interface: A PDP Approach

Antonio Rizzo[1], Sebastiano Bagnara[2]

1) Istituto di Psicologia del CNR, Viale Marx 15. 00137 Roma, Italy. email: antonio@irmkant.bitnet
2) Universita' di Siena, Via Roma 57. 53100 Siena, Italy. email: bagnara@irmkant.bitnet

1. INTRODUCTION

Interfaces play a critical role in Human Computer Interaction (HCI). They encode and translate the information to be exchanged between the computer and the user, and vice versa. In the last years, human computer communication has qualitatively improved, mainly because of the introduction of the Direct Manipulation Interfaces (DMI). These interfaces often eliminates the inter-referential distance between input and output since the user can act upon the very same objects whose modification s/he has to evaluate. At the same time, at contrast with the traditional interfaces where the symbolic communication was privileged, a DMI allows the user to act on the world s/he is willing to modify by providing a consistent metaphor in respect to the domain of action. In doing this the DMIs remove the articulatory distance between the form and the meaning of an expression in H-C communication (Hutchins, Hollan and Norman, 1985).
The DMIs positively influence many cognitive aspects of human-computer interaction, even though they mainly concern the physical tools available for the interaction. They do not take much into consideration how the user accomplishes his/her goals and intentions, and how and which action's outcomes the user pays

attention to in evaluating the results. In other words, the actual DMIs, as well as many other tools, do not provide specific help in filling the semantic distance between human and computer, both in the execution and in the evaluation gulf (see, Fig.1).

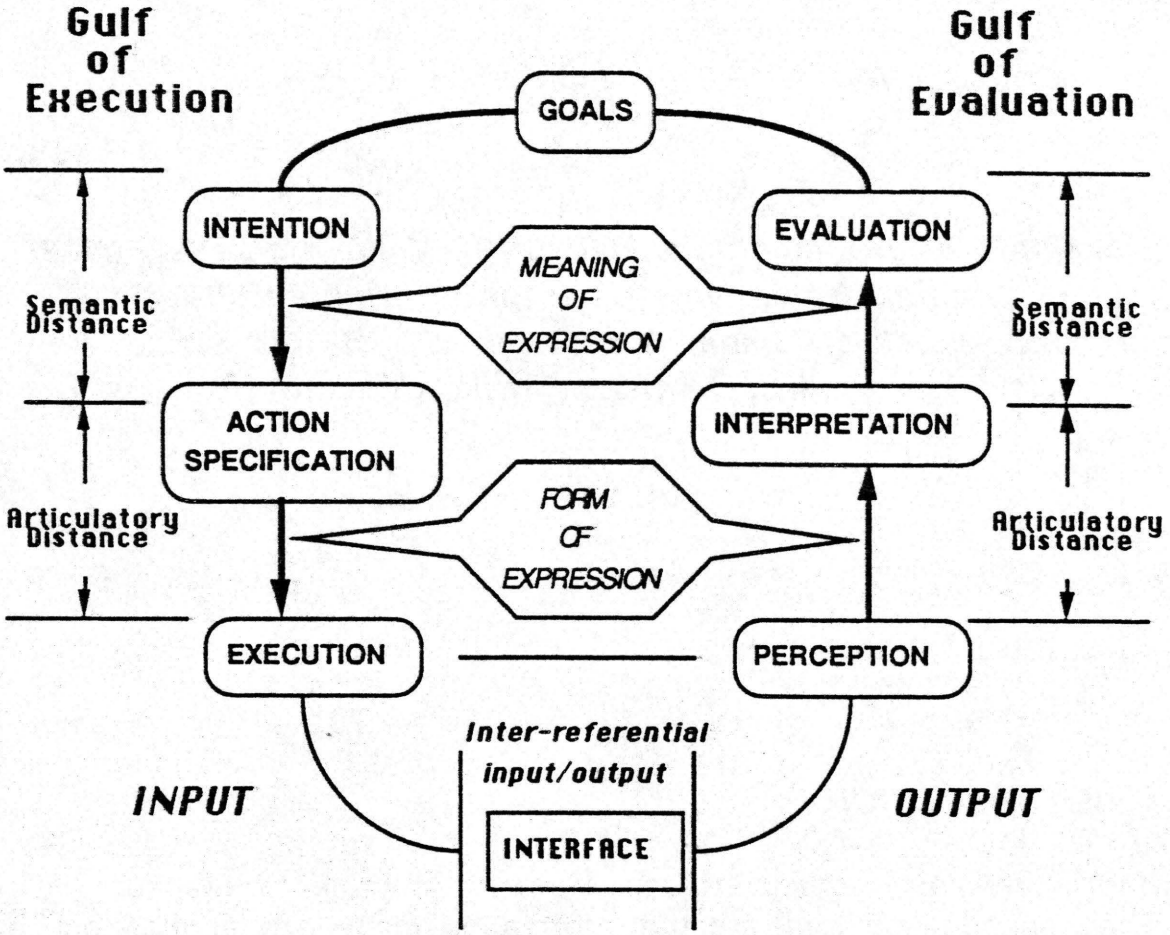


Fig.1: Human–computer communication model (Hutchins, Hollan and Norman, 1985)

The semantic distance issue has been dealt with by Hutchins (1989). He put forward the idea of collaborative manipulation interfaces, where the user faces a DMI plus an Intermediary Interface. In this situation the user can dialogue with the Intermediary about the World of Action, and both the user and the Intermediary can manipulate the World of Action. The Intermediary interfaces was thought as a Parallel Distributed Processing (PDP) system (see Fig.2).

The present paper reports a study in progress based on this idea, which represents a first tentative exploration of the potential of this approach. A PDP system is presented which learns to perform actions on the DMI (the World of Action). Actions are related to

specific goals (States of the World), and communication between the PDP system and the user concerns these States of the World.

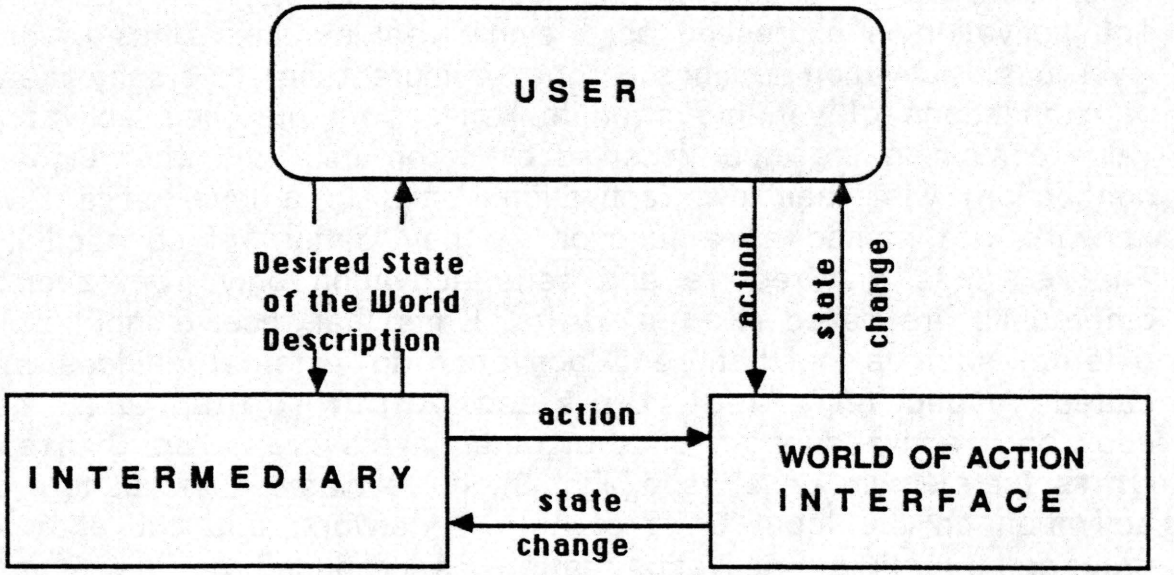


Fig.2: Collaborative Interface (After Hutchins, 1989)

The paper is organized as follow. First, the approach adopted to cope with the computational and psychological aspect of action is briefly reported. Then, a simplified world of action (HyperCard Tutor) and a simulation study are described. Finally, some conclusions on the current potentials and limits of this approach to collaborative interface are discussed.

2. A MODEL OF ACTION

The development of a collaborative manipulation interface requires a system able to plan and perform action upon the interface (the World of Action). The model of action and action planning adopted is the Environment-Action-Environment (EAE) model put forward by Miyata (1988). This model derives from the Activation-Triggering-Schema (ATS) framework for action control developed by Norman and Shallice (1986). The Miyata's model takes into account two formerly under-specified aspects of action: "the process in which an action plan is determined in order to achieve an outcome, or goal, and the learning schema from experience" (Miyata, 1988, pp. 2). This two aspects were conceptually present, though implicit, in the ATS model but not integrated as functional parts of the model. Miyata implemented the model as a PDP network.

A PDP network consists of a set of processing units organized in layers. At each moment in time every unit has an activation value which is the only dimension that describes its state at that time. The activation is expressed as a single real number. Units of one layer pass activation to units of one or more other layers by mean of connections of variable strength. Units compute their activation value combining the input received by other units (via the weighted connection) with their own activation. Units in a layer of a PDP network, are named as a function of their pattern of connectivity. That is, units that receive and send activation only from and to other units are called **Hidden Units**. Units that receive input from external sources or that send activation to external devices are called respectively **Input Units** and **Output Units**. Units that copy back activation from units of other layers are called **Context Units** (or memory units) (see, Fig. 3). If we clamp some pattern of activation on the Input Units of a PDP network, this activation is propagated until an activation pattern is produced on the Output Units. This is the output pattern of the network, and we can think of the network activity as a mapping between an input pattern and an output pattern.
So far, it seems that there are no limits (theoretical limits, at least) to the kind of patterns a network can learn to map. Learning occurs in the network as change in the weights of connections, since changing the pattern of weighted connections modifies the mapping between input and output patterns. Thus, what the network knows is encoded in the connections rather than in the units' activation. The learning algorithm used in this work is the "backpropagation of error" developed by Rumelhart, Hinton and Williams (1986). An activation pattern is clamped on the input units of the network, activation is propagated so to produce an output activation pattern. The output pattern is matched with a target pattern, which represents the desired output of the network for that given input. The connection weights of the network are modified according to the maximum gradient descent on the sum of squared error between the output and the target patterns.

The PDP networks have allowed the implementation of many cognitive models with a good level of psychological plausibility. The psychological plausibility of the model of action and action planning has a twofold importance for the present purposes.
First, it allows a match between the two modes of action planning (user/PDP network), which can result in a better shared background on action knowledge . This is a crucial aspect in communication, which has been often underscored.

Second, the EAE model is consistent with the Hutchins, Hollan and Norman's model of cognitive distance.
Finally, since human action planning is the best known way to map from a goal to the action needed to accomplish that goal in noisy conditions, psychologically inspired models can benefit from this knowledge.
The EAE model of action implemented as a PDP network has an high level of psychological plausibility, coming as is does from a psychological model which accounts for many action phenomena.

2.1. The EAE Model of Action Planning

The model copes with the problem of action planning and execution as a problem of mapping from a Desired State of the World to a representation of the Actions needed to achieve that state. This mapping depends on: 1) The internal and external context in which action are executed, and 2) The evaluation of results. The mapping is learned by experience as in many PDP network.
The Network adopt as learning algorithm the backpropagation of error but in a new learning paradigm. This paradigm involves two main assumptions (Miyata,1988): a) The environment as the teacher; and b) The goal as partial specification of environment. The first assumption maintains that the feedback provided to the network simply reflects the effects of the network's action on the environment and does not specify what action is appropriate. The second assumption is more complex, it involves four aspects: 1) The goals are multidimensional (e.g., in navigating through HyperCard the goal concerns the trajectory of movement, rather than the final state or any other single state). 2) The goal can change from situation to situation, so the trajectory of control can vary from time to time. 3) The goal does not specify all the dimensions. Only part of the trajectory may be specified even though the goal concerns the full trajectory. 4) Which subset of dimension are specified can vary from situation to situation.
These assumptions are reflected both in the way in which the network learns and in the architecture of the network.
Figure 3 reports the basic architecture of a Miyata network. It is formed by two sub-networks, the EA network and the AE network. The EA network maps from a Desired State of the World to the Actions needed to accomplish such a State. The AE network maps from the Action to a prediction of the State of the World as a function of the way it has been modified by the action. The two networks are trained separately. First, the AE network is trained so to make reliable the prediction on the modification caused in the World as a function of the actions. The error between predicted

and real change is used for this first training. Then, the EA network is trained to perform the proper actions to accomplish a given Desired State of the World. The error between the prediction (E_feedback) and the Desired State of the World (E_desired) is used for this second training on the connections of the EA network. On both networks, the time-context and the spatial-context influence the mapping by means of two sets of memory units, the A_context and the E_context. Note that in both trainings there is no teaching input that specify which actions are correct from situation to situation.

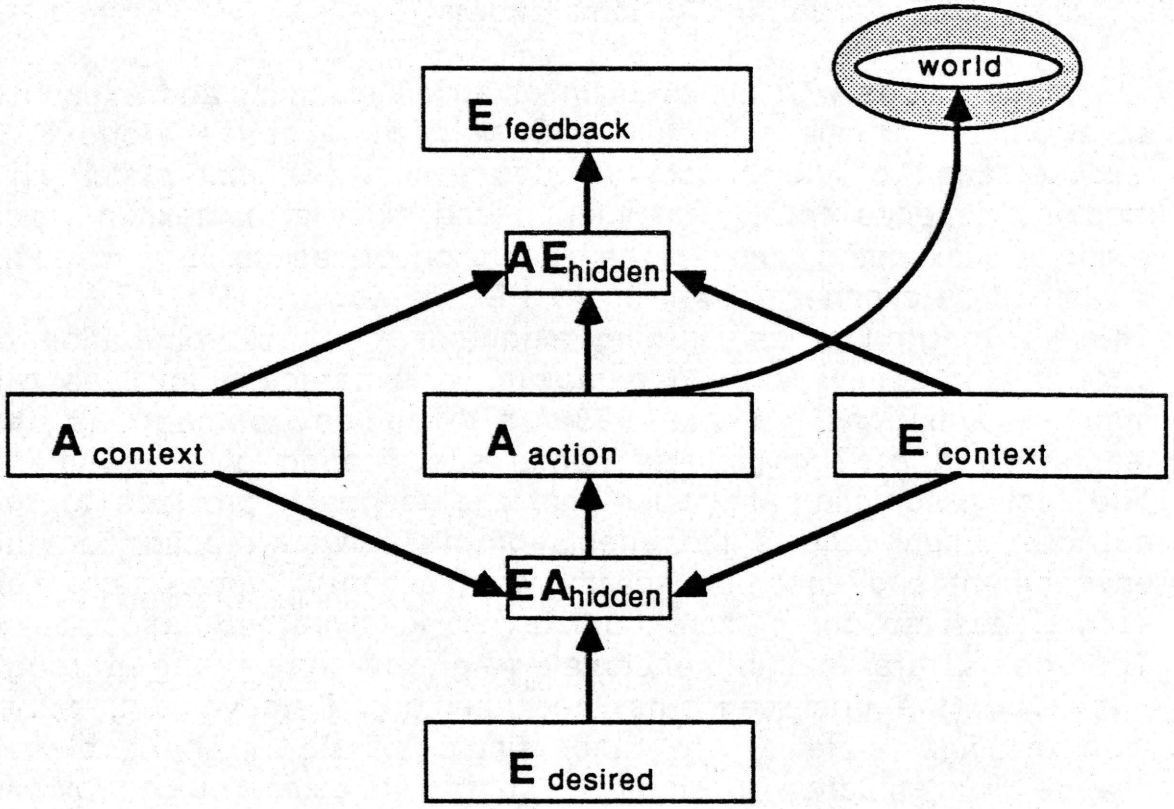


Fig. 3: Miyata basic architecture network.

Miyata has applied his paradigm to a navigation ship task (Miyata, 1988). The ship, which moves in a two dimension space, has the goal of reaching an end position passing through five gates. The Desired State of the World is defined as the trajectory (positions in space) to reach the end position. The Action is defined as the angular direction on movement. In his simulation, Miyata showed how the EAE network can learn the mapping from the Desired State of World to the Actions needed to accomplish such a state without having any direct teaching input on the correct action to perform for each step toward the final state (the environment as a teacher assumption). Moreover, he showed that this network can find a

solution even with an under-specified Desired State of World. That is, by providing the network with only some of the dimensions of the trajectory needed to reach a given state, the network can find a solution by iteratively going through the desired state of the world and the feedback from the world (the goal as a partial specification of environment assumption[1]).

The psychologically plausible EAE model is one of the first important step toward a more computationally powerful and psychologically based model of action. However, it lefts out at least three important psychological requirements for action model, some of which (b and c) are present in the ATS model: a) An action model should account for error in the definition of the State of World (mistake). b) It should describe the role of conscious attentional control and automatization (but see the HS model of Miyata (1987) on automatization). c) It should account for multiple levels of representation of action. These are the directions along with extending the present model.

3) HYPERCARD TUTOR AS WORLD OF ACTION

The interface considered for the pilot study on collaborative interfaces is that provided by an HyperCard Tutor. In HyperCard there are three basic operations: opening stacks, moving through stacks, and using stacks for their intended purpose. A stack is an ordered collection of cards. Cards can contain text, pictures and buttons. Buttons are used to navigate through HyperCard. Whenever HyperCard is running, there is a single "current stack" and a single "current card". HyperCard is a non sequential mode of information collection and manipulation. The links between Cards are usually unidirectional and information about links (buttons) is provided only on the Card from which you can move out.
The tutor being considered is a simplified version of a prototypic tutor which has the teaching strategy organized as paths through the cards and the stacks. This is a very common configuration for HyperText Tutors (see., Hambron and Hooper,1990). The users start with an "Home Stack" and from there begin to navigate toward the end of their lesson. The teaching path represents a privileged path, even though many other routes are available to the user. In the tutor there are 35 cards organized in 3 stacks, and, overall, there are 9 teaching paths (see, Fig. 4).

---

[1]This last situation needs four more assumption on the working condition of the network that will be no considered here.

It takes into consideration only the actions performed to move through Cards, except those which can be performed on the menu bar (e.g., File, Edit, Go). Pop-up windows and other operations on the Card are also not considered. These modifications has been adopted in order to simplify the world of action without losing the Direct Manipulation metaphor in the interface. In this pilot study, it was deemed important to test some potentials of this approach in relation to a generic task, not to produce a real, practical Intermediary tool.

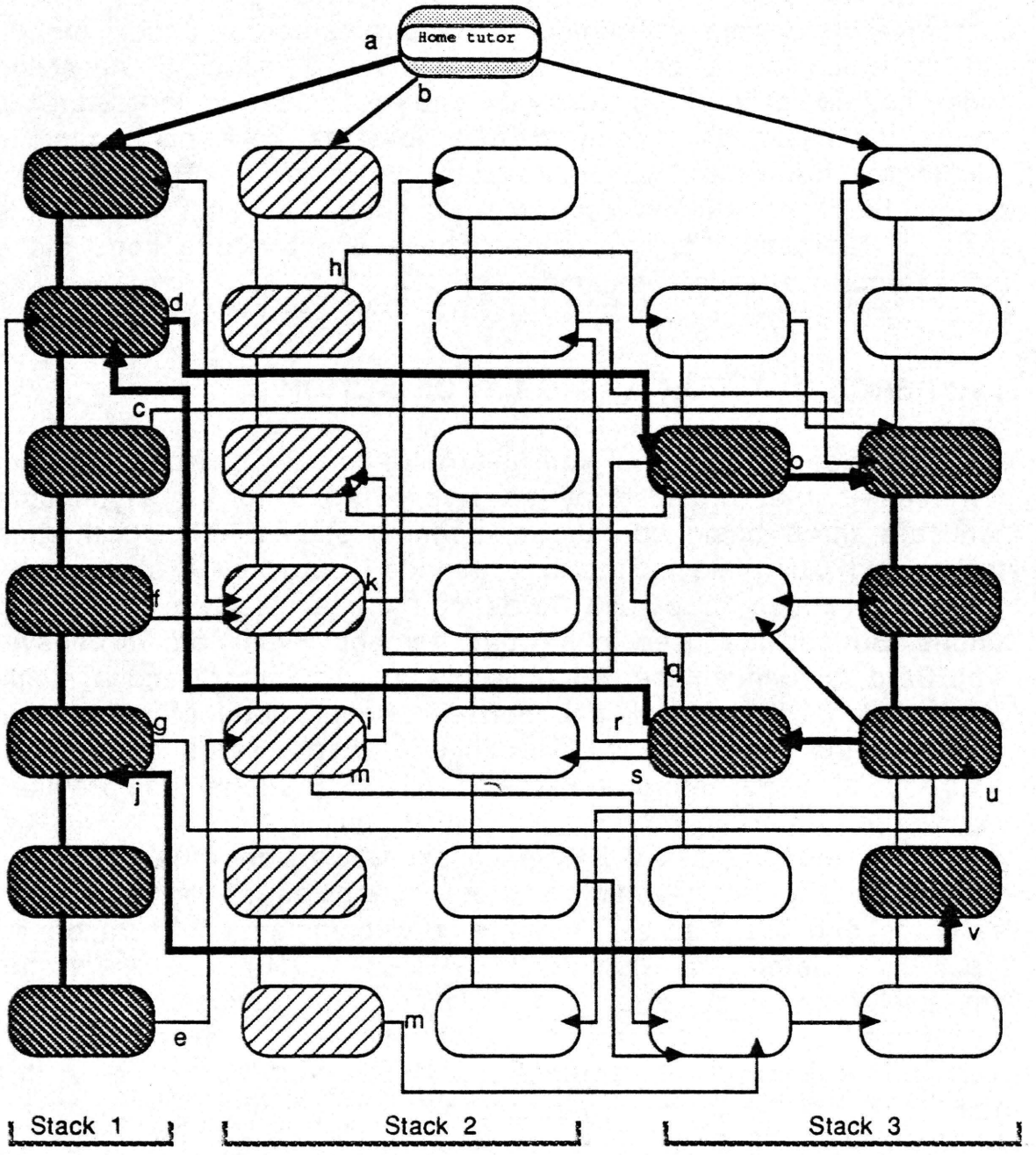


Fig. 4: A schematic view of the HyperCard Tutor.

4. SIMULATION STUDY

The task

As mentioned before, the task the Network should learn is to perform actions on a Direct Manipulation Interface. These actions are finalized to navigate through HyperCard as a function of given route (lesson). Lesson are compound by 9-15 Cards and the task requires as goal the description of the full trajectory.

The network

The network has the basic configuration of the Miyata Network (see Fig. 3). The most important difference concerns the role played and the information represented in the E_feedback and E_context layers of units. In the Miyata's study, these two layers represented the same kind of information, and the E_context units just reported the activation values of the E_feedback units at time t-1. In present study, this condition is considered too restrictive, since not all the actions are possible at each time step. Additional informations is available from the environment to cope with this issue without overloading the AE mapping.

The Desired State of the World (E_desired layer) represents a trajectory over a period of five steps. That is, five different and sequential Cards are identified. The length of the trajectory can be as long as desired, and length does not matter in a coarse code representation. Here, the length of the trajectory has been limited for computational reason, even though a localist representation of the trajectory was adopted. Each Card is represented by means of a 6 units binary vector (there are overall 45 Cards). Thus, each trajectory or Desired State of the World is represented by a 30 units binary vector.

The Action layer represents the sequence of actions performed to accomplish the trajectory. In the simulated HyperCard interface there are 24 possible actions. Actions are specified as "click" on a given button. Two buttons refer to, respectively, the forward and backward arrows that represent the privileged links in HyperCard, one refers to the end Card (the carriage return), the other 21 buttons refer to the 21 icons representing connections to non-serially ordinate Cards in or among the Stacks. Each action is represented by means of a 5 units binary vector. So the Action layer is compound by a 25 units vector.

The E_feedback layer represents the network's prediction of the trajectory produced by the executed action sequence. The E_feedback layer is compound by a 30 units vector.

The E_Context layer represents the current available information on the State of the World. This is given by the former E_feedback

units plus information on which buttons are available from position to position. This last information is represented by additional 15 units for each time step. On these units the one up to three additional buttons that can be present in a Card beyond the two arrows for forward and backward movement are reported. Since not all the action are possible in every position (Card), some actions, even if performed, will not produce any change in the environment. This is represented as a 0 value on an additional binary unit. It can be considered as redundant information available in Forcing Function situations. The E_feedback layer is compound by a 110 units vector.
The A_context layer represents the former action sequence by coping back the 25 units vector of the Action layer at time t-1[2]. Finally, there are 40 units in each of the two hidden layers.

The Training

The training took place in two phases: The first phase involved the AE network, the second the EA network. In the first phase, the network learned to map from its actions to their consequences on the world. Random sequences of action in different serial and spatial context were given as input to the network. The prediction of the results, produced as output on the E_feedback units, was matched with the "real" results, and the resulting error was backpropagated through the connection of the AE network. The training loop lasted until the network was able to predict the results of its actions with an 0.05 average squared error per output unit.
In the second phase, the network learned to map between a given Desired State of the World and the actions to accomplish it. Several trajectories were provided to the network on the E_desired units with different spatial and temporal contexts. Actions were produced in the output units (Action layer) and from there propagated to the AE network. The error between E_feedback and E_desired was backpropagated through the connections of the EA network.
Note that the error for both network was computed only for the values that identify the trajectory (e.g., the 5 serial Cards to be passed through), even though much more information from the environment was available.

Results

The network learned easily to map from a given trajectory to the actions to accomplish it. The privileged routes (lessons) made up

---

[2]The role of A_context units are relevant in this study for the possibility to discriminate between partially overlapping trajectory belonging to different teaching route.

by two or three trajectories were executed with an average squared error of 0.18. One aspect of the simulation deserves special attention: The role played by the E_context units. These units represent the sensory input of the network. At each time step they bring some information that is crucial for the efficiency of the two mappings (E_desired to Action, Action to E_feedback). In fact, when some faulty values (e.g., no buttons available on a Card that, when present, allow some actions) are provided on these units, the performance decreases notably. In learning the mapping, the network rests upon such "external" information which is fully available to it.
Moreover, it is important to note that the trajectory can be specified at different levels (e.g., a more features-based description or a fully arbitrary one, as in this simulation). the needed information will be available in the environment. The Desired State of the World can be defined at different levels providing that there is a good mapping among, and between them, and the information be available in the environment. This is consistent with the psychological requirement that asks for more level of description of the action.
Given this result and the demonstration by Miyata that it is not necessary to define the whole trajectory in order to have it accomplished (such a result was not replicated in the present study, since a localist representation of the trajectory was adopted in order to have more control on the role played by each time step), the next step is to reach a multiple representation of the goals related to the same or different information available in the environment. The relationship between multiple representation, direct feedback (prediction) and sensory information is under study.

5. CONCLUSIONS

The study was aimed at presenting some of the potential of a PDP network as Intermediary tool in Collaborative Manipulation Interface. The benefit of this approach consists mainly in the psychological plausibility of the action planning performed by the PDP network.
Many other systems which can cope well enough with action planning already exist. Nevertheless, it can be maintained that the proposed approach is very fruitful to cope with the problem of filling in the semantic distance in Human-Computer interaction. The PDP network maps from the Desired State of the World to Action, and evaluates the results, that is operates where there is a cognitive distance to be reduced. Assume, for example, that an user

faces a situation where he barely knows which end state he is is willing to reach. Even if he cannot properly define it and does not know how get there, he can try to define this Desired State of the World and communicate it as an input to the Network. The way in which the Network looks for solutions can both lead to the Desired State or can make available some informations that can help the user to better define his Desired State of World. This, interactively and iteratively, can allow to reach the goal.
It it worthwhile to note that many levels of definition of the Desired State of the World can be allowed. This can be helpful both for the Network, by providing it with further constraints for the mapping, and the user, by easing the communication with the Intermediary and allowing him to evaluate the output of the system at the proper level. In the Network, the level at which the desired state of the world is defined is always matched with feedback information available at that level: there are no cases in which cognitive distance can hamper the match.
To the best of our knowledge, there are no other systems that can afford this. Problems can arise if there is no match between how a desired state of the world is defined by the user and the levels of definition known to the Network. However, these drawbacks can be overcome by means of interactive-iterative definitions of the State of World once the multilevel definition has been implemented. Of course, the user has to adapt to the PDP level-structure of activity description. This occurs whenever users learn to use new tools. A PDP Intermediary Interface increases the chances of a correct "learning by doing".
Other benefits can come from the ability of the PDP network in recognizing the plans from the actions executed on the interface by the user. For example, through plan recognition the PDP network can be of much help in preselecting an area of a database and speeding up the recovery time of documentations when on-line documentation and manuals have to be handled.

At present the most important limits that we see in relation to the future development of an Intermediary tool implemented as a PDP network are related to the scaling problem and to the number of degrees of freedom by which a desired state of world can be accomplished. Both problems can be of difficult solution. However, it should be noted that we have to consider a toy not only the HyperCard tutor but also the psychological model of action. Thus in figuring out more complex sistuations we should also figure out more complete models of action.

References

Hambron, S., and Hooper, H., (1990). **Learning with interactive multimedia.** Redmond, Washington: Microsoft Press

Hutchins, E.L., (1989). Metaphors for interface design. In M.M. Taylor, F. Neel, and D.G. Bouwhuis (Eds.) **The structure of multimodal dialogue**. Amsterdam: Elsevier

Hutchins, E.L., Hollan, J.D., and Norman, D.A., (1985). Direct manipulation interfaces. **Human-Computer Interaction**, 1, 311-338

Miyata, Y., (1987). **Organization of action sequences in a hierarchical sequential network**. ICS Report 8707. La Jolla, CA: UCSD.

Miyata, Y., (1988). **The learning and planning of actions.** ICS Report 8802.La Jolla, CA: UCSD.

Norman, D.A., and Shallice, T., (1986). Attention to action: Willed and automatic control of behavior. In R.J. Davidson, G.E. Schwartz, & D. Shapiro (Eds.) **Consciousness and self regulation: Advances in research, Vol. IV**. New York: Plenum Press.

Rumehlart, D.E., Hinton, G.E., and Williams, R.J. (1986). Learning internal representation by error propagation. In D.E. Rumelhart, J.L. McClelland (Eds.), **Parallel distributed processing: Explorations in the microstructure of cognition. Vol.1: Foundations**. Cambridge, MA: MIT Press.

Analysis of Learning, Thinking and Social Processes in HCI

Menu Trees as Memory Aids

Pamela Briggs and Anne Lambourne

ABSTRACT

Subjects were trained in word-processing using either a conceptual approach, in which menu trees were used to illustrate the command structure; or a step-by-step procedural approach. Task performance and task-relevant knowledge were tested immediately after training, and several days later. Users given a conceptual overview acquired a greater explicit knowledge of commands, and retained this more successfully.

INTRODUCTION

Many of the difficulties experienced by novice word-processor users can be attributed to their failure to make adequate use of the self-study materials provided. The most common self-study materials are manuals and on-line training packages, however neither of these support the natural learning styles of users (Carroll, 1984): novices are reluctant to wade through cumbersome training manuals; but they also dislike following step by step instructions.

In a comparison of training techniques, Czaja et al. (1986) trained subjects in word processing, using either a tutor, manual or computer-based package. Subjects who received the on-line training package attempted fewer tasks, took longer to perform tasks, and made more errors, especially in using the command structure. Czaja and colleagues suggested that this was because on-line training packages promote passive learning while manual and tutor based instruction encourage users to be active.

In general, users prefer to engage in active exploration of the system (Carroll and Mazur, 1986), and so self-study materials which support active learning are to be preferred. Manuals have the potential to do this, but users dislike reading manuals, and will often skip vital information as a consequence (Carroll, 1984). Thus users are often left to explore the system in a rather ad-hoc, unsupported fashion. Carroll and his colleagues have suggested two means of tackling this problem: Firstly by presenting novice users with a drastically reduced manual; and secondly by actually disabling those more advanced commands which could lead to difficulty. The advantage of the latter was demonstrated by Carroll and Carrithers (1984) who devised a 'training wheels' system in which learners were given the chance to actively explore and master preselected functions. Training wheels subjects were faster and more successful overall than subjects using the full system.

The former solution is more straightforward, but seems valid given the finding that readers presented with less will often remember more: replacing an cumbersome

instructional text with simple summaries of the main points has been shown to produce superior recall for a period of up to one year later (Anderson and Reder 1980; 1982). With this philosophy in mind, Carroll et al. (1987; 1988) developed the 'Minimal Manual', in which repetition, summaries, reviews and practice sessions were all eliminated. Learners using the Minimal Manual were quicker to start tasks, made fewer errors, and made better use of the manual as a reference document compared to those using the full manual.

This minimalist approach seems promising, but it does beg the question of just what information should be included. Given the need to reduce information, is it important to give the user some conceptual overview of the system, or is it enough simply to provide them with a simple reference set of procedures? A number of studies have addressed this issue.

Procedural versus conceptual instruction

Conceptual instructions that highlight the structure of a system allow the user to generate an appropriate mental model of that system. In practice this means that subjects given conceptual training should be able to infer procedures, when unable to recall them directly. The anticipated advantage for conceptual over procedural training has been noted in most, but not all studies which have addressed this issue:

Keiras and Bovair (1984) compared subjects given a conceptual model of a control panel device to subjects who simply learned operating instructions by rote. Those given a device model took less time to learn the procedures, and after a delay of one week still retained the procedures better and executed them faster than the procedurally trained group. In a second study, the same group manipulation was used, but subjects had to infer the procedures rather than learn them. The results supported the theory that the device model group were able to directly infer answers to problems from the memory of the model, while the procedural group followed a trial and error strategy that required greater effort.

Schmalhofer (1987) demonstrated the effectiveness of instructing users as to the conceptual structure of a computer system. Conceptual training consistently resulted in faster times then procedural training. Mental models appeared to be of central importance. Encoding of the structure of the system led to a production of a mental model in which procedural knowledge was not explicitly represented, but from which procedures could be inferred. Memorization and execution of procedural instructions seemed to produce higher memory demands and more errors than inference and discovery from a mental model shaped by conceptually based instruction.

A study by Sasse, Johnson and Briggs (1988) provided rather more limited support for conceptual training. They used a tutorial approach to word processing training in which subjects who received conceptually based instructions were aided by a guide to the hierarchical command structure, while the those instructed in procedures simply received summaries of the tasks learned. The effect of the two training styles on training time, time to accomplish the tasks and number of errors made were studied. While there were significant savings in training time for the conceptual group; no significant differences were found for performance, time on task and errors.

Finally, and in complete contrast, Reder, Charney and Morgan (1986) investigated the effect of examples as elaborations of instructional text for a personal computer. Subjects either received procedural or conceptual elaborations. Novices receiving procedurally elaborated text displayed superior performance while conceptual elaborations did not aid performance at all. It was suggested that the result was obtained because the mapping between the procedural examples and the task were closer than the conceptual examples and the task. As Norman (1983) states, conceptual models rely on their functionality. If they do not correspond to the system image, or do not predict or explain the computer system, then they have no function.

The superiority of conceptual instruction has not been conclusively established, although the formation of an appropriate mental model would seem advantageous in learning to use a complex device. However, most studies have concentrated on differences between procedural and conceptual learning displayed on the same day as training. Where subjects were retested some time after the initial training session, differences appear to be clearer - as with the Kieras and Bovair (1984) study. A similar investigation using word-processing has not been made, although clearly, users trained in word-processing often find that they experience a delay between training and their eventual, sometimes infrequent use of the system.

The present study investigated the differences between conceptual and procedural learning displayed immediately and some days after initial training in the use of a word-processor. A tutorial approach was used as totally novice learners are often fearful of damaging the system (Carroll and Mazur, 1986). Tutoring allows the trainer to reassure the novice and explain the word-processor functions and tasks required in non-technical terms appropriate for the individual learner.

Conceptual training focussed upon an explanation of the system's hierarchical command structure and was supported by a guide sheet which illustrated this command structure in the form of a menu tree. Procedural training concentrated on step-by-step instructions of how to perform the tasks required - in this case the user's guide sheet involved a simple look-up table of which commands executed which functions. The conceptual training group were expected to exhibit greater knowledge both immediately after training and after an interval of several days.

One difference between this and the previous studies reported, was that an attempt was made to assess the resultant knowledge of the user in two ways: Firstly their procedural knowledge of the task at hand was assessed by measuring time and errors during task completion; but also their explicit, declarative knowledge of the command structure was assessed using free recall, and question and answer sessions. That these can be differentiated has been shown by Briggs (1990) who found that relatively skilled word processor users were very poor at constructing questions about a word-processing package, indicating a rather limited ability to explicitly access the information stored within well-learned procedures. It was felt that the advantage for conceptual training may be shown in users' explicit knowledge of the system.

METHOD

Subjects

Fourteen undergraduates from the City of London Polytechnic volunteered to be subjects in the study. None of the subjects had word-processing experience. Half the subjects were randomly allocated to the procedural training condition, and half to the conceptual training condition.

Materials and Apparatus

The word-processing package employed was WORD Version III, running on a Nimbus P.C. User input was via keyboard and mouse.

Training Material:
A guide sheet for each training condition was prepared. Each contained the same information but presented in a slightly different way. For the procedural training condition the guide sheet included step-by-step instructions of the tasks required. For the conceptual training condition, the guide sheet illustrated the hierarchical nature of the command structure.

Task Material:
The word-processing tasks involved creating and saving a file; and retrieving and editing a file already saved on disc. Three similar files were made and randomly allocated to either training, testing after training, or testing after an interval of some days. For the editing task, corrections were marked in red ink on printed copies of the documents. In addition a list of seven true and seven false statements relating to the material taught was prepared.

Procedure

The subjects were trained and tested on an individual basis. The first testing session took place immediately after training. The second testing session took place between four and seven days later.

Procedural training began with the subject being given the procedural guide sheet to study. The trainer explained the function of the keys needed in the task, and the use of the mouse. With the subject looking on, the trainer performed the training task, explaining the tasks in a step by step sequential manner, referring the subject to the guide sheet. The subject then performed the training task with the aid of the guide sheet and trainer's help whenever necessary. Before the end of the session the trainer ensured that the subject understood the guide sheet.

Training for the conceptual condition began with subjects being given a conceptual guide sheet. The functions of the keys needed in the task and the use of the mouse were explained. The subject then performed the training task. The trainer coached the subject through the task emphasizing the hierarchy of commands as illustrated on the guide sheet. The trainer ensured that the subject understood the guide sheet before the end of the session. The time of the whole training session, inclusive of the subject's questions was recorded.

Test sessions were identical. Each consisted of a word processing task, preceded and followed by free recall and the true/false test. The first test session followed training and the second test session was undertaken 4-7 days later. In the free recall task, subjects were simply asked to write down whatever commands they could recall, indicating their function; while in the recognition memory test, subjects were given a list of statements concerning commands and were asked to indicate whether they were true or false, by ticking the appropriate boxes.

For the word processing task, subjects were given the instructions and the set of amendments required. When the subject was ready the task commenced. The number of referrals to the guide sheet were recorded and the nature of the errors made were noted. The subject received no help except if an error was made from which the subject could not recover. The time taken to complete the task was recorded.

RESULTS

Analysis was made of training times, time on word-processing task, number of referrals to guide sheets, number of items recalled, and number of correct responses to the true/false test.

Training Times

Training times for both conditions were similar and not significantly different (Table 1).

Table 1: Mean training times (mins)

Procedural	37.57
	(9.36)
Conceptual	33.14
	(8.43)

Time on word processing task

Although means indicated an advantage for the conceptually trained group, a 2x2 analysis of variance did not support these differences (Table 2).

Table 2: Mean time on Word-Processing task (mins).

	Session 1	Session 2
Procedural	20.00	16.00
	(11.60)	(10.33)
Conceptual	12.86	12.57
	(2.04)	(4.61)

Referral to guide sheet

Once again the mean scores reveal an advantage for the conceptual group, which is statistically unsupported.

Table 3: Mean Number of referrals to guide sheet

	Session 1	Session 2
Procedural	2.71	4.43
	(3.45)	(6.80)
Conceptual	0.86	1.77
	(1.22)	(1.38)

Free Recall

Recall was scored by calculating the number of items recalled from a pre-specified list of 25 items. Mean scores for each group were calculated for each of the four tests given (Table 4).

Table 4: Mean Number of points recalled

	Session 1		Session 2	
	Before W.P. Task	After W.P. Task	Before W.P. Task	After W.P. Task
Procedural	7.43	10.00	3.86	6.29
	(3.95)	(4.20)	(3.44)	(4.23)
Conceptual	11.54	13.29	7.71	13.00
	(4.54)	(4.79)	(2.81)	(3.56)

The data were analysed using a 2x2x2 (training style x before/after w.p. task x test session) analysis of variance. The analysis yielded significant main effects as follows:
The conceptual group recalled significantly more than the procedural group ($F(1,12)=5.70$, $p<0.05$). Recall scores after the word processing task were significantly greater than before the task ($F(1,12)=75.60$, $p<0.01$); and recall scores in the second test session were significantly lower ($F(1,12)=11.01$, $p<0.01$). Also, the three way interaction was significant ($F(1,12)=6.85$, $p<0.05$). Interaction between scores before and after word-processing task and testing session was significantly different ($F(1,12)=5.84$, $p<0.05$). Subsequent analysis revealed that the interaction was due to the simple main effect of the conceptual group showing significantly greater recall on the second testing situation in the test after the word-processing task ($F(1,12)=6.13$, $p<0.05$). The interaction between training style and test session was not significant.

True/false test

The number of correct judgements were scored for each subject, and the mean scores calculated for each group for the four tests (Table 5).

Table 5: True/false test mean scores

	Session 1		Session 2	
	Before W.P. Task	After	Before W.P. Task	After
Procedural	9.14	10.43	10.57	10.14
	(1.77)	(2.30)	(2.23)	(2.04)
Conceptual	11.43	12.00	10.00	11.43
	(1.72)	(1.00)	(1.83)	(1.72)

The conceptual group mean scores were greater than the procedural group in Session 1. There was a tendency for scores to increase after the word-processing task. A 2x2x2 (teaching style x before/after w.p. test x test session) analysis of variance revealed a significant interaction between training style and test session ($F(1,12) = 15.45$, $p < 0.05$). The interaction was explained by the simple main effect of the conceptual group scoring significantly higher in the first test session ($F(1,12) = 5,15$, $p < 0.05$). No significant differences were found in scores in the second session.

DISCUSSION

Users who were offered a conceptual overview of a word processor's hierarchical command structure were expected to develop more complete and accurate mental models of the system. It was thought that possession of such models would prove beneficial to users - resulting in greater learning and better performance, not only immediately following training, but particularly after a period of delay without practice - as in Kieras and Bovair's (1984) comparison of procedural and device model instruction.

These predictions were not entirely accurate. If we look at user's knowledge of word-processing in terms of the *implicit* knowledge which underlies their performance on task, and the *explicit* knowledge which underlies command recall - then this study provides no hard evidence that conceptual training benefits the former, although there is reliable evidence that it benefits the latter.

Task performance was measured in terms of training and task completion times. These were not significantly different for the two groups, although mean times for task completion would suggest a slight advantage for the conceptually trained group. From the comments made by procedurally trained subjects, it seemed that they had been more reliant upon a trial and error strategy - this may account for their relatively poor performance during the first session, and subsequent improvement in session two.

In this experiment, all subjects were given procedurally complete instruction. The significant differences between training styles that were observed in Kieras and Bovair's (1984) simple control panel device experiment and Schmalhofer's (1987) mental model experiment may have resulted because inference was needed to perform the tasks. In this experiment users were not required to make inferences from the information given. Further differences in performance may not have been revealed because the tasks were not particularly complex, nor was a great understanding of the structure required. Kieras and Polson (1985) believe that conceptual understanding may only be important if errors are made or the user meets a novel situation.

These observations concerning the absence of differences in *performance*, are independent of observations of explicit task knowledge. We might infer that procedurally trained subjects found it more difficult to recall appropriate commands, on the basis of their making more referrals to guide sheets. However, while these differences were not significant - there were significant differences in the declarative tests of users knowledge.

Differences between the two groups were most marked under tests of free recall. Four such tests were given, one before and one after each of the two on-line sessions. The conceptual group yielded significantly higher scores on each of the four tests, indicating a greater explicit knowledge of the task. Recall scores for both groups were worse in the second session, as expected; however the conceptual group were expected to be more successful at retaining their knowledge after a few days break - and this difference was not supported. Unexpectedly, however a significant difference was found in the recall scores *after* the word processing task in the *second* session. Thus the advantage for the conceptually trained group was not exhibited immediately after the delay - but only after an on-line revision session. The procedural groups recall was also aided by the revision session, but whereas the conceptually trained users showed no decrement in recall between tests two and four, the procedural group showed a marked loss of information.

In the true/false test the conceptual group were again expected to exhibit increased knowledge in the second session as compared to the procedural group. An interaction between training style and session was found, but was explained by the conceptual group having significantly greater scores in the first test session only. Differences in mean scores and variation between individual subjects was small. The test did not appear to discriminate very effectively between subjects, perhaps due to a ceiling effect. The test did however indicate again that the conceptual group had a greater *explicit* knowledge of the task.

In summary, then, this study has shown that providing users with a conceptual overview of the command system offers no real performance advantages - but does mean that subjects will have explicit access to information concerning those commands. This distinction between implicit and explicit task knowledge has been observed elsewhere (e.g. Berry and Broadbent, 1988; Briggs, 1990) - users may show procedural competence on a task without having any explicit understanding of what information is necessary for task completion. This in itself is not so surprising - As Kieras and Bovair (1984) have observed, a user's level of understanding of the telephone system does not produce measurable differences in performance. Understandably, performance criteria are sometimes considered to be the most

important measures of skilled behaviour - which begs the following question: Given that conceptual training yields improved knowledge, but not improved *efficiency* - then is this any reason for us to show a preference for this type of training?

The goals of training are not only speed and accuracy - the user is expected to retain the skill, build on initial knowledge and transfer the skill to other, similar systems. Transfer of skill is aided by training in the principles of devices, because the cognitive mapping aids reconstruction of new procedures (Singly and Anderson, 1985; Fotheringhame, 1986). Singley and Anderson (1985) suggested that a major source of positive transfer between two line-editors lay in the common functionality between the two. Although the procedures were rather different they had the same logical structure. The conceptual mappings led to enhanced transfer of skill.

Conceptual training in this experiment resulted in greater structural knowledge, which may well indicate that transfer of this skill to a similar device would be superior. Once knowledge has been acquired in a given domain it facilitates acquisition of further related information. Chiesi et al. (1989) proposed that it involves a process where new input is mapped onto existing knowledge structures. As conceptual training can lead to a fuller mental model resulting in greater knowledge retention, it implies that the conceptual group would be facilitated in acquiring further related information.

Finally this study has provided evidence that conceptual training not only results in greater recall, but more significantly facilitates the retention of information after some few days delay. Infrequent users of word processing systems may benefit from an enduring mental model, so on the next occasion they return to the word processor, the mental model can be retrieved from memory which would allow reconstruction of the structure and procedures. As there is often a break between training and practical use of the skills learned this appears to be an important issue in the design of training programmes.

REFERENCES

Allwood, C.M. (1986). Novices on the computer: a review of the literature. International Journal of Man Machine Studies, 25, 633-658.

Briggs, P. (1990). Do they know what they're doing? An evaluation of word-processor users' implicit and explicit task-relevant knowledge, and its role in self-directed learning. International Journal of Man-Machine Studies, 32, 385-398.

Carroll, J.M. (1984). Minimalist Training. Datamation, 30, 125-136.

Carroll, J.M. and Carrithers, C. (1984). Blocking learner error states in a training-wheel system. Human Factors, 26, 377-389.

Carroll, J.M. Mack, R.L., Lewis, C.H., Grischkowsky, N. L. and Robertson, S.R. (1985). Exploring exploring a word processor. Human Computer Interaction, 1, 283-307.

Carroll, J.M. and Mazur, S.A. (1986). LisaLearning. Computer, 34-49.

Carroll, J.M., Smith-Kerker, P.L., Ford, J.R. and Mazur-Rimetz, S.A. (1988). The Minimal Manual. Human Computer Interaction, 3, 257-273.

Chiesi, H.L., Spilich, G.J. and Vass, J.F. (1979). Acquisition of domain-related information in relation to high and low domain knowledge. Journal of Verbal Learning and Verbal Behaviour, 18, 257-273.

Czaja, S.J., Hammond, K., Blascovich J.J. and Swede, H. (1986). Learning to use a word-processing system as a function of training strategy. Behaviour and Information Technology, 5, 3, 203-216.

Fotheringhame, J. (1986). Transfer of training: A field study of some training methods. Journal of Occupational Psychology, 59, 59-71.

Kieras, D.E. and Bovair, S. (1984). The role of a mental model in learning to operate a device. Cognitive Science, 8, 255-273.

Kieras, D.E. and Polson, P.G. (1985). An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22, 403-423.

Norman, D.A. (1983). Some observations on mental models. In D. Gentner and A.L. Stevens (Eds.) Mental Models. Hillsdale, New Jersey: Erlbaum.

Norman, D.A. (1986). Cognitive Engineering. In D.A. Norman and S.W. Draper (Eds.) User-Centred System Design: New Perspectives on Human-Computer Interaction. Hillsdale, N.J.:Erlbaum.

Reder, L.M. and Anderson, J.R. (1980). A comparison of texts and their summaries: Memorial consequences. journal of Verbal Learning and Verbal Behaviour, 19, 121-134.

Reder, L.M. and Anderson, J.R. (1982). Effects of spacing and embellishment on memory for the main points of a text. Memory and Cognition, 10, 97-102.

Reder, L.M., Charney, D.H. and Morgan, K.I. (1986). The role of elaborations in learning a skill from an instructional text. Memory and Cognition, 14(1), 64-78.

Sasse, M.A., Johnson, G.I. and Briggs, P. (1988). Introducing word processing to novice users: A study of 'procedural' and 'conceptual' approaches. In E. McGraw (Ed.) Contemporary Ergonomics, 1988. London: Taylor and Francis, 426-431.

Schmalhofer, F. (1987). Mental model and procedural elements approaches as guidelines for designing word processing instructions. Human Computer Interaction, 269-274.

Singley, M.K. and Anderson, J.R. (1985). The transfer of text-editing skill. International Journal of Man-Machine Studies, 22, 403-423.

Young, R.M. (1983). Surrogates and Mappings: Two kinds of conceptual models for interactive devices. In D. Gentner and A.L. Stevens (Eds.) Mental Models. Hillsdale, New Jersey: Erlbaum.

The use of frequency and the development of knowledge

Gill M. Brown

Department of Psychology, University of Manchester,
Oxford Road, Manchester, M13 9PL, UK.

This paper attempts to reconcile some of the explanations of expert error (i.e. slips and lapses) with some of the theory relating to the use of frequency information. Although there have previously been suggestions that novice errors may be accounted for in terms of frequency gambling, it has been assumed that experts do not use anything as crude as frequency. A study is reported that investigates the use of frequency information during learning in an every day context. An analysis of the errors that subjects made during this study, as they learnt more about the domain, suggests that experts use frequency to as large a degree as novices. The factor that may differentiate experts from novices in any domain is not just the possession of domain knowledge, but also a sophisticated frequency map of the domain. Overall, these results suggest that expert errors are just as likely to be influenced by frequency of encounter in the every day world as novice errors.

INTRODUCTION

What is Cognitive Ergonomics? Cognitive Ergonomics is concerned with the application of cognitive theory and the findings from Cognitive Psychology to real world problems. One possible difficulty with this is that we know a great deal about how people behave in laboratories and particularly about what Kolodner & Riesbeck (1986) call low knowledge experiments, for example dealing with unrelated lists of words—memorizing and recalling them, etc. However, it is difficult to see how much of this type of work can be applied to the real world, when we know that factors such as context are so important.

Consequently, the work reported here has been approached with the aim of dealing with high-knowledge real-world domains. This research, concerned as it is with the use of knowledge in everyday contexts, involved the showing of a series of videos in the form of a story. Moreover, it provided an opportunity to monitor how people used frequency information as learning

progressed. Although, this type of stimuli may have a number of limitations, i.e actually recording the learning process and the nature of the story itself, this method does have its advantages. Specifically, the material presented is controlled, and the objective content of different aspects of the videos (e.g the characters and the places) can be calculated with respect to subjects exposure to these aspects. Therefore, this appears to be a useful and interesting way of looking at how people behave in a more dynamic every day learning situation.

Errors

Having outlined the approach adopted in this research, we need to go on and discuss the central aim of this paper, which is to consider errors and how people behave in every day contexts generally. Reason (1986; 1990; Reason & Mycielska, 1982) has looked at this area in some depth, as has Norman (1981; 1983). They have provided detailed accounts of the kind of errors people make when they are very familiar with a task, such as lapses and slips, etc. Reason has also considered how people deal with the world and the role of frequency in cognition. More specifically, Reason has developed a model of knowledge retrieval: Cognitive Underspecification (1986; 1990) which is derived from the well-established observation that whenever cognitive operations are insufficiently specified, they tend to 'default' to contextually appropriate high frequency responses. This model identifies three distinct strategies of knowledge retrieval: automatic similarity matching between retrieval cues and stored knowledge attributes; frequency gambling to resolve 'fuzzy' matches (i.e where a number of items have been partially matched, and where selection favours the most frequently encountered item); and inference, involving the largely conscious application of logical operators and selected knowledge items brought to mind by similarity-matching and frequency gambling. Furthermore, Reason suggests that although all of these strategies are likely to be involved in knowledge retrieval, their relative contributions are expected to vary firstly, with the nature of the search task and secondly, with the amount of domain knowledge or expertise possessed by the searcher.

Arising out of this is the suggestion that frequency determines errors in novices. Certainly, errors in answers to detailed questions about US States (Brown, 1990) and US Presidents (Reason, 1986) reveal a strong frequency bias. However, we know that experts also make errors of the kind that Reason and Norman have highlighted (i.e. those related to automaticity). But the question remains as to how experts make errors if they are not based on frequency? There are essentially two views of the use frequency in the

every day world. The first, is that it is crude and is used less and less as we learn more about a domain. The second view is that its use may be crude at the beginning of learning, and novices may be over-reliant on it, but as we develop our knowledge of a particular domain we also develop a frequency map for that domain. In other words, our ideas about the relative frequencies of different items in different contexts become more finely tuned. Consequently, rather than using frequency less and less, we continue to use it but in a more sophisticated way, in a way that more carefully matches the particular domain in which we are working. It is these two possibilities that we are concerned with here, frequency is important for novices, but is it as important to the experts who have a good understanding of a particular task, system, domain or machine?

In support of the view that frequency is important to both experts and novices, is the idea that frequency fits in very well with the the work that has been carried out from a connectionist perspective (also referred to as neural nets or Parallel Distributed Processing; Rumelhart, McClelland and the PDP Research Group, 1985). According to this approach, frequency is represented by the strength of the links between various nodes and the patterns across the nodes within the nets /distributed system. However, although this may be an attractive argument it cannot be regarded as proof. The study reported below provides some support for the idea that frequency is used by experts.

A LONGITUDINAL STUDY

The topic of study consisted of a series of six BBC videos of 'The Forsyte Saga' by John Galsworthy. The control group involved each of six different groups of subjects watching just one of the six videos.

Rationale

The decision to select the Forsyte Saga as the course of instruction for this study was taken for a number of reasons. Firstly, one of the main considerations was the independent variable, i.e. frequency of encounter (FOE) or the number of times the subjects would be exposed to an entity or item, could be carefully controlled. Previous studies have been forced to use concepts already known to subjects, where subjects may have differed in their exposure to these concepts. Moreover, it was not clear, in these studies, to what extent subjects ratings of frequency reflected actual frequency in the everyday world. Secondly, the actual nature of the study material is not cumulative, unlike other 'taught' courses. For example, learning a foreign language usually involves (depending on how its taught) memorizing lists

of verbs and their structure in different tenses, and then using them along with non-verbs in speech or written work. On the other hand, 'The Forsyte Saga' represents a 'story', the nature of which prevents items from being learned in equal frequency. Moreover, this knowledge domain may be described as 'dynamic' when compared to other 'static' domains. In the former the story and characters change from video to video, while in the latter, individuals have a particular level of knowledge about that domain. Overall, the study provided an opportunity to look at human knowledge retrieval and the role of frequency in a controlled way that was, nevertheless, related to a story in an everyday context.

Hypotheses

1 The first hypotheses deals with the question as to whether there is any frequency effect. In other words, when individuals in the experimental group respond incorrectly to the questions in the six recognition tasks is there a significant difference between the objective frequency for these incorrect answers compared to the correct frequency score?
2 Assuming that there is a frequency effect we can now go on to consider the second hypotheses which is whether there is any difference in the frequency use of novices and experts (experimental group and control group respectively). Firstly, there is the possibility that experts do not use frequency information greatly, on the other hand it may be the case that there is little or no difference, and experts use frequency just as much as novices.
3 Finally, it has been assumed that the two groups of subjects can accurately be called experts and novices because of the difference in knowledge between the two groups i.e the experimental group have seen all the videos, while the six control groups have watched one video each. For this assumption to be valid we must be able to establish that the experimental group's (experts) performance, in terms of their number of correct answers for each video, will exceed that of the control group (novices).

METHOD

Design

The design used in this longitudinal study was a repeated measures design, i.e all the subjects took part in all six conditions (i.e watched all six videos over the six week period), whereas the 60 control group subjects (independent subjects design) watched just one video each (10 subjects per video). Order effects were not a consideration for the experimental group, as

the videos represented a story and had to be shown in order.

Independent variables:	Time/knowledge of characters. Frequency of encounter for each character.
Dependent variables:	Number answers correct Error frequency score

The calculation of this last measure will be explained in the appropriate results subsection.

Subjects

Twenty-eight subjects took part in the longitudinal study (experimental group) and they were all first and second year psychology under-graduates studying at Manchester University. The only relevant criteria for selecting the subjects that volunteered to take part was that they had not previously seen any of the videos, read the books or watched the series on television. Twenty females and eight males took part in the study and their ages ranged from 18 - 22, with the exception of one subject aged 29.

Sixty subjects took part in the control study. Thirty-five were first and second year psychology undergraduates studying at Manchester University. The remaining 25 subjects were computing undergraduates studying at Manchester Polytechnic. Again, the only criteria for taking part in the study was that the subjects had not previously seen any of the Forsyte Saga videos before, read the books or watched the series on television. There were 42 females and 18 males in the control group. There ages ranged from 18 to 32.

Materials

The recognition test was in the form of a multiple choice questionnaire, where a question was asked and the subjects were required to circle one of the six responses 'a' to 'f' (characters name). Each of the 6 recognition tests consisted of 12 multiple choice questions. The control group subjects were given exactly the same recognition tests as for the experimental group.

Procedure

Over a period of six weeks the 28 subjects in the experimental group, were shown the six videos, one a week, in order. After each video, they were asked to complete the recognition test. The average running time for the videos was approximately 100 minutes each (1 hour 40 minutes) and the recognition task took approximately 15 minutes to complete. For the control groups, a

number of showings were timetabled and people were asked to come and watch just one video. After watching the video, the subjects were asked to complete the accompanying recognition questionnaire.

RESULTS

1 *Does frequency of encounter affect knowledge retrieval?*

The most obvious question that arises when considering this study, is *does frequency have an effect on knowledge retrieval?* If frequency of encounter has no effect then we might expect the average frequency of incorrect responses to be no higher than the average frequency for correct responses for the same questions. Consequently, if there is an effect, then using a one-way analysis of variance, we might expect a significant difference between the frequency score for the incorrect response and the frequency score for the correct response for each question incorrectly answered. In short, if frequency does play a significant role then the mean frequency for incorrect responses should be higher than the mean frequency for the correct responses.

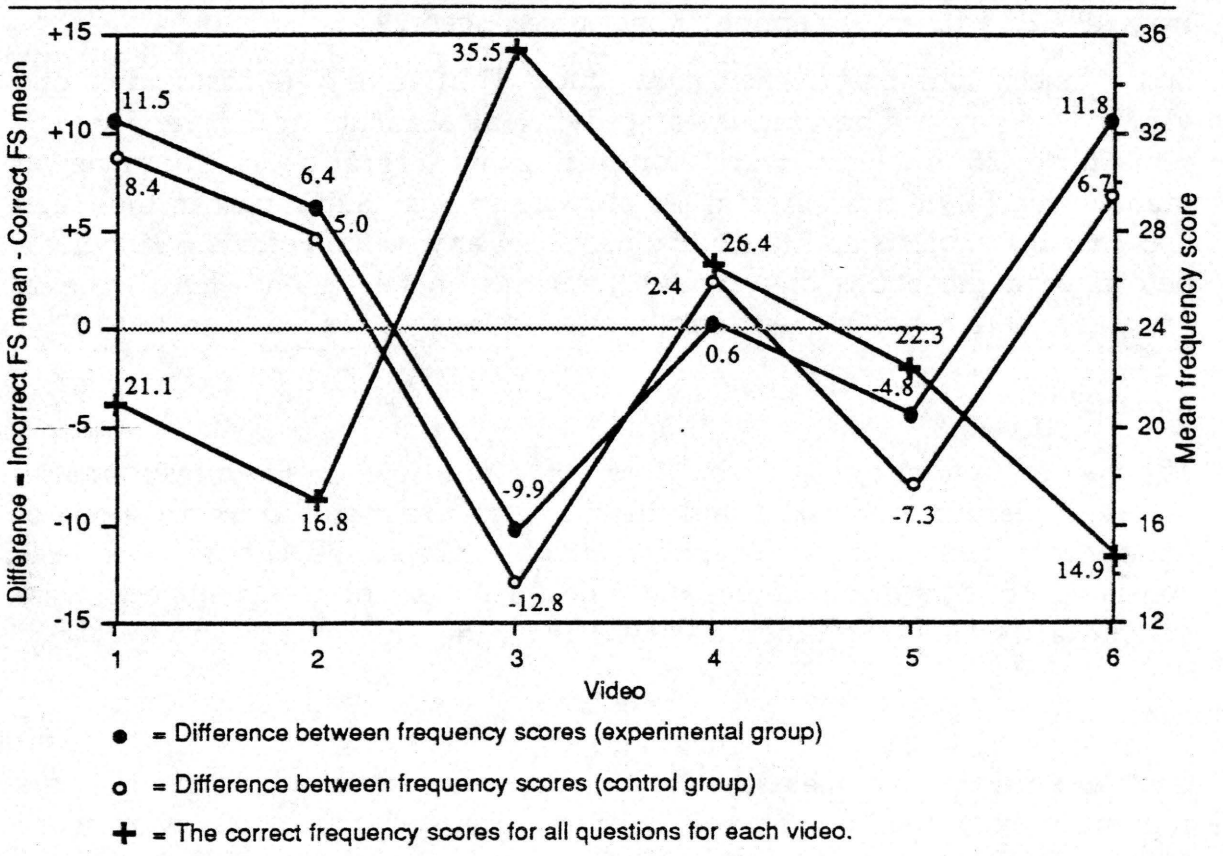


Figure 1. A graph showing the difference between the mean frequency score for incorrect responses and the mean frequency score for correct responses, plotted against video number.

	Mean	SD	N	St. Error	Skewness	Kurtosis
Incorrect Answer Frequency Scores	21.863	14.385	654	0.562	0.294	-0.013
Correct Answer Frequency Scores	24.523	15.98	1359	0.433	0.217	-1.322

Source	Sum of Squares	DF	Mean Squares	F-Ratio	Prob>F
Effect	3129.38	1	3129.38	13.061	**0.0006**
Error	482311.12	2013	239.598		

Table 1. The frequency scores for correct and incorrect responses.

As predicted, the frequency scores for the incorrect answers were higher than the frequency scores for the correct responses to the same questions (see table 1). Moreover, this result was highly significant ($p < 0.001$). The result suggests that the frequency of encounter of items in the world (or in this case in the videos that the subjects were shown) affects the way in which subjects retrieve their knowledge. That is to say, that when cognitive operations were underspecified during the questionnaires that were issued after each video, subjects tended to default to contextually appropriate high frequency responses.

2 *Error frequency scores*

Here, we consider the central question of whether subjects, as they learn more and become more expert, use frequency to a lesser degree. Alternatively, they may use frequency information to the same degree as novices. To answer this question we need to consider the error frequency scores for the experimental group as they progress through the videos.[1] The results for the experimental group can be seen in table 2. The results for the control group are provided for comparison purposes in table 3. The important aspect of these results, explained in more detail elsewhere (Brown, 1990), is that although the error frequency scores are in the opposite direction expected from some videos, the same trend is evident in the control group. This trend is evident in figure 1.

1. The mean error frequency difference was calculated using the following formula:

$$\frac{\sum\left(\frac{i-c}{n}\right)}{N}$$

where i = the frequency score for the incorrect response
c = the frequency score for the correct response
n = the number of errors the subject made
N = the number of subjects who committed errors.

EXPERIMENTAL GROUP	Mean	SD	N	St. Error	Skewness	Kurtosis
Video 1: Incorrect Freq Scores	28.789	10.894	160	0.861	-0.107	-0.570
Video 1: Correct Freq Scores	17.866	8.897	160	0.703	0.167	-0.530
Source	Sum of Squares	DF	Mean Squares		F-Ratio	Prob>F
Effect	9546.2640	1	9546.2640		96.506	**0.0001**
Error	31456.2700	318	98.9191			
Video 2: Incorrect Freq Scores	19.816	14.402	103	1.419	0.469	-0.941
Video 2: Correct Freq Scores	14.146	7.468	103	0.736	0.082	-0.524
Effect	1655.6112	1	1655.6112		12.581	**0.0008**
Error	26846.3100	204	131.5995			
Video 3: Incorrect Freq Scores	20.895	17.322	86	1.868	0.362	-1.379
Video 3: Correct Freq Scores	31.953	19.011	86	2.050	-0.557	-1.304
Effect	5258.1450	1	5258.1450		15.898	**0.0003***
Error	56225.0700	170	330.7357			
Video 4: Incorrect Freq Scores	18.855	13.684	117	1.265	0.276	-0.840
Video 4: Correct Freq Scores	21.009	16.831	117	1.556	0.453	-1.413
Effect	271.3846	1	271.3846		1.154	0.2837
Error	54581.5200	232	235.2652			
Video 5: Incorrect Freq Scores	18.380	12.029	108	1.157	1.102	1.086
Video 5: Correct Freq Scores	22.231	18.120	108	1.744	0.520	-1.295
Effect	801.1852	1	801.1852		3.388	0.0636
Error	50613.1500	214	236.5100			
Video 6: Incorrect Freq Scores	20.675	16.449	83	1.805	0.552	-1.392
Video 6: Correct Freq Scores	9.660	11.705	83	1.285	1.954	3.029
Effect	5034.7070	1	5034.7070		24.706	**0.0001**
Error	33421.3040	164	203.7884			

Table 2. The difference in frequency scores for correct and incorrect responses (experimental group). The p values marked with an astrix denote significant relationships in the opposite direction to that predicted

CONTROL GROUP	Mean	SD	N	St. Error	Skewness	Kurtosis
Video 1: Incorrect Freq Scores	27.250	9.375	64	1.172	0.442	0.016
Video 1: Correct Freq Scores	18.469	8.716	64	1.089	0.119	-0.576
Source	Sum of Squares	DF	Mean Squares		F-Ratio	Prob>F
Effect	2467.5312	1	2467.5312		30.117	**0.0001**
Error	10323.4400	126	81.9320			
Video 2: Incorrect Freq Scores	20.293	13.704	58	1.799	0.426	-0.787
Video 2: Correct Freq Scores	16.621	8.352	58	1.097	0.060	-0.799
Effect	391.1120	1	391.1120		3.037	0.0803
Error	14679.6720	114	128.7691			
Video 3: Incorrect Freq Scores	18.458	17.478	65	2.168	0.476	-1.455
Video 3: Correct Freq Scores	33.077	18.174	65	2.254	-0.685	-1.017
Effect	6945.2330	1	6945.2330		21.848	**0.0001***
Error	40689.7900	128	317.8890			

	Mean	SD	N	St. Error	Skewness	Kurtosis
Video 4: Incorrect Freq Scores	22.897	16.010	58	2.102	0.157	-1.101
Video 4: Correct Freq Scores	20.241	16.001	58	2.101	0.601	-1.096

Source	Sum of Squares	DF	Mean Squares	F-Ratio	Prob>F
Effect	204.4483	1	204.4483	0.798	0.3772
Error	29204.0010	114	256.1754		

Video 5: Incorrect Freq Scores	14.492	10.719	61	1.372	0.338	-0.885
Video 5: Correct Freq Scores	22.820	18.686	61	2.392	0.399	-1.532

Effect	2115.2790	1	2115.2790	9.117	**0.0034***
Error	27842.7700	120	232.0230		

Video 6: Incorrect Freq Scores	17.474	14.757	78	1.671	1.135	-0.273
Video 6: Correct Freq Scores	11.615	12.762	78	1.445	1.486	1.001

Effect	1338.7753	1	1338.7753	7.035	**0.0087**
Error	29307.5100	154	190.3085		

Table 3. The difference in frequency scores for correct and incorrect responses (control group). The p values marked with an astrix denote significant relationships in the opposite direction to that predicted.

Consequently, we can conclude that there is no difference between the experimental and control groups in terms of their use of frequency. Therefore, it appears as though frequency is as important to experts as it is to novices. This, however, assumes that the experimental group are experts by the time they have reached the end of the course of videos.

3 *Number of correct responses*

Although we might wish to conclude that frequency is as important to novices, how do we know that the experimental group are experts? In short, has learning occurred in the experimental group? A count of number correct is provided in table 4. A histogram is provided in figure 2, and line graph of these results is provided in figure 3.

	Mean	SD	N	St. Error	Skewness	Kurtosis
Video 1: Experimental Group	6.286	1.941	28	0.367	-0.185	0.219
Video 1: Control Group	5.600	1.897	10	0.600	-0.271	-1.542

Source	Sum of Squares	DF	Mean Squares	F-Ratio	Prob>F
Effect	3.4647	1	3.4647	0.930	0.3434
Error	134.1143	36	3.7254		

Video 2: Experimental Group	8.321	1.634	28	0.309	-0.061	-0.783
Video 2: Control Group	6.200	1.874	10	0.593	-0.168	-0.865

Effect	33.1613	1	33.1613	11.511	**0.0020**
Error	103.7071	36	2.8808		

Video 3: Experimental Group	8.929	1.844	28	0.349	-0.446	-0.708
Video 3: Control Group	5.500	2.173	10	0.687	-0.351	-1.247

Effect	86.6166	1	86.6166	23.208	**0.0001**
Error	134.3573	36	3.7321		

	Mean	SD	N	St. Error	Skewness	Kurtosis
Video 4: Experimental Group	7.893	1.618	28	0.306	-0.289	-0.357
Video 4: Control Group	6.300	2.406	10	0.761	-0.231	-0.645

Source	Sum of Squares	DF	Mean Squares	F-Ratio	Prob>F
Effect	18.6951	1	18.6951	5.482	**0.0235**
Error	122.7787	36	3.4105		

Video 5: Experimental Group	8.143	2.085	28	0.394	-1.411	3.036
Video 5: Control Group	5.900	1.969	10	0.623	0.596	0.178

Effect	37.0662	1	37.0662	8.760	**0.0055**
Error	152.3286	36	4.2314		

Video 6: Experimental Group	9.036	1.953	28	0.369	-1.055	1.124
Video 6: Control Group	4.200	1.476	10	0.467	0.441	-0.398

Effect	172.3041	1	172.3041	50.610	**0.0001**
Error	122.5643	36	3.4046		

Table 4. The difference between the experimental group and control group for each video, in terms of the number of correct responses.

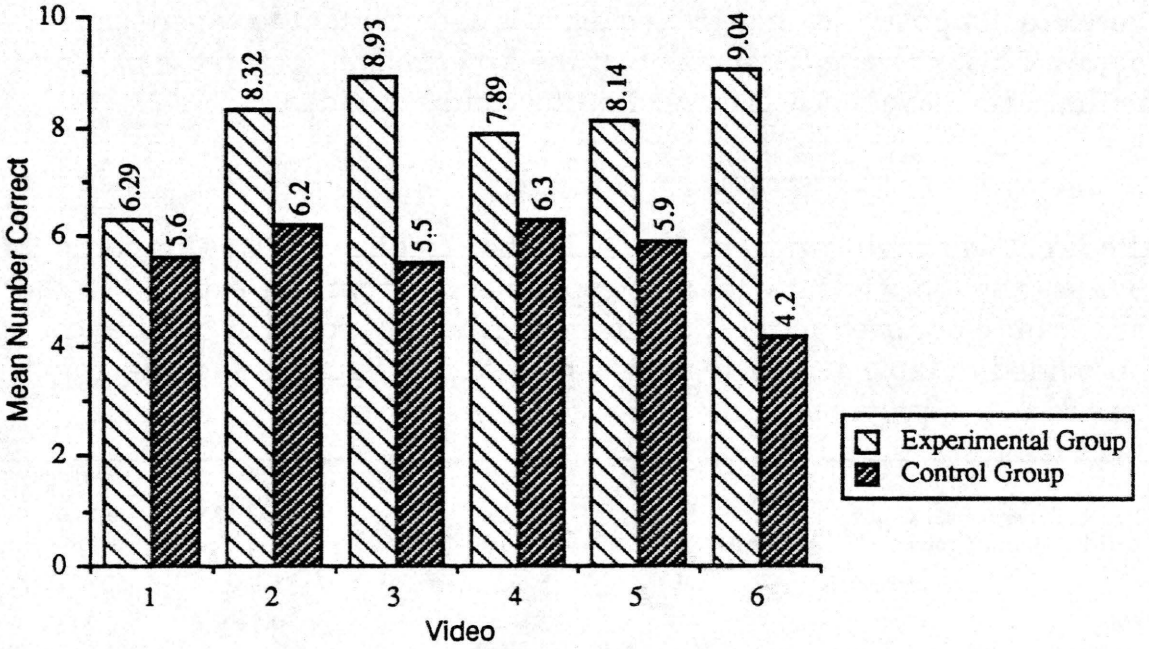


Figure 2. A bar graph showing the mean number correct for the experimental and control groups for each video.

From the results presented above it can be concluded that learning did occur in the experimental group, and the finding suggested in the previous results subsection, that frequency is as important to experts as it is to novices, is valid.

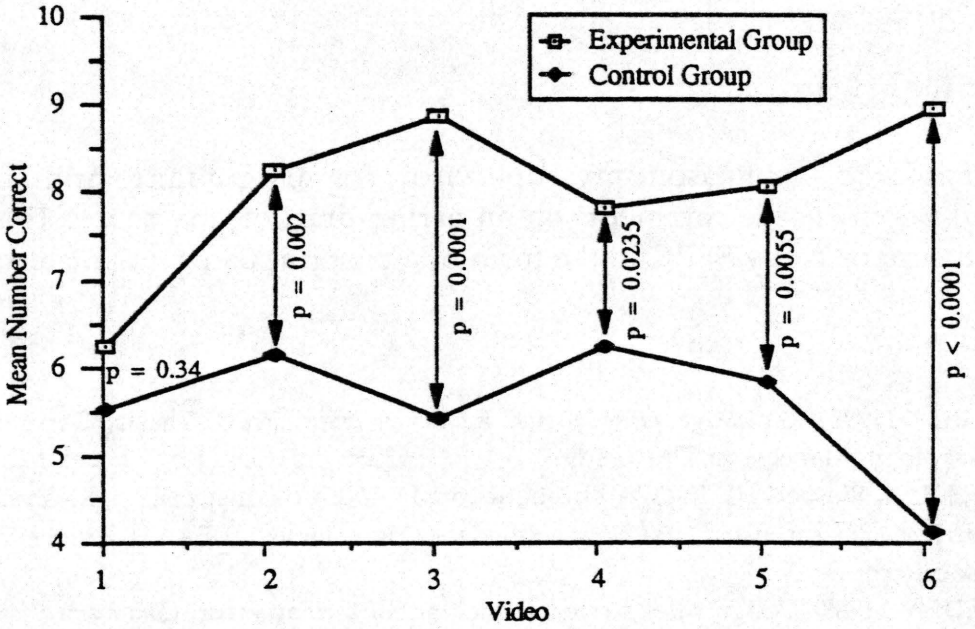


Figure 3. A line graph showing the difference between the mean number correct for the experimental and control groups. The p values are for analyses of variance. These can be seen in table 4.

DISCUSSION

There was no difference between the experimental and control group in terms of their use of frequency information. Nevertheless, despite the lack of any difference in the use of frequency between the two groups, the experimental group do appear to have undergone a learning process. Their performance exceeds that of the control group for all videos, except, of course, video 1. Similarity matching may have increased for experts, as the accuracy of these subjects increased with learning. However, there might be some doubt as to whether frequency gambling and similarity matching can really be separated, given the continuing use of frequency by experts. These concepts are metaphors and in reality they may well be part of the same process of knowledge retrieval. The notion of a frequency map, based loosely on a neural net analogy, does provide a model for conceptualizing knowledge retrieval as a unified process. Moreover, this would explain why experts used frequency information to the same extent as novices.

Finally, the results of this study have shown experts do use frequency information, but probably in a more refined way than novices—with a better developed frequency map. However, the underlying principle or basic premise of Underspecification—i.e when cognitive operations are under-

specified they tend to default to contextually appropriate high frequency responses, still holds. But more importantly, it appears to be true not only for novices but for experts also.

ACKNOWLEDGEMENTS

Many thanks to Jim Reason, my supervisor, for his guidance and support, and Paul Booth for his comments on an earlier draft of this paper. This work has been supported by SERC in the form of a postgraduate studentship.

REFERENCES

Brown, G. M. (1990) *Knowledge retrieval and frequency maps.* Ph.D. Thesis, Department of Psychology, Manchester University.

Kolodner, J. L. & Riesbeck, C. K. (1986) Introduction. In: J. L. Kolodner & C. K. Riesbeck (Eds.), *Experience, memory and reasoning.* Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates.

Norman, D. A. (1983) Design rules based on analyses of human error. *Communications of the ACM, 26 (4),* 254-258.

Norman, D. A. (1981) A psychologist views human processing: human errors and other phenomena suggest processing mechanisms. In: *Proceedings of the International Joint Conference on Artificial Intelligence, Vancouver.*

Reason, J. T. (1990) *Human Error.* Cambridge, UK: Cambridge University Press.

Reason, J. T. (1986) Cognitive underspecification: Its varieties and consequences. In: B. Baars (Ed.), *The psychology of error.* London, UK: Plenum.

Reason, J. & Mycielska, K. (1982) *Absent-minded? The psychology of mental lapses and everyday errors.* Englewood Cliffs, New Jersey, USA: Prentice-Hall.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group, (1986) *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, Massachusetts, USA: MIT Press.

TYPE OF COMMAND USED DURING A SEMANTIC CATEGORIZATION TASK ON P.C. : THE ROLE OF SUBJECTS' LEVEL OF EXPERIENCE AND TASK CHARACTERISTICS .

DEWIER Agnès* **and KARNAS Guy**
Laboratoire de Psychologie Industrielle et Commerciale
Université Libre de Bruxelles
50 Avenue F.-D. Roosevelt - B 1050 Bruxelles - Belgium

SUMMARY

This study aims at highlighting the existence of an interaction between cognitive processes due to expertise and general task independent processes in Man-Computer Interaction. It examines the kind of command (simple or high-level) subjects with or without computer background chose during a computer-driven semantic categorization task acccording to the characteristics of the material (e.g. the length of the lists of words). Characteristics of simple and high-level commands are discussed. The prediction is that subjects would try to maintain a balance between the load imposed by the task and the load imposed by command complexity, independently of their experience level.

1. INTRODUCTION

In a previous study (Dewier and Karnas, 1989), experienced and naive subjects were submitted to a text-editing task. Blocks of text (1 character, 1 word, 1 sentence or 1 paragraph) had to be moved. Two strategies were possible : (1) to use the simple command of deletion, move the cursor and type text from keyboard, or (2) to use a more complex "cut and paste" command with automatic reinsertion of the block from a buffer memory. Although naives are supposed to prefer simple commands, the result showed that the longer the block to move was, the more the subjects (whether experienced or not) used the cut and paste command, probably to avoid too much keying.

This unexpected similar behavior in experienced and naive subjects may be explained by the intervention of two different processes during Man-Computer Interaction (MCI) : domain-specific processes due to expertise and general domain-independent processes. The effects of expertise are to be found in the time required to make the correction or in the number of errors, which are smaller for the experienced subjects. The domain-independant processes would act more on the strategies developed by the subjects, e.g. on the choice of the type of command with respect to the length of the block to move.

One cannot ignore the great part played by the user's level of experience in MCI. Nevertheless, the user is first of all a human being, and his or her behavior depends also on factors external to the task. As stated by Moran (1981, p.3) : "The user is engaged in a goal-oriented activity. He attempts to accomplish his goal as effortlessly as possible, within the constraints imposed upon him by the structure of the task, by what he knows and by his own processing limits". The same idea has also been expressed by Embley

* Research Assistant of the National Fund for Scientific Research - Belgium.

and Nagy (1981, p.46) : "Our inability to learn, remember, and effectively use complex command sets, balanced against our desire to achieve editing objectives with a minimum expenditure of effort and time limits the range of reasonable design options.". This may be one of the reasons why computer systems are not often used as efficiently as they could, be for people do not want to make the effort to reach a high level of use when they can manage well enough with basic commands.

One interpretation of the results obtained in the experiment mentioned above may be, then, that subjects tried to maintain a balance between the load imposed command complexity and the load imposed by the amount of typing.

The experiment presented here has been designed to test this "balance" hypothesis against the more classical "level of experience" hypothesis which predicts that naive subjects prefer simple commands and experienced users, complex commands. Experiments on the interaction between the type of commands and the characteristics of the task can be found in the literature, but they do not deal with the level of experience of the subjects at the same time. For example, Card, Moran and Newell (1980) showed that during a text-editing task, their experienced subjects chose between several ways of making a correction according to simple rules which were dependent on the characteristics of the task. For example, the command used to move the cursor depended on the number of lines between the current line and the target line. In the present experiment, the characteristics of the user, those of the task, and those of the commands will be varied.

1.1. The characteristics of the commands.

Morrison, Green, Shaw and Payne (1984) evaluate the complexity of a command on the basis of the length of the transaction cycle to which it gives rise (the transaction cycle being the number of actions required before the system responds). Simple commands lead to short transactions. They take few parameters, but achieve rather little. They are conceptually simple and knowledge of the result comes quickly. Because of the limited effect of each command and the step by step feed-back, errors are quite easy to correct. The authors assume that short transactions should be preferred by the naive subjects. High-level commands lead to long transactions. They take more parameters but achieve more. Fewer dinstinct command names have to be learned and fewer commands are necessary to perform a task. The authors assume that only the experienced user is happy with long transactions. It may be added that the global and delayed feedback of long transactions makes errors more complicated to undo. A representation of how the system works is needed to conceptualize the effect of these commands.

This distinction between simple and high-level commands is too dependent on the command length, due to the fact that Morrison and al. worked with a line-editor. In the examples of commands they give, a long transaction is a combination of short transactions. In today's screen text-editors, simple and high-level commands available to achieve a given editing objective cannot be compared in such a straightforward way. The user works directly "into" the text. The shift from line editors to screen editors has made the interactive command language evolve from a strict set of command words combined into phrases according to syntactic rules to new styles of interaction, such as choosing options in menus or pointing to icons. As a consequence, the interaction consists now in hitting a series of specific keys (arrows, page-up, delete, insert, function keys, or the mouse buttons...), which each affects the task environment. Let us take the example of modifying a word in a screen editor :

With simple commands, the user has to search the text for the word, making it scroll on the screen with some cursor moving key, then locate the cursor precisely on the line, use the backspace or delete key to erase the word and type the other one at the keyboard. Now using the high-level "Modify" commmand, maybe the user has to have the menu displayed (pressing Escape), to choose the right option (entering its first letter or a number), to type the word to change in one field and the word to substitute in another field and to press Return.

It seems obvious in this example that the goal of having the word changed into another is likely to be achieved faster with the high-level command than with the basic ones. At a finer grain of analysis, considering the delay between each keystroke and its specific effect, it is also obvious that the high-level command may be decomposed into a series of transactions as short as those obtained with the simple commands. A notion of "physical distance" should be introduced to allow to distinguish between both ways of modifying the word.

The "physical distance" refers to where on the screen the user is working : simple commands generally allow the user to handle the object of his or her manipulation directly (e.g. inserting or deleting a character where the cursor is). With a high-level command, the user has no longer any direct control over the object (e.g. the word in the buffer is not accessible).

Besides those surface changes in the execution stage or in the operators, deeper structures are involved, e.g. the procedures or methods underlying the user's performance must also be considered in trying to characterize simple and high-level commands. With an interactive command language such as the one described by Morrison et al., the sequence of actions leading to goal achievement is roughly the same with short and long transactions, except that actions are executed one at a time with short transactions or all at the same time in a long transaction. This is no longer the case with the new styles of interaction. A current operation such as modifying a word may give rise to different ways of carrying it out spreading on a simplicity-complexity continuum. The position on the continuum varies in relation to the operators chosen by the user. An intermediate solution in the example would be to follow the first procedure but choose a specific command to delete the wrong word in one go. All these possibilities may be seen as implicit adaptative interfacing.

Considering command use in a larger context involving all stages or components of MCI models (Norman (1984), Card et al. (1981)), it is suggested that simple and high-level procedures do not break the high-order objective of the user into the same sequences of low-order objectives. With the simple procedure, the feed-back of any user's action is directly related to the objective of seeing a new word on the screen in place of the old one. Moreover, the sequence of actions closely corresponds to what would be done in a paper-pencil situation (the only one to which naives can refer). In the high-level procedure, only the last feed-back globally fulfills the first intention. Intermediary steps are not directly related to the word modification : the incorrect word must be typed, and the new correct word is not entered at the right place in the text.

The translation of the intention or goal into a method and the selection among the available commands or operators may require more or less transformation. Treu (1975) has proposed a model evaluating the mental work required by the use of a specific command, which is based on the number of transformations that take place between an intention and a correct command.

In conclusion, if both simple and high-level procedures can achieve the same editing objective in the same temporal delay, and can both be decomposed into short transactions, they are not underlined by the same sequences of sub-goals. Moreover, high-level procedures often require the use of commands which have nothing to do, from the user's point of view, with the editing objective (such as Escape in the example), but which are necessary from the system's point of view. These may be called computer-oriented commands, as opposed to user-oriented commands.

1.2. The characteristics of the task.

The physical characteristics of the task should be distinguished from its complexity. A user may prefer to use high-level commands such as defining a glossary, to avoid long, complex or repetitive entries from the keyboard, or such as defining macros to spare keying on frequently used combinations of simple commands. On the contrary, due to the user's processing limits, high-level commands in a difficult task may be much too demanding.

1.3. The characteristics of the user.

The most important one is level of experience. It may be assumed to modulate the load imposed by high-level commands, making it less for an experienced operator than for a novice. The user may also prefer commands that allow him to work according to his or her cognitive strategies or to handle information according to his or her favorite representation mode. More subjective dimensions such as the satisfaction the operator gets by using a given command or by feeling that he or she masters the system or uses it efficiently must also be taken in consideration.

Hypothesis.

The characteristics of simple and high-level commands alone could lead to the conclusion that, given the needs of experienced and naive users, experienced people would rather use high-level commands whereas naives would prefer the simple ones. But the results obtained in the preliminary experiment suggest that the characteristics of the task must also be taken into account together with those of the user.

It is predicted that in a situation where the task is simple and the subjects have the choice between commands of different levels of complexity and efficiency, the performance of experienced and naive users will be quite similar with respect to the choice of the type of commands according to the characteristics of the task : subjects are expected to use simple commands when the physical load imposed by the task is low, and high-level commands when the physical load of the task is high. When the task is difficult, the naive users are expected to return to simple commands. Experienced users are not involved, as computer experience is assumed to reduce the load imposed by high-level commands.

2. METHOD

The experiment is designed in such a way as to allow naive and experienced subjects to choose between two types of commands (simple or high-level) to carry out a task varying with respect to complexity and physical load. The experiment was conducted in two sessions. The subjects had to classify words presented in lists into semantic

categories. The length of the lists has been varied (number and length of words). In the Physical Load condition, the words have to be classified into the same five semantic classes, which are known by the subjects. In the Difficult Task condition, the semantic classes are different and not given to the subjects. For each list, the subjects could choose between simple or high-level commands.

2.1. Material

The lists of words.

15 experimental lists (session 1) and 14 experimental lists (session 2) were constructed. The words in the Physical Load condition lists belong to the same 5 semantic classes (Money, Language, Transport, Law and Religion). Short words are 4- or 5-letters long, medium words are 8-letters long and long words are 12- or 13-letters long.

For session 1, the Physical Load condition comprises 3 lists of 15 short words (for a total number of characters ranging from 60 to 75), 3 lists of 15 long words (from 180 to 195 characters), 3 lists of 25 medium words (200 characters) and 3 lists of 5 medium words (40 characters). The Difficult Task condition consists of 3 lists of 15 medium words (120 characters).

For session 2, the Physical Load condition consists of 12 lists ranging from 40 characters to 420 characters : 35 long words (420), 35 medium words (280), 23 long words (280), 35 short words (180), 23 medium words (180), 15 long words (180), 10 long words (120), 23 short words (100), 10 medium words (80), 19 short words (80), 6 long words (80) and 10 short words (40). The Difficult Task condition comprises 2 lists of 23 medium words (180 characters).

The screen.

The list of words is displayed at the top of the screen. The middle part of the screen consists of a 5-columns array. Each of the columns corresponds to a semantic category in which words must be placed. In the physical load condition, the columns headings indicate the semantic classes. In the difficult task condition there are no headings.

The commands.

The SIMPLE commands : each word of the list is typed from the keyboard and displayed immediately in a light bar cursor which can be moved from one column to the next with the arrow keys. Spelling errors are corrected with the Backspace key. The Return key confirms the entry of the word in the chosen category. This method is close to the manual situation and fits the characteristics of the simple commands described earlier.

The HIGH-LEVEL commands : Hitting the F1 key makes brackets appear around the first word of the list. They can be moved from word to word with the arrow keys. Pressing Return highlights the word between brackets. Several words of the same category may be marked, and a highlighted word may be unmarked by pressing Return. The F4 key exits the Word Choice mode. Hitting the F2 key selects the Category Choice mode. A cursor appears in the first column of the array. It can be moved from one category to the next with the left and right arrow keys. Return highlights the heading of

the category in which the cursor is. Hitting the F5 key leaves the Category Choice mode. Hitting the F3 key transfers the highlighted words into the selected category.

A third set of commands was introduced which combined the disavantadges of both simple and high-level commands (keying + functions keys). It will not be considered here.

The type of commands was chosen from a menu presented to the subject together with each list of words. The simple commands were called "Keyboard" and the high-level commands were called "Mark", in order to avoid desirability effects.

Simple and high-level commands have been compared to eachother from a theoretical point of view. The minimum number of keystrokes (MNK) required to classify each list with each type of command has been computed under the condition of an error-free performance. The result is that the MNK with the simple commands is always higher than with the high-level command, excepted for the 10 short words list in session 2. One can conclude that the physical load is higher with the simple commands than with the high-level commands.

As far as the complexity of both types of commands is concerned, it can for example be computed that the percentage of computer-oriented keystrokes is always greater with the high-level commands than with the simple commands. The sequence of sub-goals with the simple commands is close to the one subjects would follow in a paper-pencil situation, which is not the case for the high-level commands. The mode changes required by the high-level commands is another point in favor of their greater complexity (Monk, 1986).

Procedure.

The experiment was run in two hour-long sessions separated by a period of 24 hours. Subjects were divided randomly into two groups (Practice or Control). Each group was composed of 6 Experienced subjects and 6 Naive subjects. Naives had never worked on a computer. Experienced subjects were all familiar with applications such as databases, word processors or spread-sheets.

In the Practice group, during session 1, each type of commands is explained and then practised with one warming-up list followed by 5 experimental lists. The subjects are not allowed to choose between the commands during the session 1. They have an equivalent amount of training with each type of commands.

In the Control group, during session 1, all types of commands are first explained then practised each with one warming-up list, one after the other. The subjects are then presented with all the remaining experimental lists, one at a time, for which they already have the choice between the types of commands.

Session 2 is the same in both groups. Subjects are given a recall of the commands with the warming-up lists. They are free to choose the type of commands for each of the next 14 experimental lists.

The order of presentation of the commands and of the lists was balanced within both groups.

3. RESULTS

3.1. Classification times according to list length.

The data from the subjects in the Practice group during session 1 provide the opportunity of comparing the classification times with simple and high-level commands according to the list length. It should be remembered that those subjects were constrained to use each type of commands with each type of lists.

Table 1

list		Simple	High-level	S-H
15 short words (64)	N	159.4 (15.1)	181.2 (21.3)	-21.8
	E	115.9 (24.4)	152.8 (31.3)	-36.9
5 medium words (40)	N	55.7 (6.0)	89.3 (29.2)	-33.6
	E	35.9 (3.7)	72.1 (15.7)	-36.2
15 long words (184)	N	277.1 (75.3)	199.2 (38.2)	77.9
	E	182.7 (45.6)	143.8 (31.3)	38.9
25 medium words (200)	N	309.5 (51.5)	234.9 (62.8)	74.6
	E	200.8 (43.3)	172.6 (19.8)	28.2

Table 1 gives the mean classification times in seconds for the 6 Naives and the 6 Experienced subjects of the Practice group during session 1. Mean times (and standard deviations) are given for each of the four Physical Load condition lists and for each type of command (simple or high-level). The last column shows the difference between the mean times with the simple and high-level commands.

A three-way analysis of variance (Type of List X Level of Experience X Type of Command) has been run. Main effects of Type of List and Level of Experience are significant (respectively, $F=90.6$, 3 df, $p<.0001$ and $F=49.3$, 1 df, $p<.0001$), but not the main effect of Type of Command. The interactions between Type of List and Level of Experience, and between Type of List and Type of Commands are also significant (respectively, $F=4.3$, 3 df, $p<.008$ and $F=10.8$, 3 df, $p<.01$).

Generally the naives are slower than the experienced subjects, although the difference depends on the type of list. The classification time of a list depends on its length, the shorter ones being classified faster than the long ones. Overall, the time with the simple commands is not different from that of the high-level commands, but how they compare depends heavily on the characteristics of the lists.

More local comparisons show that there is no significant difference between types of command for the first list (15 short words), that the simple commands are faster than the high-level ones for the second list (5 medium words, $t=3.13$, 5 df, $p<.025$ for the naives and $t=5.46$, 5 df, $p<.005$ for the experienced subjects). For the longer lists, there is no time difference between types of commands in the experienced subjects, whereas the naives are faster with the high-level commands ($t=3.24$, 5 df, $p<.025$ for the third list and $t=3.39$, 5 df, $p<.01$ for the fourth list).

3.2. Frequency of Simple Commands use (SC) *.

Table 2

		all lists	PL vs DT	Short vs Long	
Control					
Session 1	N	5.20 (3.0) 5	5 4	3.4 (1.5) 5	0.8 (0.8) 3
	E	3.75 (1.3) 4	4 0	3.5 (1.3) 4	0.0 (0.0) 0
Session 2	N	4.25 (4.7) 4	4 1	3.0 (2.5) 4	0.8 (1.5) 1
	E	2.00 (1.0) 5	5 0	1.8 (0.8) 5	0.2 (0.5) 1
Practice					
Session 2	N	3.00 (1.3) 6	6 1	2.3 (1.3) 6	0.5 (0.8) 2
	E	4.00 (1.0) 3	3 1	3.3 (0.6) 3	0.3 (0.6) 1

Table 2 gives the mean number of lists classified with the simple commands for each level of experience in the Control group for both sessions and in the Practice group for session 2 (all lists). The maximum number of lists is 15 for session 1 and 14 for session 2. The standard deviation is given between brackets, followed by the number of subjects (maximum 6) who used the simple commands for at least one list. The "PL vs DT" column gives the number of subjects who used the simple commands to classify at least one list of the Physical Load (PL) and the number of subjects who used the simple commands to classify at least one list of the Difficult Task (DT) condition. The "Short vs Long" column gives the mean number of short lists (lesser than or equal to 120 characters, maximum 6) and long lists (longer than 120 characters, maximum 6), that have been classified with the simple commands. Each mean is followed by the corresponding standard deviation and the number of subjects.

No overall significant differences are found between naives and experienced subjects : naives are not more numerous in using the SC than experienced subjects and the naives who used the SC did not classify more lists with them than the experienced subjects. During session 2, the experienced subjects of the Practice group classified more lists with the SC than those of the Control group.

Physical Load condition against Difficult Task condition

The results show that subjects are more numerous in using the SC for the lists of the Physical Load condition than for those of the Difficult Task condition. During session 2, and considering all subjects collectively, it was found that 6 of them never used the SC, 3 of them used the SC with both types of lists, 15 of them used the SC only for the Physical Load lists and no subject used the SC only for the Difficult Task lists (Mac Nemar corrected for continuity = 13.07, $p<.001$). The only significant difference between naive and experienced subjects is obtained in the Control group during session 1, where 4 naives out of 6 used the SC with the Difficult Task lists against no experienced subject ($p=.03$, Fisher's exact test).

---

* In order to present the results as completely as possible, only main significant differences will be mentioned here. Details about the data and their statistical treatments are available on request to the authors.

Short lists against long lists

The 12 lists of the Physical Load condition have been divided into two groups according to their total number of characters : the 6 short lists are shorter than or equal to 120 characters and the 6 long lists are longer.

During session 2, and considering all subjects collectively, it was found that six of them never used the SC, five of them used the SC with both short and long lists, and the remaining 13 subjects used the SC only with the short lists (Mac Nemar corrected for continuity = 11.08, $p<.01$). The distribution is also significant in the Control group during session 1 ($p<.05$). All the comparisons made between the number of short and long lists classified with the SC are significant : subjects used the SC more often with the short lists than with the long ones.

The comparison between naive and experienced subjects in each group once for the short lists and once for the long lists are all non significant : regardless of their level of experience, most of the subjects used the SC with the short lists and most of them did not use the SC with the long lists. The only significant result is that the naives in the Control group during session 1 classified more long lists with the SC than the experienced subjects.

4. DISCUSSION - CONCLUSION

It was shown in the Practice group during session 1 that the simple commands are faster or do not take more time than the high-level commands with the short lists, whereas they are significantly slower with the long lists, at least for the naives.

This may be due to the fact that the naives were poorer typists than the experienced subjects. This result provides some support for the fact that to be efficient with each type of list, subjects should not always use the same commands. It was computed that the number of keystrokes was always higher with the simple commands than with the high-level commands. The result discussed here also supports the idea that repetitive use of a given number of computer-oriented keys, as is the case with the high-level commands on a short list, may be more time consuming than the use of about the same number of subject-oriented keys.

The results found in subjects who had the choice between commands confirm globally the balance hypothesis : the naives were not more numerous than the experienced subjects in using the simple commands, neither did they choose them more often. The simple commands were mainly used with the short lists, when typing was reduced, and the high-level commands were preferred when they could spare a lot of keying. The subjects' choice depended, then, more on the task characteristics than on their computer experience.

Evidence against the hypothesis is the low frequency of simple commands use in the naives with the Difficult Task condition lists during session 2. One explanation may be that, although the high-level commands present some of the characteristics of complexity that have been discussed in the Introduction, they may provide a "cognitive support" to the subjects' mental operations during the task. For example, moving the brackets from word to word in the list may help its reading, a pre-classification is possible by the highlighting of a series of words before execution, without over-loading

the subject's memory. This last point is particularly helpful in the Difficult Task condition, where the semantic classes are unknown.

Two results found in the Control group during the first session plead in favour of the level of experience hypothesis : more naives than experienced subjects used the simple commands with the Difficult Task condition lists, and the naives also classified more long lists with the simple commands than the experienced subjects. It is worth noting that these results were obtained during the first session in the Control group, that is to say when the subjects had no experience with the task and the commands. These differences between levels of experience no longer showed up during the second session. One effect of the kind of training is that during session 2, the experienced subjects in the Practice group chose the simple commands more often than those in the Control group.

The tendency for the naives to use the simple commands more often than the experienced subjects, as well as the inclination of the experienced subjects to neglect the simple commands, even when they prove to be more efficient than high-level ones, may, then, be reduced with some practice, provided that the subjects are given the opportunity (or are constrained) to do it.

REFERENCES

CARD, S., MORAN, T. and NEWELL, A. (1980). Computer text-editing : an information processing analysis of a routine cognitive skill. *Cognitive Psychology*, **12**, 32-74.

DEWIER, A. and KARNAS, G. (1989). Level of experience in text-editing : the role of mental representations on performance. Communication presented at the 8th Interdisciplinary Workshop on "Informatics and Psychology" : Mental models and human-computer interaction. Schaerding (Austria), 16-19 May 1989.

EMBLEY, D.W. and NAGY, G. (1981). Behavioral aspects of text-editors. *Computing Surveys*, vol. 13, n°1, 33-70.

MONK, A. (1986). Mode errors : a user-centred analysis and some preventative measures using keying-contigent sound. *International Journal of Man-Machine Studies*, **24**, 313-327.

MORAN, T. (1981). An applied psychology of the user. *Computing Surveys*, vol. 13, n°1, 1-11.

MORRISON, D., GREEN, T., SHAW, A. & PAYNE, S. (1984). Speech-controlled text-editing : effects of input modality and of command structure. *International Journal of Man-Machine Studies*, **21**, 49-63.

NORMAN, D. (1984). Stages and levels in human-machine interaction. *International Journal of Man-Machine Studies*, **21**, 365-375.

TREU, S. (1975). Interactive command language design based on required mental work. *International Journal of Man-Machine Studies*, **7**, 135-149.

IS ABILITY TO UNDERSTAND A QUERY LANGUAGE FACILITATED BY TRAINING PERTAINING TO THE "DEEP STRUCTURE" OF THE SEARCH TASK?

Lena Linde and Monica Bergström, Dept. of Psychology,
University of Stockholm, S10691 Stockholm.

ABSTRACT

The present experiment was designed to investigate the impact on performance, while learning a query language, of specific instructions pertaining to the "deep structure" of the search tasks. The deep-structure instructions comprised training in analysing queries in natural language in terms of their logical constituents. The effect of such instructions was investigated for two different types of query languages. Learning performance was assessed from number of committed errors and assistances given by the experimenter, during performance of query construction or very simple programming tasks. A two-factorial design was used. One factor was prior training in analysing deep-structure of queries vs. no such training. The other factor was type of query language being taught. One was a textual and one was a "graphic" query language. Twenty subjects, were randomly assigned to the four experimental conditions (2 male and 3 female in each group). A two-way analysis of variance, with no. of errors and assistances on the query construction tasks as dependent variables gave a significant interaction ($P<.05$) between the two experimental factors but no significant main effect was found. One-way analyses of variance ($N=10$) showed that pretraining on deep-structure had a significant, positive effect ($P<.05$) when the graphic query language was used. It is suggested that the usefulness of explicit training in analysing queries in terms of logical deep-structure is dependent on the semantic constraints inherent in the formal query language. The textual query language, containing keywords and a syntactical template such as GET _ WHERE_ EQ, assumedly induced stronger cognitive constraints than the present graphic.

INTRODUCTION

The motive for the present study was the idea that understanding and mastering of complex tasks, such as formal query writing in database search, is facilitated by explicit instructions pertaining to the "deep structure" of the task. According to Craik and Lockhart (1972) the ability to retain linguistic information in memory is related to the depth of processing. They also claimed that depth of processing was related to familiarity with the linguistic material (the more familiar the material, the deeper the level of processing). Linde and

Bergström (1988; 1990) showed that there was a relationship between ease of learning the basic search principles in a relational database and prior knowledge about the informational content and the organization. One may interpret those results in terms of a levels of processing framework. That is, prior knowledge of the informational content and organization might have had an effect on the cognitive processing of the queries. For example, this knowledge may have made the subjects more inclined to create mental models on the basis of the logical content of the queries. An argument for a distinction between the surface structure of a program, and its deep structure is made by Rist (1989). According to him the surface structure consists of a template of the program code (e.g. input and output functions), whereas the deep-structure consists of plans and goals of the problem or task. Rist showed that novice programmers' creation of plans first showed a bottom-up development (from goal to subgoals and relevant program code). Then once the programmers had learned complete plans, their plan creation showed top-down development and was implemented in the order, in which plan pieces occurred in the finished programs. The study of Rist indicates that comprehension of programming tasks at a "deep-structure" level is essential even at a low degree of expertice. The Rist study addresses the question of how task-relevant knowledge is accessed and used in programming tasks. A different question is how programmers with varying level of expertice represent their knowledge. The latter question has been treated by Adelson (1981). She used a recall procedure (of randomly presented program lines) to infer that novices tended to use a syntax-based organization, whereas experts used a hierarchical organization based on the principles of program function.

The purpose of the present study is to investigate what effect training in using a botttom-up (from goals to subgoals) strategy has on novices'learning performance in a database search task.The concept "deep-structure" is used for the following basic linguistic constituents of queries to relational databases; information to be retrieved and condition(s). A distinction can be drawn between syntax (surface-structure) and semantic (deep-structure) level of queries formulated in a query language. Another way to draw the distinction between deep- and surface-structure is that the deep-structure of a given query is invariant over different query languages, whereas the surface-structure may vary. From the user's point of view the syntax level may seem based on arbitrary rules for combining the different constituents, whereas the semantic level corresponds to non-arbitrary

characteristics of search in a relational database. It is hypothesized that explicit training in analysing queries in terms of deep-structure in the above sense will facilitate the acquisition of subsequently given instructions about formal query language rules. Moreover, the transfer effect of such training on a subsequent, related task is investigated. The subsequent task constituted the writing of "procedures" in a special programming language. The procedures should be used to retrieve information from a database instead of the formal query language.

METHOD

Subjects

Twenty subjects volunteered to take part in the experiment. They were all recruited from a three-year, undergraduate education in behavioral sciences. The subjects were paid 250 SKR for their participation. Twelve subjects were female and eight were male. None of the subjects had any basic training in computer science or in computer languages such as PASCAL or BASIC.

Design

The subjects were randomly assigned to one of four experimental conditions according to a two-factorial design(see Figure 1). One factor was pretraining in analysing textually represented queries in term of information to be retreived and logical conditions vs. no such training. The other factor comprised type of query language, in which training was given. One was a formal, textual query language (see appendix). The other was a "graphic" query language, in which the subject indicated information to be retrieved and logical conditions by pointing at a graphic overview of the data base content and query constituents (See Fig 2).

The subject's tasks

A relational database system called MINER was used under all experimental conditions. The first part of the experiment comprised the learning of one of the two query langauges mentioned above. The second part comprised the learning of a simple special-purpose programming language in MINER. The second part of the experiment was equal to all subjects.

The first part contained 10 search tasks (queries formulated in Swedish) including one or two conditions combined by AND or OR, the

relational operators EQUAL or LESS THAN and queries, in which data from one or two columns had to be retrieved. The subjects performed the search tasks interactively at a VDU terminal.

The second part comprised four procedure construction tasks (see appendix): a/ Writing a procedure, by which a user could execute a specific, formal query repeatedly by one EXECution command (one line program code). b/ Writing a procedure, by which a user could execute a query having a constant information to be retrieved constituent and a variable condition (two lines program code). c/ Writing a procedure, by which a user could execute a query having both a variable information to be retrieved constitent and a variable condition (three lines program code). d/ Constructing a procedure, which allowed a user to choose between one of the three procedures mentioned above from a menu (three lines program code).

The subjects were instructed to first write sketches of each procedure on paper, then enter each procedure to the system, test and (if necessary) correct it interactively using a line-oriented editor (in the MIMER system).

Manual and oral instructions

Pretraining in analysing textually represented search tasks in terms of basic query components was given in the form of a/ a table containg three columns and six rows and b/ four queries pertaining to information in the table. The subjects were asked to analyse each query in terms of information to be retrieved and information constituting logical condition(s).

Subjects presented with the textual query language were given a manual containing information about basic concepts (definitions), keywords, relational and logical operators, syntactical rules and query templates. Subjects presented with the graphic query language were given a manual containing e.g. information about how to point at the graphic overview presented in Figure 2 and about how to use the keyboard. Instructions pertaining to relational and logical operators were the same under both the textual and graphic query language conditions. Both manuals were supposed to contain necessary information, but no more, for accomplishing the tasks.

All subjects were also given an overview of the dataorganization, as shown in Figure 3. A 1-page description of a scenario, pertaining

to the contents of the tables was also given to all subjects.

The manual for the second part of the experiment comprised a description of different modes (edit mode etc.) in the MIMER system, of some relevant edit commands, instructions about three types of statements in the special-purpose programming language and some general information about what purposes procedures might be serving. The three types of statements were a/ defining and assigning values to variables b/ execution of queries containing variables and c/ constructing a menu option, by which a user could gain access to a certain procedure.

The subjects' were also given oral instructions by the experimenter. These instructions included a summary of the contents of the manual and a demonstration of its organization (given to all subjects). The experimenter also gave assistance when a subject was found to be stuck during the performance of a task either by referring to a specific section of the manual or by informing the subject about the nature of the error.

Experimental procedure

Each subject took part in the experiment individually on two occasions (separated by at least one day and at most one week). Every subject work four hours on each occasion. The first four hours comprised training in formulating specific queries (interactively) to a relational database using one of the two types of query languages. The next four hours comprised training in writing "procedures" using a special programming language (see section The subject's tasks).

Each 4-hour session was ran as follows: First, the experimenter gave an oral presentation of the manual content and its organization. Then the subject could read the manual silently as long as (s)he wanted. After that the experimenter posed questions about essentials of the manual (such as definitions of the technical concepts). The presentation and reading of the manual, and questioning took between one and two hours. Then the subjects performed as many of the training tasks (10 query construction or 4 procedure construction tasks) as they could manage interactively at a VDU using the manual. They were allowed to ask the experimenter for assistance when necessary. The experimenter could also provide verbal assistance, when the subject appeared to be "stuck". The query construction tasks were presented in the same order to all subjects and one task had to

be successfully completed before proceeding to the next. As for the procedure construction task the criterion of a successful solution was that the procedure should be in working order. The subjects were free to enter the program code of a new procedure before testing a previous one.

Collection and analyses of data

Log-files of transactions (such as either attempts to construct queries, or entering or editing program statements in the "procedures") were collected. Parts of the verbal communication between the experimenter and subject were tape-recorded and typed, all of which were task-relevant dialogs between experimenter and subject.

All errors (with the exclusion of typing and spelling errors) were noted by one of the authors using 14 categories for query construction and 25 categories for procedure construction (see appendix) on the basis of log-files and verbal protocols. "Assistances" given by the experimenter and "self-corrections" of errors (errors spontaneously corrected by the subject without assistance) during query construction were identified on the basis of the verbal protocols by the same author.

The cause of every error during query construction was scored by two independent judges (the authors) using six categories: 1/ Misunderstanding of oral and written instructions. (The criterion was that the error pertained to information given in immediate connection with its appearance.) 2/ Lack of attention. (The criterion was that the error was corrected by the subject spontaneously, without interference from the experimenter, or that an error appeared in a previous correct part of a query) 3/ Disregard of syntactic constraints. (The criterion was that the subject seemed unaware of the necessity to strictly adhere to the formal rules of the query language.) 4/ Inability to analyse query properly. (The criterion was that the subject seemed unable to understand the deep-structure of the query.) 5/ Disregard of data organization. (The criterion was that the subject used the wrong table, or associated a table label with a column label, which did not exist in that table.) 6/ others (retrieval of redundant information or inefficient but correct solution of task).

RESULTS

Performance on the query construction task (Day 1)

Table 1 shows the average number of errors (with the exclusion of errors that were spontaneously corrected by the subject; "self-corrections"), assistances and errors +assistances/2 under the four experimental conditions. Each number is corrected for the number of completed tasks according to the formula $N_{total}/N_{completed} * X_i$ where N total is 10, $N_{completed}$ the number of completed tasks of each subject and X_i number of errors, assistances or errors+ assistances/2, respectively, of each subject. The left column shows the average "efficiency", i .e. number of completed search tasks, under different experimental conditions. As may be noted the differences between experimental conditions with respect to efficiency are small. As to both number of errors and assistances given, the condition **graphic query language/ no training on analysing deep- structure** exhibits the lowest performance and the condition **graphic query language/ training in analysing deep-structure** the highest performance. It can also be seen that the pretraining on deep structure factor seems to have opposite effects in combination with the two query languages.

Table 2 shows total numbers as well as proportions of errors scored in terms of different causal categories. Initial categorizations are those, which both judges agree on from independent scoring procedures and final are those, which the judges agreed upon after a mutual decision. The total number of initial categorizations is 113 and the total number of final categorizations is 192. Accordingly, an interrater-agreement coefficient was computed as 113/192=.59. The separate interrater-agreement coeffecients for each experimental conditions were .65 (TQL/PADS), .51(TQL/no-PADS), .83(GQL/PADS) and .51(GQL/no -PADS), respectively. The figures indicate that it may have been more difficult to classify the errors under the **no training on analysing deep-structure** conditions. Table 2 indicates that the training on deep-structure factor may have an effect on the distribution of types of errors. Training on analysing deep-structure tends to give relatively more errors attributable to lack of attention and the no such training conditions relatively more errors attributable to disregard of syntactic constraints and inability to understand a query properly.

Performance on the procedure construction task (Day 2)

Table 3 gives the number of errors committed and assistances given

during the procedure construction task. All subjects had been instructed to write the program code for each procedure on paper before entering it to the system. The table gives errors and assistances committed during the two phases separately. All numbers are corrected for number of completed tasks according to the same formula as for the performance on Day 1. The largest number of errors during the "writing on paper" phase is committed by subjects under the condition **graphic query language/ no training on analysing deep-structure**, the smallest number by subjects under the condition **textual query language/ no training on analysing deep-structure** condition. As to the "entering code on terminal" phase the largest number of errors is committed by subjects under the **graphic query language/ training on analysing deep-structure** condition.

Table 4 shows the number and proportions of errors of different types. The categories are based on 25 categories shown in the appendix and include 14 of those. It is noteworthy that subjects who had been trained in the graphic query language on Day 1 commited slightly more errors related to mismatch between transaction and system mode compared to subjects trained on the textual query language.

Statistical analyses

Table 5 presents two-factorial, univariate analyses of variance using different performance measures as the dependent variable. The independent variables are training vs. no training on analysing deep-structure and textual vs. graphic query language on Day 1. It can be seen that the overall effects of both independent variables are non-significant, but that there is a significant interaction between the training and query language factors on Day 1 for all three dependent variables. It is also noteworthy that the interaction between query language and training approaches a significant level on Day 2 with respect to errors while writing program code on paper ($p=.053$).

To further elucidate the effects of such pretraining in deep-structure, and the significant interaction between pretraining and query language, one-factorial analyses of variance were made separately for each query language. Training in deep-structure had a non-significant effect on errors+assistances/2 ($p=.424$) when the query language was textual, but yielded a significant effect ($p=.015$)

when the graphic query language was used.

Table 6 shows correlations between different dependent variables, computed on the error component of the variance (after extraction of variances attributable to query language effect or training effect). One may observe that the correlation between Day 1 and Day 2 in the overall performance variable "errors+assistances/2", is only +.30. One may also note that the correlation between no. of errors and no of. assistance on Day 1 is +.89, whereas the correlations between no. of error and no. of assistances on Day 2 are +.40 (paper and terminal).

DISCUSSION

The motive for this study was the idea that one important obstacle in database search tasks is that the user is unable to analyse the deep-structure of the queries to be constructed in a specific query language. In the present study the subjects were given queries in written Swedish. Their task on Day 1 was to translate them into one of two different query languages. In practice the user's task may be less well-defined than translating a query presented in written language into a formal query language. It is more likely that an information-seeking task is a part of a larger task (e.g. writing a memo, decision-making) and that the formulation of the information-need in written language may be an intermediate step before constructing a query in a formal query language. Studies related to the learning of database query languages, as a rule, have adressed the problem of translating queries in written language into a formal query language (e.g. Reisner, 1977; 1981; Katzeff, 1989). An obvious reason for this are the many factors inherent in a naturalistic learning situation, which could make a controlled investigation not feasable.

The most pertinent result of the present study is that the effect of training in analysing queries in terms of deep-structure was found to be dependent on the type of query language being learnt. The deep-structure of a query refers to properties, which are invariant over different query languages, i.e. independent of the formal aspects of a certain query language. Subjects given the graphic query language were found to benefit from training related to "deep-structure" in constrast to subjects given the textual query language. What made the training more efficient with the graphic query language than with the textual? We suggest that the usefulness of giving

training in analysing queries, as in this study, may be related to constraints with respect to underlying structure of the query, imposed by the semantics and syntax of the query language. The textual query language contained keywords such as GET___WHERE___EQUAL , which might have enforced the subjects to analyse the queries in terms of a "information-to-be-retreived and condition" scheme. Thus, the meaning that had to be given to GET, WHERE etc. in this specific query construction task was assumedly congruent with the meaning of these words in natural language. The graphic query language (see Fig. 2) , on the other hand, was based on the principle that the user should point at different columns in a graphic overview of the dataorganization, then put these columns at specific portions of a graphic figure. (A special pencil was used to perform these transactions.) The syntactical rules (rules about where information to be retreived and conditions should be put) of this query language may have appeared abritrary for subjects having had no training in analysing queries. Table 2 indicates that these subjects made more errors found to be related to inability to analyse the query or disregard of syntactic constraints than other subjects. One may also observe from Table 3 that the interaction between query language and training effects to some extent seems to remain (P=.053 for no. of errors) while the subjects wrote program code on paper on Day 2. However, it is completely absent while they entered program code on terminal. One may observe from this study as well as from two earlier ones on database search (Linde and Bergström, 1988; 1990) that the impact of task-relevant prior knowledge reveals itself at the onset of a more complex task and soon diminishes. Adelson and Soloway (1985) investigated the role of domain experience in software design. Programmers were given three different programming tasks; interrupt handler (domain familiar/objects familiar), electronic mail system (domain familiar/objects unfamiliar) and library system (domain unfamiliar/ objects unfamiliar). The results indicated that when programmers lacked domain knowledge, they tended to rely on effortful mental simulations and note-making. These data seem to bear a relationship to results obtained by Linde and Bergström (1988). In the present study the experience with the query construction task may have fulfilled a similar function for the performance in the procedure construction task as did domain experience in the Adelson and Soloway study.

REFERENCES

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

Adelson B. and Soloway E. (1985) The role of domain experience in software design, IEEE transactions on software engineering vol. SE-11,1351-1360.

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.

Katzeff C. (1989) Cognitive aspects of human computer interaction: mental models in database query writing, Doctorial dissertion, Akademitryck, Stockholm.

Linde L. and Bergström M. (1988) Impact of prior knowledge of informational content and organization on learning search principles in a database, Contemporary Educational Psychology, 13, 90-101.

Linde L. and Bergström M. (1990) An experimental study on mental models in database search, in D. Ackermann and M.J. Tauber (eds.) Mental models and human computer interaction 1, Elsevier Science Publishers.

Reisner P. (1977) Use of psychological experimentation as an aid to development of a query language , IEEE transactions on sofware engineering SE-3, 218-229.

Reisner P. (1981) Human factors aspects of database query languages, Computing surveys, vol 13.

Rist R.S. (1989) Schema creation in Programming, Cognitive science, 13, 389-414.

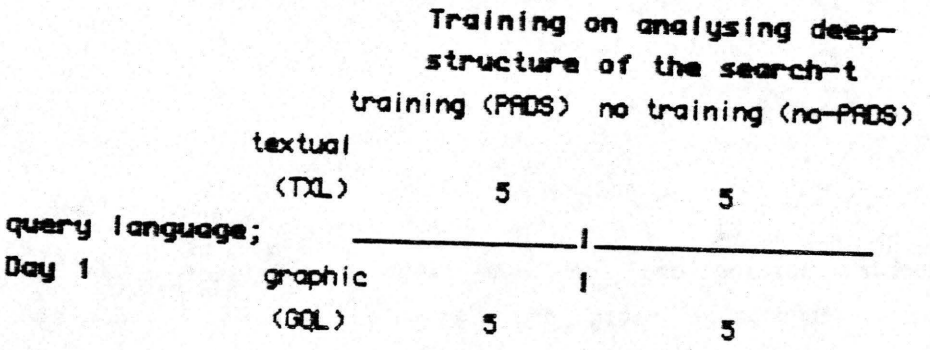


Figure 1: Design of the experiment.

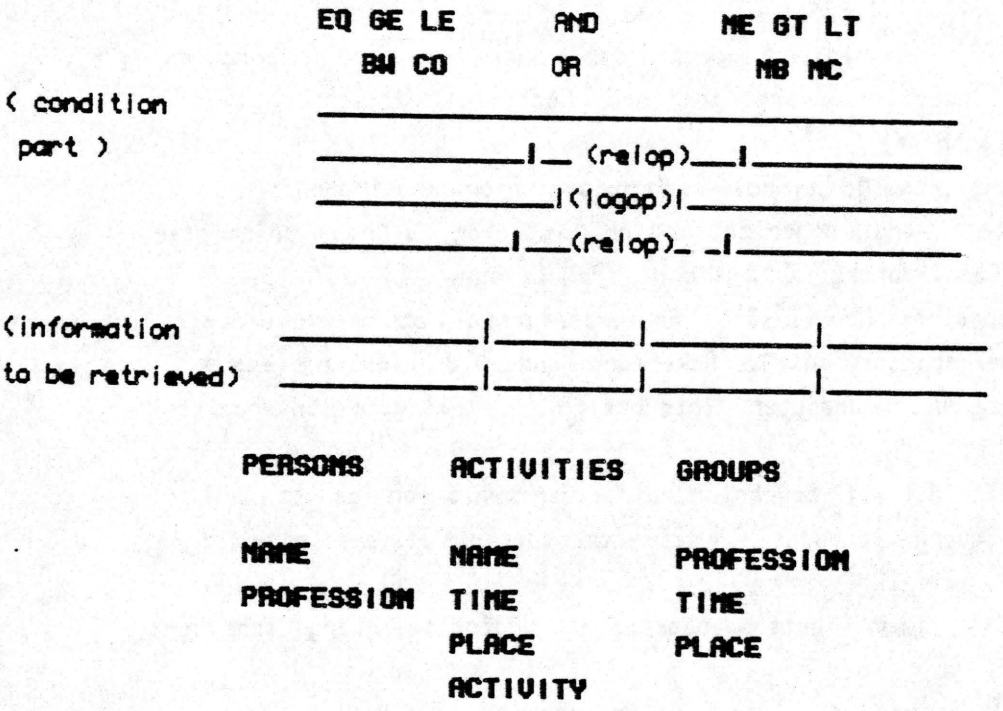


Figure 2: Overview of the graphic query language. Information within brackets was not presented to the subjects.

ACTIVITIES

TIME	NAME	PLACE	ACTIVITY
745	Jim	x-town	road-blocking

(92 rows)

PERSONS

NAME	PROFESSION
Jim	policeman

(25 rows)

GROUPS

PROFESSION	PLACE	TIME
policemen	x-town	745

(4 rows)

Figure 3: Overview of the dataorganization, as presented to the subjects.

TABLE 1: Day 1 (Query construction) Means and standarddeviations for no. of errors (not self-corrected), "assistances" and errors+"assistances"/2. All figures are corrected for number of solved tasks (see text) PADS= Pretraining in Analysing Deep-Structure of queries; TQL=Textual Query Language; GQL= Graphic Query Language. (Efficiency= no. of completed tasks)

			Errors		"Assistances"		Errors+Assistances/2	
		Efficiency	M	S	M	S	M	S
TQL	PADS	7.4	**10.6**	10.2	**11.7**	12.5	**11.2**	11.3
	no-PADS	8.0	**6.7**	5.2	**6.1**	6.3	**6.4**	5.7
GQL	PADS	6.8	**4.6**	2.3	**6.3**	4.6	**5.2**	2.9
	no-PADS	6.8	**19.1**	10.4	**23.5**	13.6	**21.3**	11.3

TABLE 2: The number and proportions of errors of different types (see text!) under each experimental condition; 1= misunderstanding of oral and written instructions; 2=lack of attention; 3=disregard of syntactic constraints; 4= inability to understand query properly; 5= disregard of dataorganization; 6=others. Initial.= Classifications, for which the judges independently had reached a common decision. Final= Classifications agreed upon after a unanimous decision.

category	TQL				GQL			
	PADS		no-PADS		PADS		no-PADS	
	initial	final	initial	final	initial	final	initial	final
1	2(.06)	2(.04)	2(.10)	2(.05)	0	0	2(.05)	4(.05)
2	14(.41)	23(.44)	3(.15)	13(.33)	8(.40)	12(.50)	9(.23)	10(.13)
3	7(.21)	10(.19)	5(.25)	9(.23)	5(.25)	5(.21)	11(.28)	26(.34)
4	8(.24)	11(.21)	6(.30)	9(.23)	4(.20)	4(.17)	12(.31)	27(.35)
5	2(.06)	3(.06)	3(.15)	4(.10)	3(.15)	3(.13	2(.05)	4(.05)
6	1(.03)	3(.06)	1(.05)	2(.05)	0	0	3(.08)	6(.07)

TABLE 3: Day 2 (Procedure construction): Means and standarddeviations for no. of errors, "assistances and errors+assistances/2 for each experimental conditions. All figures are corrected for no. of solved tasks (see text):

	Writing program code on paper				*Entering program code on terminal*				*Total no. errors+ assistances/2*	
	errors		asssistances		errors		assistances			
	M	S	M	S	M	S	M	S	M	S
TQL PADS	6.4	3.6	7.4	8.0	12.0	5.7	12.0	7.0	19.0	7.9
TQL no-PADS	3.6	2.3	6.0	3.7	12.6	5.5	10.3	6.6	16.3	5.6
GQL PADS	6.1	3.3	7.1	5.7	15.9	9.6	7.7	6.2	18.4	11.4
GQL no-PADS	8.6	1.6	8.6	3.5	12.8	5.9	9.6	4.1	19.5	4.2

TABLE 4: Numbers and proportions of errors of different types during day 2 (procedure construction). The numbers are corrected for number of solved tasks. (see text)

Type of error	TQL		GQL	
	PADS	no-PADS	PADS	no-PADS
1.Variable definition	1.1(.02)	5(.10)	0	5(.08)
2. Formulation of prompts and other guiding text for the user	23.9(.49)	16.4(.32)	20.5(.40)	19(.31)
3. Mismatch between transaction and system mode	13.7(.28)	20.7(.41)	23.9(.47)	26.2(.43)
4. Improper use of editor	10.3(.21)	8.7(.17)	6.8(.13)	10.7(.18)

TABLE 5: Two-way analyses of variance of the different performance variables.

performance variable		F	DF	P value
DAY 1: Errors+Assistances/2				
	E. Query language	1.34	1	.265
	E. Pretraining	2.18	1	.159
	Interaction Q X P	7.38	1	.015 *
DAY 1. Errors				
	E. Query language	.84	1	.373
	E. Pretraining	2.28	1	.150
	Interaction Q X P	6.91	1	.018 *
DAY 1: Assistances				
	E. Query language	1.80	1	.198
	E. Pretraining	1.67	1	.214
	Interaction Q X P	6.52	1	.021*
*				
DAY 2: Errors+Assistances/2				
	E. Query language	.14	1	.716
	E. Pretraining	.06	1	.815
	Interaction Q X P	.29	1	.599
DAY 2: Errors/Paper				
	E. Query language	3.43	1	.083
	E. Pretraining	.01	1	.907
	Interaction Q X P	4.36	1	.053
DAY 2: Assistances/Paper				
	E. Query language	.22	1	.648
	E. Pretraining	.00	1	.984
	Interaction Q X P	.34	1	.566
DAY 2: Errors/Terminal				
	E. Query language	.45	1	.511
	E. Pretraining	.15	1	.704
	Interaction Q X P	.37	1	.552
DAY 2: Assistances/Terminal				
	E. Query language	.87	1	.366
	E. Pretraining	.00	1	.957
	Interaction Q X P	.44	1	.515

TABLE 6: Correlations (based on the variance within the error component) between different performance variables.

	1	2	3	4	5	6	7
1. Errors, Day 1							
2. Assistances, Day 1	.89						
3. E+A/2, Day 1	—	—					
4. E+A/2, Day 2	—	—	.30				
5. E, paper, Day 2	.31	.31	—	—			
6. E, terminal, Day 2	.02	.01	—	—	.23		
7. A, paper, Day 2	.45	.42	—	—	.40	—	
8. A, terminal, Day 2	.29	.34	—	—	.60	.40	.41

APPENDIX

Example of a query given to the subjects at Day 1:
In what activity was Anthony involved at 230 p.m.?

Formal query expression:
GET EVENTS.ACTIVITY WHE EVENTS.NAME EQ ANTHONY AND EVENTS.TIME
EQ 230 P.M;

Tasks given to the subjects at Day 2:

1. Write a procedure, which always answers the query " What is Anthony's profession?"

Program code:
GET PERSONS.PROFESSION WHE PERSONS.NAME EQ ANTHONY;

2. Write a procedure, by which a user can pose a query about any person's profession.

Program code:
+LET &A=' NAME OF THE PERSON:''
GET PERSONS.PROFESSION WHE PERSONS.NAME EQ &A;

VDU information (from the user's perspective):
NAME OF PERSON:

3. Write a procedure, by which a user can pose a query about any column value for any optional person in the table EVENTS.

Program code:
+LET &A=' NAME OF THE PERSON:''
+LET &B=' OPTIONAL COLUMN:''
GET PERSONS.&B WHE PERSONS.NAME EQ &A;

VDU information
NAME OF PERSON:
OPTIONAL COLUMN:

4. Insert the procedures in a menu.

Program code of the menu:

```
+CASE PROC1 'THE PROFESSION OF ANTHONY';
+CASE PROC2 'THE PROFESSION OF AN OPTIONAL PERSON';
+CASE PROC3 'OPTIONAL COLUMN OF AN OPTIONAL PERSON';
```

VDU information

```
1 THE PROFESSION OF ANTHONY
2 THE PROFESSION OF AN OPTIONAL PERSON
3 OPTIONAL COLUMN OF AN OPTIONAL PERSON
```

Classification of errors; Query construction

1. DEL: missing or incorrect delimiter 2. KOL: incorrect or missing column label 3. LOG: incorrect logical operator 4. KEY: incorrect or missing keyword 5. INT: unnecessary interruption of a query 6. PAR: missing or incorrect parameter 7. REL: missing or incorrect relational operator 8. MIS: conditional part or information-to-be-retrieved part completely missing 9. SYN: multiple syntactical error 10: TAB: missing or incorrect table label 11. CON: one, of two conditions, missing or incorrect 12. VAL: missing or incorrect value of the condition 13. IRC: irrelevant condition 14. IRP: irrelevant parameter

Classification of errors; Procedure construction

ERRORS WHILE WRITING PROCEDURE CODE ON PAPER

1. Syntax error due to improper use of query language 2. incorrect definition of variable in variable definition line 3. incorrect definition of variable in query line 4. incorrect or misleading prompt 5. Others

ERRORS WHILE ENTERING PROCEDURE CODE TO THE SYSTEM

6. incorrect labeling of the procedure 7. program code of a "new" procedure entered under the label of a previous procedure 8. several, distinct lines of program code entered to the system as one line 9. failure to enter proper mode 10. mismatch between initiated transaction and mode state 11. incorrect use of (line-oriented) editor 12. other, formal syntax errors

ERRORS WHILE CONSTRUCTING MENU

13. formal syntax error in +CASE line 14. relevant procedure label omitted in +CASE line 15. failure to understand, why each procedure must be linked to a special +CASE line 16. misleading or incorrect guiding menu text at a certain option 17. mode error 18. others

ERRORS WHILE TESTING PROCEDURES

19. failure to understand one's own prompt text 20. failure to understand one's own menu text 21. inability to correct prompt text in a way that makes it comprehensible for the potential use 22. inability to correct menu text in a way that it comprehensible for the potential user 23. mode error 24. incorrect use of editor 25 others

A Framework for exploring Exploration

J. M. Slack & C. Conati
Istituto per la Ricerca Scientifica e Tecnologica
38050 Povo (TN), ITALY
e-mail: slack@irst.it

1. Introduction

When a user engages an information retrieval (IR) system for the first time the resulting interaction is determined by a number of factors, including (a) the user's goals in engaging the system; (b) her familiarity with the interface; and (c) her knowledge of the information domain. The initial interactions might be termed exploratory, but this description is ambiguous with respect to these three factors. First, the user's interactions might be exploratory in the sense that they are not directed toward the attainment of a specific goal, rather the user entertains a non-specific goal such as 'to learn something about ...'. Second, the user might be said to be exploring the interface meaning that she is familiarising herself with its capabilities. Finally, the user might be exploring the information domain for the first time. With minimal knowledge of the domain, the user's goal might be to gain an understanding of the domain's content and organisation. It is this latter interpretation of exploration that the paper focuses on; a situation that has been referred to as the *exploration of an information space* (Egido & Patterson, 1988).

The exact situation we want to investigate is that in which a user engages an IR system either to access information necessary to attain a particular goal, or just to explore the information domain. In both cases, however, the domain is relatively unfamiliar to her. To minimise the influence of the interface and the user's level of experience with it, we have focused on an application domain which employs a natural language interface.

Application domain: *AlFresco* is a multimodal dialogue prototype system for the exploration of art history (see Carenini et al.,1990). Specifically, this system allows the user to retrieve information about Fourteenth Century italian painters and frescoes. The AlFresco domain is encoded in a knowledge representation module called Krapfen (derived from the KL-One representation language (Franconi, 1990)). Krapfen comprises a structural network containing frames for the the definition of the general concepts (terms) involved in the domain, such as *Painter, Fresco, Monument.* The frame describing the concept of *Painter*, for example, has slots containing the period of the painter, the cities where he worked, his works, the school he belonged to, and so on. AlFresco is connected to a touchscreen and videodisc storing images of frescoes. A user can interact

with the system (asking to see a fresco or for additional information) both through typing his requests on a keyboard or by referring to particular details in a fresco by touching them on the touchscreen (Samek & Strapparava, 1990).

The paper describes a processing framework in which the cognitive forces that drive exploratory behaviour in this type of situation can be studied. In particular, we consider the psychological construct of *interestingness* as a major factor underlying the user's interactions. The next section outlines the major cognitive approaches to the notion of interestingness, illustrating the full complexity of the construct. The present goal of the study is to specify a conceptual framework and corresponding simulation environment in which to investigate the construct of interestingness and the user's exploratory behaviour.

2 Exploration and Informational Significance

The construct of interestingness as a cognitive factor has emerged in research on text processing. For example, Schank (1979) has identified certain interest-eliciting conditions associated with the construction of texts: A text elicits interest in the reader when (a) structural expectations are violated or deviated from; (b) structurally-relevant information is missing, or (c) the content of the text refers to certain universally salient themes such as death, and personal relations, which are interesting regardless of their context. While these ideas have proved influential in the research domain of story processing (Hidi & Baird, 1986) they are more difficult to apply in accounting for the exploration of a new knowledge base. In this situation, the user has only a minimal knowledge of the new domain and thus generates few structural expectations and has only a superficial notion of which information is structurally-relevant to the domain. However, the notion of universally interesting themes is directly applicable to the explanation of the user's exploratory behaviour.

Another aspect of Schank's analysis which restricts its applicability is that it does not specify the processing aspects of interest. Interestingness is viewed as a structural or thematic phenomena with little emphasis on interest as a process. To understand how interest drives a user's exploration of an information space it is necessary to specify the response elicited by the structural and thematic conditions. In this respect, Hidi and Baird (1986) have argued that what is central to the response of interest is that a person is compelled to increase her intellectual activity to cope with the greater significance of the incoming information. The crucial notion which these researchers introduce is that of *informational significance* which embraces all forms of interest-eliciting event. The significance of an

item of information has its origins not only in the structural relation of the stimulus to the user's related knowledge base, but also in the user's goals, wishes and values. van Dijk & Kintsch (1983) have argued that in discourse processing, opinions, attitudes, values and emotions are activated in addition to knowledge-activation. From this perspective, interest is generated through the interaction of the stimulus with the user, making it impossible to identify the origin of the interest as one source to the exclusion of the other. However, underlying the notion of interest there is a dimension of relative significance varying from the universal to the completely subjective and personal. In common with the findings of attribution theory, universal interest tends to be attributed to the stimulus, and variable interests to the person.

In exploring an information space, the user generally has to cope with multiple sources of informational significance which compete for his processing resources. In such a situation, the most important information for the user does not always have the highest informational significance. There is little research directly on this topic, but in the related research area of the processing of expositions the studies tend to show that interest and importance are not closely related and that both factors tend to affect what is remembered (Taylor, 1980; Winograd, 1984; Baird & Hidi, 1984). The general conclusion from this work is that more research needs to be done to specify the manner in which an affective component like interestingness can increase intellectual activity and facilitate the recall and comprehension of important material.

In common with Hidi and Baird (1986), we are viewing interest as the *process* of responding to the *significance* of information. As a process it has triggering conditions, it has a duration, and there are conditions which ensure its continuation. Thus the study of interest requires research into the variety of ways in which the content of the information base has *significance* to the user. Fundamentally, this requires an understanding of the origins of interest beyond the user's knowledge system to his value system mediated through affective experience. However, such a study is beyond of the scope of the present work. What drives the user's exploratory behaviour is his response to the informational significance of the material delivered by the information retrieval system and this is determined through the interaction of the material's semantic content with the user's knowledge of the domain, goals, values and general preferences. In this respect the user's path through the information space is determined by the following principle:
Principle of maximal interestingness - In situations in which a user explores an information space, her route through the space is along the path of maximal interestingness.

3 Exploration and Reactive Planning

In the case where the user engages the system with a well-defined goal, or identifies such a goal in the course of the dialogue, the informational significance of a stimulus will be determined, in a large part, in relation to the informational requirements of the user's partial plans for attaining her goal. In pursuing a well-defined goal, a user may be blocked because she is lacking the necessary information and does not know how to access it directly and so the goal is suspended. But, when the requisite knowledge is encountered at some later point in the interaction, the goal becomes reactivated as the current goal. Thus, some goals may be latent, and others suspended while the current goal is pursued, even though the current goal might be non-specific. However, the interaction that determines the informational significance of retrieved information is sensitive to such goals. The stimulus information interacts with the goals in such a way that it can reactivate a suspended goal or plan. This type of goal-directed behaviour or planning has been referred to as *Opportunistic Planning* (Birnbaum & Collins, 1984), or *Reactive planning* (Georgeff & Lansky, 1987) and is typical of exploratory situations.

4 The need for an integrative framework

The full informational significance of a stimulus can only be determined through the integration of multiple sources of knowledge, including knowledge of the user's goals, values and preferences with respect to the information domain. The coordination of such diverse sources of information requires an integrative framework embracing a wide range of cognitive functions. One such framework is the Soar cognitive architecture developed by Newell et al. (Laird, Newell, & Rosenbloom, 1987) at Carnegie-Mellon University. The choice of the Soar architecture as the modeling environment is based on two complimentary views of the system, both of which make it a suitable integrative framework. First, Soar has been proposed as a uniform cognitive architecture (Newell, 1989), that is, as a uniform architectural basis accommodating the full range of human mental functioning. Second, at the software level, Soar can be regarded as a general modeling language for integrating multiple cognitive sources (Bonatti, 1989).

Soar specifies a cognitive architecture comprising a relatively fixed set of mechanisms that permit goals and knowledge of the task environment to be encoded in memory and brought to bear to produce behaviour. The architecture has been applied to human behaviour in a broad spectrum of domains. It is based on the notion that all goal-directed cognitive activity can be understood as search in a *problem space*. A problem space is defined as comprising a set of states, and a set of operators that can be applied to move from one state to another. Given a

task, encoded as a set of problem spaces, the Soar system uses knowledge to select problem spaces, states and operators to move towards a desired state. All of this knowledge is encoded as productions which fire to elaborate a context, comprising a goal, problem space, state and operator. When the knowledge directly available in a given context is insufficient to determine the next operator to apply, Soar sets up through subgoaling new contexts in which it attempts to obtain the necessary knowledge.

The structure of Soar is shown in figure 1, and its functional organisation is based on decision cycles. The decisions relate to the selection of problem spaces to work in, states to proceed from, and operators to apply. Several productions may fire in accumulating the knowledge in Working Memory to make a given decision. The final component, the *chunking mechanism*, learns from Soar's subgoaling experience and creates productions, called chunks, for use in future problem solving situations.

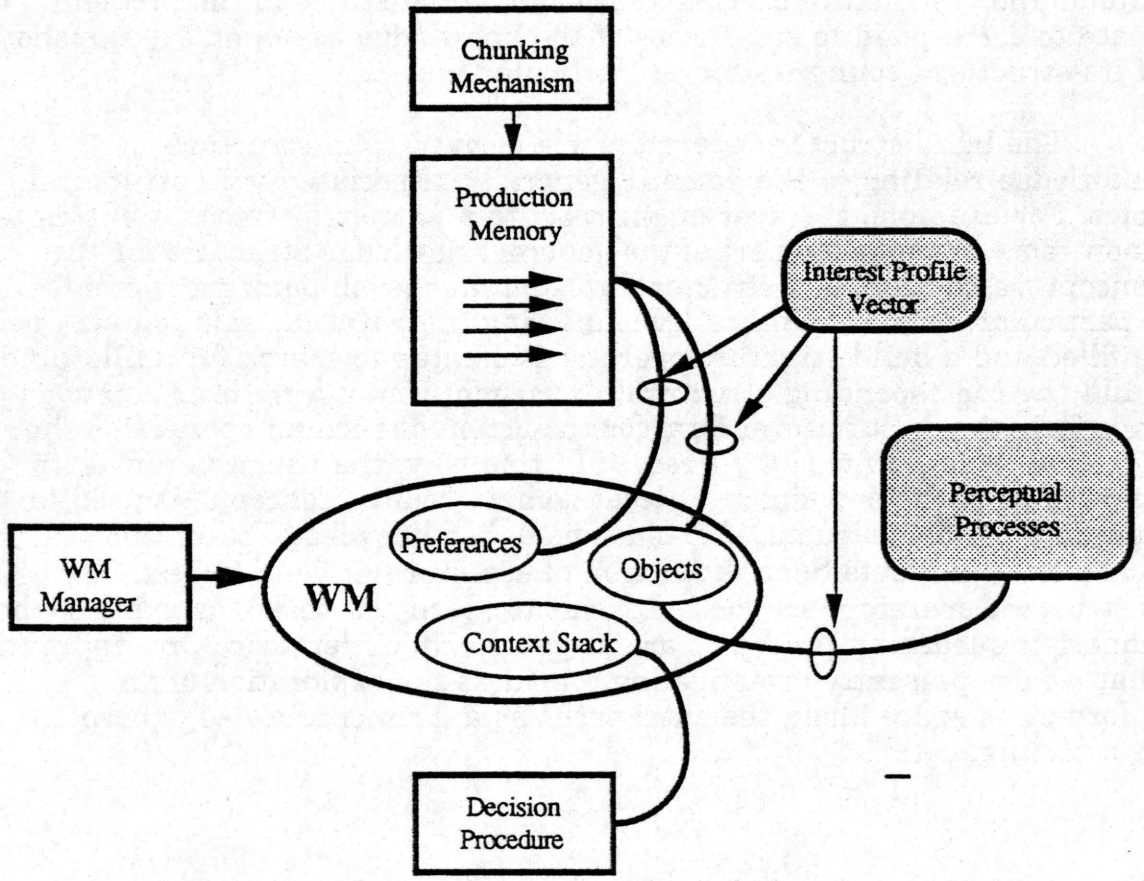


Figure 1 Soar-based framework

5 Exploration: A processing framework

In the processing framework shown figure 1, the shaded components represent functional elements additional to the standard Soar architecture. They comprise a set of *Perceptual Processes* which encode objects or description elaborations into Working Memory, and the *Interest Profile vector* which encodes the distribution of salience over the user's knowledge base[1].

Encoding the task: To encode the user's task in Soar it is necessary to specify the four components that comprise a context; the goal, a problem space, a state and an operator. One of the difficulties in modeling exploratory behaviour is that the overall goal is difficult to specify. However, viewing the user's task as navigating through an information space suggests the rather non-specific goal of extending and modifying her knowledge base. This allows the other context slot categories to be specified as follows: The top-level problem space comprises the user's knowledge base, encoded as a structural network, and the set of operators used to extend and restructure it. This allows individual states in the problem space to correspond to sub-spaces of the knowledge base and/or 'snapshots' of its structural configuration at particular instances.

The build-structure operators are derived from structural knowledge relating to the domain, generally, associated with structural roles. For example, the user might react to a particular fresco, wanting to know more about it. As part of the general knowledge structure for the concept *fresco* there is a structural role *painter* (each painting is created by a particular painter). On first encountering the painting, this role will be unfilled and a build-structure operator is created to search for a filler and build the corresponding structural extension. Using a natural language interface, the most natural form for the action of such an operator is the question '*Who painted this fresco?*'. In this way, the user's minimal knowledge of the domain, or at least domain-related concepts, is used to generate possible structural extensions to her knowledge base; this scheme is similar to Schank's notion of structure-driven interest. The restructure operators are more difficult to specify in detail, as many of the underlying learning processes are as yet poorly understood. One approach that we are presently investigating considers the exploration of an information space along thematic paths as a form of knowledge base restructing.

---

[1] Both components represent distinct functional elements, but at the implementation level they are encoded as normal Soar objects and productions.

Interest and attentional processes: Modeling the user of a relatively sophisticated information retrieval system requires an elaborated model of perception. Building perceptual processes for Soar is presently an active research area. In common with the tenor of the architecture, perceptual processes are modeled in terms of operators. Newell (1989) proposed three classes of perceptual operators; Attend, Encode and Comprehend operations. These classes of operations simulate attention mechanisms, the automatic processing of visual inputs, and the deliberate processing of inputs, respectively. The encoding operations function independently of the rest of the system making them insensitive to goals and interests. The other two classes of operation, on the other hand, are under the control of the architecture.

In the proposed scheme, the perceptual encoding processes create a structural description of a stimulus, such as a retrieved image.[2] The attentional processes prune this description creating a partial description of the input, under the control of the interest profile (IP) vector. This is achieved by merging the structural description with the IP vector and deleting those nodes and interconnections whose activation levels fall below a lower-bound threshold.

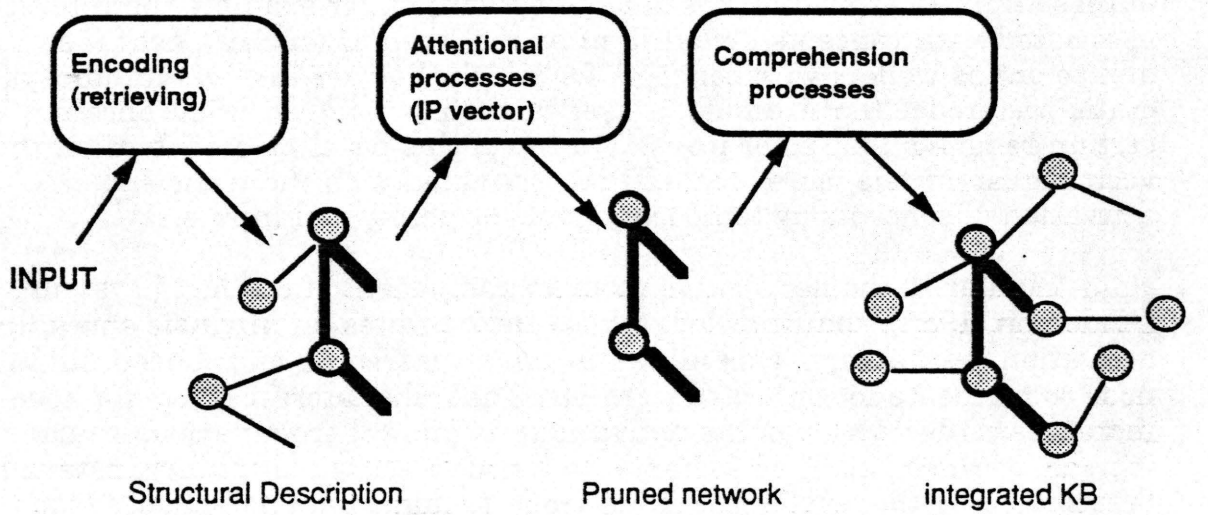


Figure 2 Perceptual processes

The comprehension processes integrate the pruned network with the existing knowledge base. Perceptual products not integrated are lost through decay. As yet, the comprehension processes have to be fully defined, but they need to perform at least three tasks, as follows,

[2] In the implementation, the structural descriptions of the video images are pre-encoded.

(i) they need to link the nodes of a perceptual sub-network to the corresponding type nodes in the knowledge base;
(ii) the identity and integrity of each perceptual sub-network must be maintained through the use of a label, or title, node for each picture;
(iii) additional cues, or descriptors, need to be attached as part of the integrated encoding to represent the affective, connotative and goal-oriented aspects of the IP vector involved in pruning the original perceptual description.
While the first two tasks are relatively simple to implement as Soar operators, the third task requires a more thorough understanding of the influence of non task-oriented factors on comprehension before an adequate simulation can be achieved.

The interest profile vector: The top-level problem space, comprises a network of labeled edges and nodes representing structural roles and fillers, respectively. An additional level of structure is specified by associating an activation level with each node and a strength measure with each edge. The distribution of activation over the network can be thought of as an *interest profile vector*. That is, the distribution reflects the relative interestingness of concepts over the knowledge base. The vector is a composite representation in the sense that it summates the individual effects derived from the manifold factors underlying interestingness. Some of these influences lead to a residual distribution, for example the interest associated with concepts denoting universal themes (such as death, sex, and so on) as well as with concepts for which the user exhibits an idiosyncratic preference (for example, a user might have a keen preference for certain religious images or icons). On top of this baseline distribution, the vector must encode more local effects associated with the transient activation of concepts by temporary goals or perceptual inputs.

Similar to the declarative memory component of the ACT* system (Anderson, 1983), the Knowledge Base incorporates an intrinsic spreading activation mechanism. This allows interest centred on one conceptual node to permeate to connected concepts. Thus, the interest level of a node increases or decreases as the activation reaching it from various sources increases or decreases. The change in a node's activation at any instant is determined by the level of the input from its immediate neighbours (the proportionality is given by the strength of the coupling factor between adjacent nodes), and through a decay process, by its current level of activation (that is, when not receiving input, the level decreases proportionately to its current level).[3]

---

[3] In the implementation, the equations and parameters governing the dynamics of activation are similar to those employed in ACT*.

Viewing spreading activation as an intrinsic component of the Knowledge Base allows us to divorce the interpretation of the activation distribution from the functional properties of the mechanism, permitting the model builder to place additional constraints on the vector. For example, viewed as an interest profile, the vector might be regarded as sparse, corresponding to the notion that local interest phenomena have a limited capacity. This is easily achieved by setting a relatively high threshold on each element, except those corresponding to residual interests (universal themes, etc.).

Forms of external activation: Nodes in the network receive activation from three sources external to the network. First, when the comprehension processes integrate a pruned network into the knowledge base, the constituent nodes act as a source of activation. This results in a form of feedback from the reduced perceptual encodings to the IP vector; the latter is used to prune the original encoding and the reduced network then reinforces the related IP elements. Second, the production rules encoding the various influences underlying interestingness specify actions which result in the relevant conceptual nodes being activated. Third, goals in the context stack activate the nodes in their associated sub-structures. The level of activation is determined by the level of the goal in the stack; the higher up the stack the goal is the higher the level of internal activation within a context.

Operator Selection: Situations often arise in which Soar is unable to make progress towards a goal because the directly available knowledge is inadequate. A common situation of this sort is an *operator-tie*, when several operators are proposed for application to a given state, but there is insufficient knowledge to choose between them. When this happens, the system reacts to the impasse by creating a subgoal to do the selection. In turn, this subgoal creates an evaluation subgoal which tries to evaluate each operator in order to generate preferences for them. In the problem solving domain, evaluation functions normally establish some form of metric over the set of possible states relative to a final goal state. However, such functions can only be defined given well-defined goals. In the case when no definitive goal state can be specified, one possibility is to use an evaluation function which is intrinsic to the problem space. In our framework, this can be achieved using the IP vector and basing the function on the maximisation of interestingness. That is, in this situation operator selection satisfies the principle of maximal interestingness.

To determine the path of maximal interestingness it is necessary to identify the build-structure operator with the highest activation value . The IP vector and spreading activation mechanism determine the distribution of activity over the entire knowledge structure, including the structural roles without fillers that are used to generate build-structure operators. Allowing the structural roles to communicate their activation levels to

their associated operators means that the evaluation subgoal merely has to search for the most activated operator.

Topic-oriented exploration: Given the dynamics of the framework it is not clear how it achieves topic-oriented exploration as opposed to jumping around the knowledge base in reaction to each new input. Two sources of stability can be identified, both of which result in IP vectors maintaining a dominant element over the course of processing. First, vector constraints can be used to bias the distribution toward one or a few elements; this is equivalent to a form of best-choice filtering. Second, external sources of activation can maintain a dominant topic, either through the feedback from perceptual inputs, or through the activation generated by the goal of the immediate context. Topic-oriented exploration becomes unsustainable once these contributing forces cease. When the pattern of mutual reinforcement breaks down, discontinuities appear in the exploration path and continue until another mutually consistent pattern emerges.

6 Conclusions

We have outlined a mechanism that implements the process of interest, without specifying the conditions which evoke that process. Such conditions need to be identified through empirical research. However, it should be possible to use the simulation environment to test various predictions regarding the influence of different types of information on the choice of path through the space. Such studies are crucial if we want to build systems that can effectively guide the user through new information domains, dependent on his overall goals and inclinations. For example, when users react to a particular feature of a retrieved stimulus due either to an idiosyncratic inclination or because of its compatability with some latent goal, the user dialogue will exhibit definite discontinuities, or sudden changes of path through the space. Initially, the simulation model uses such discontinuities to infer the structure of a user's interest profile, but once a user model has been learnt, it can begin to predict them and guide the user accordingly.

The inherent modularity of production system architectures allows us to continuously increment the modeling environment as we gain a better understanding of the affective and cognitive factors that influence a user's interests. However, certain predictions regarding the nature of exploration can be derived from the form of the system's functional structure. For example, the way in which the IP vector and perceptual processes interact suggests that notions such as *novelty* only become significant once the system can experience expectation failure. Such predictions are presently being investigated.

References

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

Baird, W., & Hidi, S. (1984) The effect of factual importance and interestingness on recall from naturally-occurring school texts. Paper presented at the annual meeting of American Educational Research Association, New Orleans, LA.

Birnbaum, L., & Collins, G. (1984) Opportunistic planning and Freudian slips. In the *Proceedings of the Sixth annual conference of the Cognitive Science Society*, Boulder, CO.

Bonati, P (1989) Evaluating the Soar architecture through a formal specification. IRST-Technical Report #8912-08, Trento, Italy.

Carenini, G., Ponzi M.,and Stock, O. (1990) Combining natural language and hypermedia as a new means for information access. In the *Proceedings of ECCE-5*, Urbino, Italy.

Egido, C., and Patterson, J. (1988) Pictures and category labels as navigational aids for catalog browsing. In the *Proceedings of CHI-88*, Washington D.C..

Franconi, E. (1990) The YAK manual: yet another KRAPFEN. IRST Technical Report #9003-01, Trento, Italy.

Georgeff, & Lansky, (1987) Reactive Planning. In the *Proceedings of AAAI-87*.

Hidi, S, & Baird, W. (1986) Interestingness - A neglected variable in discourse processing. *Cognitive Science*, **10**, 179-194.

Laird, J.E., Rosenbloom, P.S., & Newell, A. (1984) Towards chunking as a general learning mechanism. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, Texas.

Newell, A. (1989) *Unified Theories of Cognition*. Harvard University Press: Cambridge, Mass..

Samek-Lodovici, V., and Strapparava, C. (1990) Identifying Noun Phrase references: The Topic Module for the AlFresco System. In the *Proceedings of ECAI 90*, Stockholm, Sweden.

Schank, R. C. (1979) Interestingness: Controlling inferences. *Artificial Intelligence*, **12**, 273-297.

Taylor, R. (1980) Children's memory for expository text after reading. *Reading Research Quarterly*, **3**, 399-411.

van Dijk, T. & Kintsch, W. (1983) *Strategies of discourse comprehension*. New York: Academic Press.

Winograd, E. (1984) Strategic difficulties in summarizing texts. *Reading Research Quarterly*, **14**, 404-425.

Cognitive Studies in Software Engineering

Alistair Sutcliffe and Neil Maiden

Department of Business Systems Analysis,
City University,
Northampton Square,
LONDON, EC1V OHB, UK.
Phone: +44-71-253-4399 ext 3420,
E-Mail: sf328@uk.ac.city (JANET).

1 Introduction

Cognitive studies of software engineering have focused on the solution of well-structured problems, for example the construction of a simple program in a specified programming language (e.g. Adelson 1984). However software engineering involves two distinct phases; first acquisition and comprehension of a problem domain, and second creation of a software design representing a solution. Programming activity occurs during the second phase. The first phase, systems analysis, is more complex yet it is poorly understood. Some studies have identified a few determinants of good analytic performance (e.g. Vitalari and Dickson 1983) however no process model of the reasoning underlying analytic tasks has been developed.

Better planning and more critical testing of hypotheses have been suggested as qualities which differentiate expert from novice analysts (e.g. Guindon and Curtis 1988). Furthermore, experts appear to use better heuristics and retrieve richer knowledge structures from memory. Psychological studies of program designers have demonstrated that novices have fewer preformed mental schema which can be retrieved from memory and that novices tend to focus on the surface aspects of the problem (i.e. lexical/syntactic features of the programming language) rather than the semantic level of the problem itself (e.g. Adelson 1984, Koulek et al. 1989). In addition studies of program debugging have suggested novices fail to scope problems, resorting to a strategy of bug isolation and repair, whereas expert strategies are directed towards building multiple domain models. Systems analysis requires different and in many ways more demanding skills than programming, hence novice software engineers might be expected to have difficulty during an analytic task. A mental model of the analytic processes used by novice software engineers is necessary to inform the design of future CASE tools. This paper reports an investigation into an applied mental model of the analytic process for software engineering, based on protocol analyses of software engineering practices, followed by experimental studies on analogical reasoning during systems analysis.

Guindon and Curtis (1988) cite analogical reasoning and the reuse of domain knowledge as important determinants of expert analytic performance. Experts are able to retrieve previous knowledge stored in memory: however, novices do not have past experiences which are likely to help the analytic task. Software reuse is an important source of domain knowledge, which novice software engineers could exploit by analogical reasoning. Many theories of analogy have been proposed, for example Gentner (1983), however these theories are often incomplete and few task-related studies of analogical reasoning have been reported.

Analogous reasoning in problem solving is knowledge-intensive, hence problem solving by analogy is not suitable for complete automation. It is usually considered to have 3 steps, which mirror the steps involved in software reuse: (i) retrieval, (ii) transfer, and (iii) translation. In our scenario retrieval of reusable software is likely to be a tool-based task, however transfer and translation of analogous software requires the software engineer. Two empirical studies are reported, which investigated mental models of analogous reasoning in software reuse. One study investigated whether software reuse enhances analytic performance while the other study examined the mental processes of novice and expert software engineers during reuse, to enhance the previously described applied mental model of the analytic task. The problems presented to the subjects in the latter two studies were similar in scale to that provided in the first study, so that results from all 3 studies can be discussed in a similar context.

2 Study 1 - Applied mental model of the analytic process

An empirical study investigated cognitive processes of novice software engineers engaged in systems analysis. The objective was to describe the mental processes of systems analysis and create a model which had prescriptive value in that context, rather than a more general model of reasoning (e.g. GPS (Newell and Simon 1972) and ACT* (Anderson 1983)). Previous studies had investigated the behaviour of expert software engineers, hence this study investigated the analytic behaviour of novices, to contrast these previous studies and to provide an empirical basis for the diagnostic component of an intelligent CASE tool. A more detailed report of this study is given in Sutcliffe and Maiden (1990a).

2.1 Method

Protocol analysis was used to investigate systems analysis behaviour of 13 novice software engineers (11M, 2F students in MSc in Business Systems Analysis). Subjects were asked to develop a specification for a delivery scheduling system described in narrative form. All subjects had background domain and method knowledge necessary to develop a specification. Domain knowledge included all aspects of a delivery scheduling problem, obtained through experience on a case study. Method knowledge refers to procedural steps dictated by the Structured Systems Analysis (SSA) method, and its main representation technique, data flow diagrams (De Marco 1978). SSA is typical of techniques which currently exist to assist the software engineer during systems analysis.

Subjects were requested to think aloud and their verbal protocols were recorded on audio tape. Protocols were recorded following the practice of Ericsson and Simon (1984). Subjects were given 35 minutes to complete a solution, and were retrospectively questioned by the experimenter about the analytic behaviour for a further 10 minutes. Finally all subjects completed a questionnaire identifying details of all previous analysis and programming experience.

Protocol transcripts were analysed by matching mental behaviours to speech segments, usually sentences or incomplete utterances. Six major categories were used, divided into four mental and two non-mental behaviours:

Recognise goal -	statement of a higher level problem goal, often a system requirement;
Assertions -	verbalisation of a belief or statement of facts about the problem

domain, directly attributable to the problem narrative;

Reasoning - verbalisation of the creation, development and testing of hypotheses about the problem or its proposed solution;

Planning - meta-level control over the analytic process;

Information acquisition - searching for and retrieval of data from the problem text;

Conceptual modelling - physical construction of the system specification, recorded as a data flow diagram.

Reasoning was distinguished from Assertions by the degree of inference applied. Planning differed from Goal recognition in that it was a domain-independent statement of intention. Inter-observer agreement on protocol categorisation of 79% was achieved by two cross markers experienced in protocol analysis. Differences were then rationalised and the protocols were recategorised as necessary.

Completeness scores were allocated to each subject's solution specification, to represent their success in solving the problem. Completeness was measured by the inclusion or otherwise of each component in an expert's solution to the problem. A subject received a score for a component if it was included in their solution or verbalised as important to the required system.

2.2 Results

All subjects physically drew or verbalised a solution to the problem in the time available to them. 12 subjects developed data flow diagrams, all of which were incomplete. Subjects overall performance was poor, averaging only 11.4% of the expert score.

Frequencies of mental behaviours were counted for 5 minute segments. No marked trends over time were apparent, although Information acquisition, Assertions and Planning decreased with time and Reasoning increased in the later stages of the protocols. Inter-individual differences were apparent in total occurrences of each particular behaviour.

No correlation was found between experience scores and solution completeness (Spearman rank order coefficient). Correlations between totals of behaviour and individual's experience of solution completion were also non-significant apart from Reasoning with completeness ($p<0.05$, Spearman Rs). As this suggested reasoning ability may be linked to analytic success, reasoning behaviour was investigated in more detail.

A sequential analysis of reasoning behaviour was undertaken. The life history of each hypothesis was traced by its thematic content until eventual rejection or resolution. Hypothesis reasoning behaviour was categorised as generate, develop, test, confirm, modify and discard. Protocols were analysed to develop network models of reasoning behaviour. The total number of arcs between each pair of categorised reasoning utterances (e.g. generate hypothesis -> modify hypothesis) were counted for each subject, and the total number of arcs for all subjects is represented graphically in Figure 1. Reasoning strategies could be discerned for 11 subjects by reference to these graphs and examination of the nature of hypotheses in the transcripts. All subjects used the weak Generate and Test method.

From the protocol transcripts it was noticeable that subjects with low completeness scores only evaluated hypotheses with general tests which resulted in weak conclusions.

Subjects with high completeness scores generated domain scenarios to evaluate hypotheses, suggesting effective testing of hypotheses may be a determinant of good analytic performance.

Planning behaviour was investigated in more detail. The majority of plans (97%) were short term describing the next sub problem task and were executed within 5 minutes. When questioned about overall plans and strategies in the retrospective protocols all subjects had difficulty verbalising coherent plans, suggesting either the absence of long-term plans or the unconscious nature of such plans. Examination of hypotheses suggested analytic behaviour was triggered by reading the contents of the problem narrative.

Conceptual modelling (drawing the physical data flow diagram) was analysed by investigating Reasoning with hypotheses sharing closely related subject matter (i.e. linked to the development of a conceptual model by a single thematic strand). This behaviour was termed model-based reasoning, and was exhibited by six subjects. They constructed more complete solutions than subjects who were judged not to be model-based reasoners, hence construction of conceptual models appeared to be related to improved analytic performance.

Analytic strategies of novice software engineers were investigated by examining their Planning and Reasoning behaviours in combination of the types of knowledge employed. Concurrent and retrospective protocols were examined for particular phases in analytic strategies. All subjects attempted to scope the problem but eight were ineffective in doing so. Problem scoping included identification of the system boundaries and recognition of key system requirements. Poor scopers became embroiled in the system details before determining the problem space. Five poor scopers may have been misled by SSA method heuristics, and immediately attempted to identify software engineering features (e.g. data stores). Good scopers had high completeness scores, suggesting that effective problem scoping may be another determinant of analytic success.

Only three subjects overtly employed method heuristics throughout the protocol. most subjects proceeded to 'divide and conquer', analysing one area in depth before advancing to the next. Three subjects also disobeyed SSA dictates, by mixing analysis and physical design details in their solution. One subject developed the most complete conceptual model with several sort routines at its core, and there was no evidence that subjects were impeded by the inclusion of implementation features.

2.3 Conclusions

This study suggested good analytic performance was determined by effective hypothesis testing, model-based reasoning and good problem scoping during early phases of analysis. However, contrary to previous studies (e.g. Vitalari and Dickson 1983), results were individually variable (see Figure 2). Some weaker subjects exhibited behaviour to suggest that they relied on method knowledge to structure the analytic task, while better subjects were able to exploit domain knowledge to develop scenarios, structure the problem space and reason more successfully. Our proposed applied mental process model of systems analysis is described graphically in Figure 3. Software reuse is one means of providing novice software engineers with domain knowledge, hence we further investigated software reuse by novice and expert software engineers.

3 Study 2 - An initial study of software reuse by novice software engineers

Reuse of domain knowledge can occur between analogous problems, so an experiment was designed in which software engineers were required to reason analogously during software reuse. Specification-level reuse was investigated, to test whether software engineers could exploit a repository of potentially reusable specifications available in a CASE tools, and to investigate reuse as a means of assisting software engineers during systems analysis. The study also investigated the mental processes and knowledge structures involved in analogous reasoning during specification reuse.

The use of abstraction in the construction of conceptual models, represented as data flow diagrams or entity life histories, to describe application domains at a high level is widely advocated by software engineers. Furthermore, the use of abstract templates and generic objects has been proposed as a method of delivering reusability. In the light of this we investigated the reuse of specifications presented in concrete and abstract forms, as concrete presentation of analogy is often advocated in HCI (for example direct manipulation interfaces) whereas abstraction is valued for problem solving in software engineering. A detailed report of this study is given in Sutcliffe and Maiden (1990b).

3.1 Method

The 30 subjects (23M, 7F) were full-time MSc students in Business Systems Analysis with little or no commercial analytic experience. Subjects were asked to develop a Jackson Structured Development (JSD) process structure diagram for a scheduling function allocating videos to hotels. All subjects had the necessary background domain and method knowledge to complete the task.

A between subjects, two condition experiment was conducted with:
- a Control group, where subjects were given the problem narrative alone,
- an Abstract Analogy group (Group AA), where subjects were provided with the problem narrative and an abstract JSD template of a general scheduling problem,
- a Concrete Analogy group (Group CA), where subjects were given the problem narrative and a JSD specification of a real analogous production planning application.

Each group of 10 subjects was balanced with respect to subjects' experience.

In concurrent protocols lasting 1 hour Group CA and AA subjects were requested to verbalise reasoning and recognition of analogous similarities and the transfer of knowledge from the analogy. A video camera captured all written work, and the verbal protocols were tape recorded. Retrospective protocols and a written questionnaire captured problem solving and reuse strategies. As with the first study subjects' solutions were marked for completeness as well as validity, by comparing them with an expert's solution to the problem.

3.2 Results

All subjects attempted a solution to the video hiring problem. Reuse of analogous specifications appeared to improve the completeness of solutions, and reusable material presented in an abstract form enhanced performance more than presentation of concrete specifications (completeness was measured as a percentage of the expert's solution - the Control group scored on average 24.4%, those given the concrete analogy scored 32.8%, whilst those given the abstract analogy scored 41.1%). This effect was significant for the abstract analogy (T test using the approximating Z distribution for non-normal

populations; Z=2.23, p=0.05). This may be because abstract templates were more easily recognised as analogous (all subjects recognised the abstract analogy, 80% recognised the concrete analogy). Although specification reuse improved completeness it resulted in similar error rates hence reuse does not appear to help the creation of more accurate specifications.

Subjects were divided into successful and unsuccessful reusers, based on a minimum completeness score for their solutions and the explicit transfer of key features of the reusable specification. Eight Group AA and five Group CA subjects successfully exploited the analogy. Completeness and error scores for these subjects indicated that those who reused the concrete analogy produced more complete and valid solutions, although differences between groups were non-significant. These results suggested that reuse from concrete analogies may have been more effective once the analogy had been recognised.

Analysis of errors made during knowledge transfer revealed importance differences in the mistakes made by Group AA and Group CA subjects. The number and naming of key components by the AA subjects were closely related to the analogous template, suggesting they may have copied the material rather than reasoning about it. Specification copying in the sense of lexical tailoring of specification components, without reasoning, accounted for many errors. Errors in eleven of the 13 successful subjects' solutions suggested a general failure to understand the analogous specification. Retrospectively subjects admitted to transferring key concepts without understanding them, or omitting to transfer critical components which they did not understand.

The failure to understand the analogy led to mappings based on surface similarities between the problem and the reusable specification by 11 of the 20 Group AA and CA subjects. Subjects appeared unable to construct mappings where no surface similarities existed.

3.3 Conclusions

This study suggested that novice software engineers were mentally lazy during software reuse, indicating that software engineers may adopt reuse as an easy cognitive strategy when given the opportunity. Abstract analogies appear to be easier to recognise and assimilate, however reuse of concrete analogies appeared to be more effective, possibly because the subject is forced to reason more about the analogy rather than syntactically copy the reusable specification. Novice software engineers are unlikely to have many domain analogies to draw on when constructing new mental models of a problem domain. It is therefore not surprising that when presented with an analogical prompt they take it as a potential ready-made solution.

Results suggested the need to investigate the analogous reasoning behaviour of novice software engineers in more detail. A comparable study of the analytic strategies and analogous reasoning behaviour of expert software engineers was also undertaken, to identify shortcomings in novice's analytic behaviour.

4 Study 3 - A study of reuse by expert and novice software engineers

Protocol analysis was used to investigate the cognitive processes during reuse of concrete analogous specifications, as described in the previous study. Expert and novice software

engineers were requested to reuse a specification to develop a specification for an analogous problem domain. A detailed report of this study is given in Maiden and Sutcliffe (1990).

4.1 Method

Protocol analysis was used to investigate analytic and analogous reasoning behaviour of 10 expert and 5 novice software engineers. The 5 novices (3M, 2F) were MSc students in Business Systems Analysis, with a maximum of 6 months experience of SSA techniques, while the experts (8M, 2F) had up to 20 years commercial analytic experience, and were drawn from different commercial and academic backgrounds.

Subjects were asked to develop a specification for an air traffic control (ATC) system, and to reuse a specification describing an analogous flexible manufacturing system (FMS). All subjects had background method knowledge and were provided with the domain knowledge necessary to solve the problem. Subjects were asked to develop two data flow diagrams (De Marco 1978). The analogous specification was represented using data flow diagrams and supported by a narrative describing key system functions. Each protocol session lasted 75 minutes and was followed by 25 minutes written and verbal retrospective questioning. Data was elicited using methods similar to those in the first two studies.

Protocol transcripts were analysed using the categories described in the first study. Inter-observer agreement (83%) for protocol categorisation was achieved. Subjects' solutions were also marked for completeness and validity, by comparison with an expert's solution to the problem.

4.2 Results

All subjects attempted a solution to the ATC problem. Experts developed marginally more complete and valid solutions (average completeness scores - 64.7% to 62.3%, average number of errors - 1.5 to 1.8).

Frequencies of mental behaviours were counted for 5 minute segments. Assertions and Planning decreased with time, while Reasoning peaked during the middle of the session and Conceptual modelling increased towards the end of the protocol. Experts began Reasoning and Modelling before novices, suggesting experts were able to scope the problem and develop a solution more quickly.

No correlation was found either between experience scores and solution completeness, or between totals of behaviour and individual's experience or solution completion. Experts exhibited more Reasoning behaviour than novices, suggesting as in the first study that reasoning ability may be linked to expert analytic behaviour.

A sequential analysis (see Study 1) of detailed reasoning behaviour was carried out by examining hypothesis life histories. All subjects exhibited a Generate and Test cycle (see Figure 4), however experts exhibited more extensive development and testing of hypotheses and were able to test more effectively (i.e. further extended hypotheses as a result of testing). Examination of protocol transcripts suggested that experts may be more able to reason about a problem, to create test cases and scenarios which can be constructively evaluate hypotheses.

Reasoning behaviour was also analysed by examining the topic focus of subjects' reasoning. Each reasoning utterance was categorised to identify reasoning about the target domain, the reusable domain or the analogous links between the two domains. Experts and novices reasoned equally about the analogy, however experts reasoned more about the target and reusable domains (see Figure 5), and the increase in expert reasoning about the target domain neared significance (t-test:p=0.074). These results suggest that experts may develop a better understanding of the target and reusable problems. The tendency of experts to develop a better understanding was investigated further by examining their analytic strategies and tactics.

Analytic strategies were identified by examination of subject's attention and non-mental behaviour throughout each protocol session. Eight strategies were identified on the basis of mental and physical behaviour:

- Information acquisition,
- Summarise problem documents,
- Reuse the analogous specification,
- Construct a conceptual model directly from the problem narrative,
- Summarise the subject's solution,
- Evaluate the conceptual model against the problem narrative, and
- Evaluate the conceptual model against the analogous specification.

The length of time and reasoning behaviour verbalised of each strategic bout was quantified (see Figure 6). Novices spent more time reusing the analogous specification and summarising their own solutions, while experts appeared to be stronger information acquirers and evaluated their solutions against both the reusable specification and the requirements document. Scoping behaviour was investigated in more detail by examining the analytic tactics used to understand the reusable specification. Experts and novices spent equal amounts of time reading the reusable specification; however investigation of the topic focus of subject's reasoning revealed some differences. Thirty percent of experts reasoned primarily about the source domain while all novices reasoned about analogous mappings between the two applications. These results suggest that novices may have adopted analytic strategies and tactics which did not enhance their understanding of the reusable domain. Examination of individual protocols supported this conclusion. Forty percent of novices began reuse of the conceptual models without reading the supporting narrative, while 80% novices retrospectively admitted to copying and word substitution during reuse. Seventy percent of experts deliberately avoided a copying strategy, and voiced concern about reusing a specification that they did not understand.

Experts appeared to use different analytic strategies to understand the analogy, although results varied by individual (see Figure 6). Sixty percent of the experts who extensively reused the analogous specification tended not to evaluate their solutions against it, while 10% developed more of their solutions directly from the problem narrative exploited the analogy to evaluate their solutions.

Subjects' understanding of the analogy was examined during retrospective protocols. Novices claimed to recognise the analogy by mapping features of the solution such as processes (60% novices and 30% experts), while experts initially recognised mappings between domain concepts (30% experts, no novices). Subjects were asked to identify key analogous mappings, and experts identified more mappings with two key domain concepts; first concept -60% experts, 20% novices, second concept - 40% experts, no

novices. Experts were also more able to generalise these key analogous mappings in abstract terms when requested (experts generalised 5 mappings and novices generalised one mapping). Results suggest that subjects developed analogous links between mental models at different levels of abstraction; novices developed mappings between computer solutions (e.g. files processes) while experts recognised mappings between the underlying problem domains.

Protocol transcripts were re-examined to identify subject's reasoning about the problem domain or the computer solution. Reasoning utterances were categorised as reasoning about (i) the subject's computer solution, (ii) the goals which the subject's computer solution must achieve, and (iii) the problem domains in which the computer system must operate. Experts reasoned more about the problem domain (average 30.2\67.7 utterances) and significantly more about the computer system's goals (average 4.6\13.5, t-test:p=0.037). This indicates that experts may have developed more complete models of the domains which underlie the target and analogous specifications. Better problem scoping may be one cause for improved understanding of the domain.

4.3 Conclusions

Experts appeared to concentrate and reason more about the reusable specification and the underlying problem domain, and to test their solution against both the target requirements and the analogy. As in the first study results suggested that experts were better scopers and testers. Differences between expert and novice performance were small, suggesting specification reuse may have improved the analytic performance of novice software engineers to a near expert level. This conclusion is supported by the poor performance of novices in the first study, and the potential benefits of software reuse on the analytic process identified in the second study.

5 General Discussion

The three studies reported in this paper can be discussed in terms of an applied mental model of analytic behaviour and use of analogical reasoning during software reuse.

The results from the first study suggest that systems analysis is complex and individually variable. Determinants of good novice analytic performance include proper problem scoping, strong solution testing and development of well-formed mental models. The investigation of expert analytic behaviour in the third study supported these findings, and suggested that reasoning about the problem domain underlying a computer system may be another determinant of good analytic performance. Results from the first study suggest that procedural method knowledge failed to support the analytic process, and novice software engineers who followed method dictates too closely may have been hindered from good problem scoping.

Results from the experimental studies suggest that reuse may have enhanced the performance of software engineers, although novices tended to copy rather than reason analogously with the reusable specifications. Copy-cat strategies are unlikely to support successful reuse of analogies since this involves more customisation of the specification, hence tutorial support (e.g. Polson and Richardson 1988) may be necessary to encourage and guide analogous reasoning. A realistic reuse scenario may be a CASE tool incorporated with an intelligent advisor (e.g. Cummings and Self 1989) which can 'reason' along side and prompt the software engineer. We are developing didactic

strategies and tactics from detailed analysis of expert reuse strategies, and issue-based diagnosis may be based on software engineers bugs and misconceptions identified in the third study.

Results from the third study support several existing theories of analogy. Expert software engineers reasoned about the analogy in terms of causal relations which described the domain, supporting Gentner's structure-mapping theory (1983). Both experts and novices also generalised during analogous reasoning, supporting Greiner's (1988) model which purports that analogous transfer only occurs between concepts which share a common abstraction. Gick and Holyoak (1980) propose analogous reasoning between two problems is only possible if they have common goals. Many subjects recognised the analogy through similar system goals, hence similar goals appeared to support analogous reasoning. Existing theories of analogy are supported by simplistic examples. The scale of software engineering problems employed in these investigations suggests analogy may be more complex than implied by existing theories. Analogy may have goal, structural and abstraction components, each of which is a critical determinant of an analogy.

References

Adelson B., 1984, When novices surpass experts: the difficulty of a task may increase with expertise, Journal of Experimental Psychology: Learning, Memory and Cognition 10(3), pp 483-395.

Anderson J.R., 1983, The architecture of cognition, Havard University Press.

Cummings G. and Self J., 1989, Learner modelling in collaborative intelligent educational systems, in Teaching knowledge and intelligent tutoring, edited by P. Goodyear, Norwood, NJ; Ablex.

De Marco T., 1978, Structured analysis and systems specification, Prentice-Hall.

Ericsson and Simon 1984, Protocol analysis, MIT Press.

Gentner D., 1983, Structure-mapping: a theoretical framework for analogy, Cognitive Science 7, pp 155-170.

Gick M.L. and Holyoak K.J., 1980, Cognitive Psychology 12, pp 306-355.

Greiner R., 1988, Learning by understanding analogies, Artificial Intelligence 35, pp 81-125.

Guindon R. and Curtis B., 1988, Control of cognitive processes during software design: What tools are needed ?, Proceedings CHI'88: Human Factors in Computer Systems, edited by E. Soloway, D. Frye and S.B. Sheppard, pp 263-269, ACM Press.

Koulek R.J., Salvendy G., Dunsmore H.E. and Lebold W.K., 1989, Cognitive issues in the process of software development: Review and reappraisal, International Journal of Man-Machine Studies 30, March 1989, pp 171-191.

Maiden N.A.M. and Sutcliffe A.G., 1990, Cognitive models of software reuse, in preparation.

Newell A. and Simon H.A., 1972, Human problem solving, Prentice-Hall.

Polson M.C. and Richardson J.J., 1988, Foundations of intelligent tutoring systems, Lawrence Erlbaum Associates.

Sutcliffe A.G. and Maiden N.A.M., 1990a, Analysing the analyst: cognitive models of software engineering, submitted for publication.

Sutcliffe A.G. and Maiden N.A.M,, 1990b, Software reusability: Delivering productivity gains or short cuts, Proceedings INTERACT'90, August 27-31, Cambridge, UK.

Vitalari N.P. and Dickson G.W., 1983, Problem solving for effective systems analysis: An experimental exploration, Communications of the ACM 26(11), November 1983, pp 948-956.

Figure 1 - Network model of hypothesis life histories for all subjects: figures on the arcs represent the total number of transitions which occured.

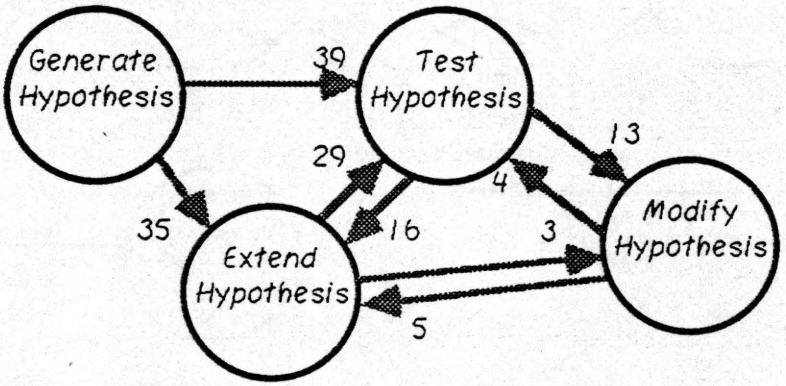


Figure 2 - Summary of possible reasons for poor and good analytic performance

Subject(s)	Good problem scoping	Conceptual modelling	Critical testing
E	√	√	
G		√	
I	√	√	√
J	√	√	√
K	√	√	√
N		√	√
P	√		√
D, H, M, O R & S			

Figure 3 -Applied mental process model of systems analysis

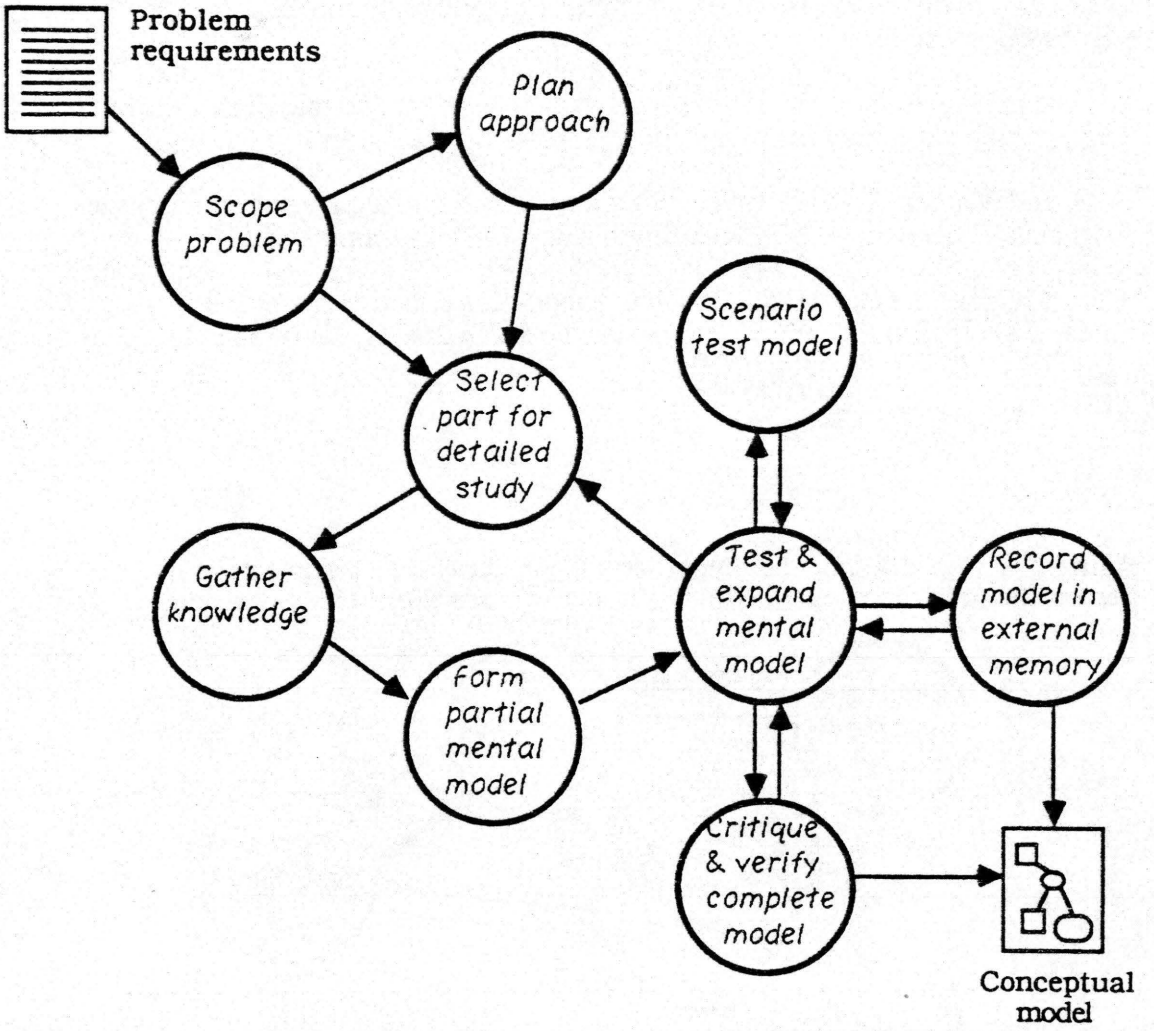


Figure 4 - Hypothesis life histories representing averages number of transitions which occured for experts and novices - only major reasoning categories are shown

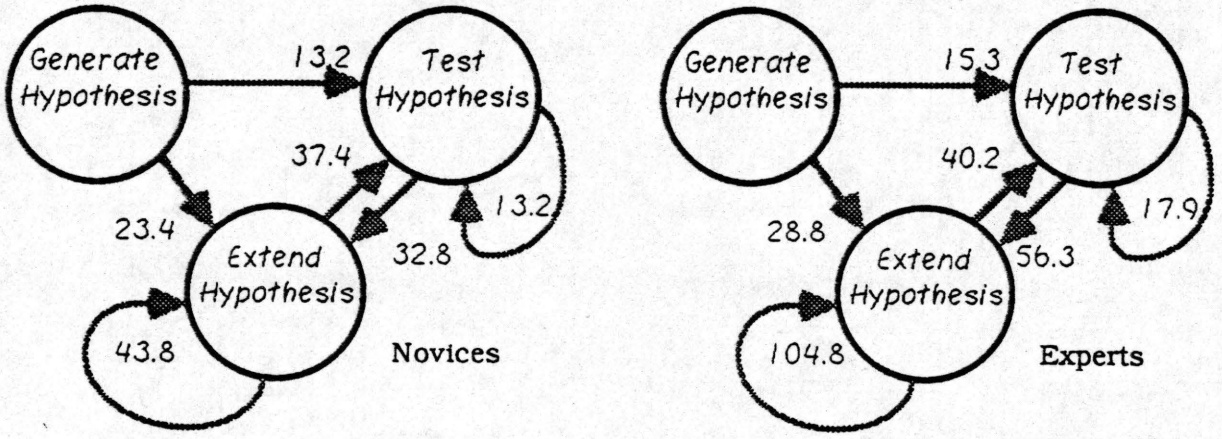


Figure 5 - Total number of utterances made by novices and experts about the target domain, source domain and the analogy

Topic focus of subject's reasoning	Number of reasoning utterances	
	Novices	Experts
Target domain	146.8	215.8
Source domain	9.8	24.5
Analogy	70	71.5

Figure 6 - Average lengths of time and reasoning utterances during reasoning strategies spent by novices and experts

Analytic strategy	Avrge times (mns)	
	Novices	Experts
Information aqusn	19.6	21
Summarise problm	0.4	0.3
Reuse specification	24	14.6
Construct solution	5.4	10.2
Revise solution	3.2	0.8
Summarise soluton	8.25	3.3
Evaluate vs analgy	0.6	3.4
Evaluate vs reqts	9.8	11

Analytic strategy	Av reasoning utt.	
	Novices	Experts
Information aqusn	37.8	86.4
Summarise problm	2.4	5.8
Reuse specification	119	111.3
Construct solution	16.2	51.3
Revise solution	9	4.8
Summarise soluton	33.2	23.7
Evaluate vs analgy	3.4	24.3
Evaluate vs reqts	28.2	51.1

ON-LINE ASSISTANCE IN FLIGHT SIMULATION (*)

MICHELE VISCIOLA, ARMANDO ARMANDO
AERITALIA - Centro di Simulazione - TORINO
SEBASTIANO BAGNARA
UNIV. di SIENA - Dip. di Filosofia e Scienze Sociali - SIENA

(*) Findings reported in this paper are part of a research project granted by the Italian "National Research Council" (Progetto Finalizzato FATMA) on the theme: Methods of analysis for the reliability of complex systems, of accidents/incidents, and methods for the analysis of errors".

ON-LINE ASSISTANCE IN FLIGHT SIMULATION

INTRODUCTION

Interagent modalities of communication is considered as one of the major points of concern for the design of systems whose purpose is to assist users in their intention generation processes and goal achievement. At the moment this seems to be also the weakest point into models of interaction between agents (may be humans but also machines) who share knowledge and expertise. Our understanding of factors and of contextual reasons that guarantee good cooperation relies very little on investigation about modalities of communication and of knowledge transfer.

The analysis of communication's modalities has also interesting implications for the design of the so called "Glass Cockpit". Not surprisingly, the flight deck of an advanced cockpit has been compared to an automated office (Wiener, 1985; Speyer and Fort, 1985) and pilots of future aircraft to managers assisted by computer-based communication tools (Chambers and Nagel, 1985). The problems with these methaphora is that they help to conjecture about the impact of electronics on pilots abilities but they do not assist the designer to resolve key questions such as dynamic task assignment and task coordination in dynamic environments.

This contribution presents preliminary results on an explorative investigation about modalities of assistance during a training stage. These modalities of assistance or patterns of communication are aimed at finding germane conditions for the detection and correction of errors. Moreover, they assist the pilot to avoid erroneous decisions and actions well before they may happen.

Although they are descriptive of a biased context (such as that of learning into a simulator) our research has not the same constraints that have investigations on simulation of learning such as that on "training wheels" (see, Carroll and Carrithers, 1984; Catrambone and Carroll, 1987). The assistance is available any time during the training (i.e., it is "on-line"), does neither follow guidelines nor prescriptions, can be activated both by the pilots and by the instructors and concerns a full system learning process. Moreover, this investigation does not aim at assessing artifacts such as for example an help system. Error detection and recovery capabilities are augmented by experienced pilots.

Our general hypotesis are that effective learning in real world does rarely occur without assistance and that in learning domains error detection and recovery are mediated by the type of assistance available. Our purpose is to investigate what determines effectiveness of "intelligent" advisory in these contexts. And the expert-trainee communication's patterns seem to typify the ideal conditions for our purpose.

METHODOLOGICAL APPARATUS

This investigation is part of a research project on methods of analysis of pilot errors and on the definition of principles for the assistance in flight of problem-solving situations. The method used for error identification and classification is already described in Visciola et al.(1989). The following description is focused only on aspects of relevance for the presented findings.

Training procedure

Subjects of our investigation are all licenced pilots. The training stage we are following and observing aims at providing the pilots of the necessary familiarization into the simulator with a new aircraft. Activities of transition into the simulator before the passage to training activities in the aircraft have nowadays ecological validity. Flight simulators reproduce with great fidelity both environmental conditions and the characteristics of aircraft (the overall duration of the training can be registered on the personal flight-book as 2h. of real flight).

The sequence of the training in the simulator is incremental. Trainees start with the acquisition of confidence in the flight instruments, cockpit layout and checklists, then proceed with learning the aircraft's handling qualities and to apply procedures. At the end of this process they are able to simulate full missions. The overall process lasts 7 hours for each pilot; however, since any training stage is arranged for a group of four pilots, we do not exclude that the extra-time of training, spent to observe friends, has effects on the learning acquisition.

Any group of pilots has its own instructors (usually two for each group) that follow all the training from the control room. The control room is provided with software keys, through which operator can intervene on the current process, for example, by simulating emergencies, perturbations in the environment, etc. Moreover, it has also video and audio recording instruments which make also possible the instant observation of all the pilot actions into the cockpit.

Verbal interaction

For the purposes of our investigation pilots are requested to verbalize their thoughts and their important decisions. Most of the times this request does not represent an additional task. Verbalization during training is both a necessity and a natural process for the pilots.

What is important for this part of the research is the verbal interaction between the instructor and the pilot. We do not make specific requests to the instructors. However, we have noticed that, as a consequence of our presence during the training and, perhaps, as an effect of their connossaince of our scope, they have developed a specific sensibility in detecting errors and some of the instructors do now use also words and concepts typical

of our method. This, of course, does not mean that we have altered tacitly their attitudes towards training. These consequences are, probably, both inevitable and a sign of efficacy of our method.

As for our method we use a large criterion to identify forms of errors and rely at a large extent upon the nature of the communication. Errors are never left undetected even when they do not compromise seriously the current state of affair in the simulated mission. Therefore, our taxonomy comprises also those episodes of scarce utilization of cockpit resources, of bad optimization of action shemata and of small deviations from the "correct" decision-action sequence of performance.

Error detection modalities and related time of intervention are almost entirely left to the decision of the instructor. They, in fact, may be advised soon after error occurrence or after a varied time interval (i.e., during the debriefing of the mission or in a given time of the mission itself). The time when error is detected plays an important function for learning effectiveness.

As we shall see, it seems that the time of detection has at least the same importance of modality of detection. However, from the methodological standpoint, the important issue is that hitherto we have decided to not interfere significantly into the communication process. For example, the well known distinction between the modality of giving instuctions on "how to do" a correct performance, and instructions on the consequences of the performance on the systems (i.e., "how the systems work") are both significant modalities of assistance in error detection, and at the moment we can only make conjectures on their impact on error correction processes and on their capability to facilitate a permanent learning.

This distinction as well as the choice of time of detection of an error might be both worth of experimental approach since the instructors seem to follow tacit criteria to decide "what" and "when" is better to give advise and, viceversa, pilots use their own criteria to decide "what" and "when" to call for assistance.

However, "naturalistic", phenomenological approaches have the advantage that data not comprised into the experimental design are not lost. From this point of view, collected data are not biased by the observator/experimenter. Data reflect entirely the communication modalities between pilot and instructor devoted to error detection, recovery and prevention.

For the analysis of patterns of communication we have not used quantitative methods for the simple reason that they would have not improved our understanding of the matter under investigation. To describe how we have analysed data it might be sufficient to sign up the sequence of our procedure. The verbal data were first transcribed on paper and then codified according to the primary source of communication (i.e., **who activated the process**) and according to the contents of the communication (i.e., **what was the reason that activated the process**). In some circumstances the reason was not immediately clear and we had to rely on our notes to interpret the ultimate meaning of the communication

process. This methodological limit, even though it was not really compelling, could be interpreted as a sign that even the "on-line" assistance given by human experts may encounter the problems of intention indeterminacy that usually are registered as troubling cases in the use of help systems as well as in off-line consulting (see, Carroll and Aaronson, 1988).

PATTERNS OF COMMUNICATION

At the end of our analysis, which comprises two full sessions of observation (two groups of four pilots) we have found out ten common patterns of communication (see fig.1). Among these, only three were genuinely activated by the pilots, five by the instructors and two represented mixed cases; that is the communication was either in between a different pattern (it was the consequence of the current main communication process) or was triggered by events that required collaborative solutions. As a consequence, communication activated by pilots are called **request** for assistance patterns, while those activated by instructors are called **advisory** patterns.

Request of information not accessible or not directly available
Request for procedure clarification
Request for feedback under subjective uncertainty
Advise after an incorrect action
Advise on coordination of activities
Advise on how to interpret events and displayed data
Advise for short-cut procedures
Advise on deviations between simulated and real flight
Explanation of sub-systems functioning
Updating and validation of procedures

Fig.1: Ten common patterns of communication in on-line assistance

*** Request of information not easily accessible or not directly available.** This pattern takes place when the pilot lacks information to take actions and he does not know where to get it or wonder whether exists a quicker access to it than that learned in previous experiences.

It seems that possibility to make comparison is functional to a quick and efficient learning; so questions are often addressed by having in mind experiences made on other aircraft. Implicit in these requests are also expectations that new systems will improve flight operations: In fact some times questions take place before real systems capabilities are assessed. However, given the "competence" that these pilots already have, regardless their actual knowledge of the particular aircraft, these questions are never naive questions.

And it seems that in this kind of request there is also an implicit need to have a simpler access to information. In addition, by taking into account the moment when requests are made, (usually, during planning activities, whose interruption may be detrimental) it seems that they are made in order to gain time and to optimize mental resources.

For example, during the navigation on a planned route one pilot asked how to know his time-deviation from the next waypoint given the current values. Usually, the planning of time on the points chosen for the navigation is made before take-off with simple calculation which take into account flight parameters like speed, fuel consumption, configuration and weight of the aircraft, etc. These are "template" values and are used by the pilots for a moment-by-moment accomodation of flight parameters so that the deviations from the plan can be recovered or maintained within tolerance's limits. The pilot wanted to know how to get that value without making mental or paper and pencil calculations.

* **Request for procedure clarification.** While in the previous patterns requests concern information necessary to accomodate the already selected plans of action in this second pattern requests are mainly aimed at obtaining a better or more complete model of systems' logic. The questions are therefore focused on specific technical details of the systems and evidence specific lacks in the current model of the learner. They take then the typical form of interrogation about the consequences of given actions or about the extent of restrictions of cockpit resources during system failures. Examples of this pattern are requests on what happens to stored data when the main computer is turned off or which are the conditions in which altitude data disappear on main display.

While in most cases lacks of current learner's model are explicitly recognized and requests are therefore coherentely formulated, there are also cases when pilots make "uncorrect" anticipations of events and correlate data co-occurred for mere coincidence.

For example, during the simulation of the Head Up Display failure one pilot noticed that one of the Head down instruments (i.e., the Horizontal Situation Indicator) was not working. He made the false deduction that the two events were due to the same cause. In reality the HSI was simply disactivated into the simulator for unrelated reasons.

Active misconceptions and false correlations seem to be frequent and common strategies during learning to improve knowledge on the systems characteristics. It seems that the state of full openness to get results from experiences on simulator triggers processes of causal thinking that have to be tested and verified. The important aspect of these common strategies is in fact that they are not "cooked and eaten" strategies. They do not stop once data have been correlated or conceptions (hypothesis) have been generated.

In other words it seems that awareness of the fact that assistance is available and that answers may be obtained is essential to active involvement of learner. And although it is not enough clear whether these active misconceptions are produced consciously, it seems that they are very functional to learning process.

* **Request for feedback under subjective uncertainty.** The main characteristic of this pattern of communication is the explicit request of help to resolve the feeling of uncertainty soon after intentions have been generated and plans of action have been selected. In these cases events related to the current state of the systems are not fully under pilot control and anticipation mechanisms seem to be vague as feedback may not be at a prompt disposal or because pilot is not yet able to get significant feedback from available information.

When this pattern is activated, the verbal interaction between pilot and instructor takes the form of interlocutory acts and very few hypothesis are generated and evalued before they are definitely discarded or accepted. The key-goal for the pilot seems to be driven by the desire to ascertain whether he has made some mistakes or the system is responsible of the assumed "anomaly". What is in fact necessary to activate this pattern is not that something is going wrong, while it is essential that the pilot has got the impression of not having a full control or of a mismatch between his capability to generate correct hypothesis and to assess them properly.

The following example may better illustrate this pattern of communication. The pilot was in navigation automatic mode. When the system is in this state, any time the aircraft crosses over a waypoint of the planned route the distance to the next waypoint is displayed. In this example the pilot decided to land on the first airfield of the route after a diversion on a point out of the route. He selected the desired waypoint on the navigation data entry and prepared for the approach manouvres. As soon as he reached the airfield he was surprised that the displayed distance was much higher than that he was able to evaluate visually and verbalized his suspect that something was wrong.

The instructor drew the pilot attention on an important information that seemed to be neglected by the pilot. The distance was in fact referred to the next waypoint, as the display signed. At this point the pilot seemed to have realised the "trick" and selected once again the waypoint of the airfield. The instructor kept the pilot going on his explorations.

After the approach manouvres, situation presented the same conditions and this time the pilot was disappointed as he noticed that the waypoint had turned to be not that of the airfield. The difficulty of the pilot was due to his misconception that once a waypoint was manually selected the system would automatically have reverted to manual mode.

* **Advise after an incorrect action.** This pattern is activated by the instructors as soon as an error has been made. However, in most of cases the error is not soon evidenced even though it has been detected. In fact, the pilot seems to be unaware of the error episode and the instructor does not want to loose opportunity to take profit from it. So the communication takes the form of an interlocutory act aimed at drawing gradually the pilot attention on aspects of the situation that could improve his assessment of the task domain.

Task of the instructor is to facilitate error detection. Questions are therefore addressed quite softly and start by asking to make controls. If the detection is not immediate at first address, instructors make correlations among data without drawing conclusions. The solution is given only after the failure of the second address.

Correlation among events seems to be an effective advisory pattern if the error has not triggered other anomalous events. In this latter case, that is not unfrequent during simulation, pilot and instructor have not activated a timely error detection process and the task-context does not offer many margins for recovery either.

However, it seems that to prevent such difficult conditions for recovery (that yet are of much importance for training effectiveness) instructors choose to give the advisory under the "take care of" form. Usually this form of advise is given during highly dynamic situations and it is characterized both by indications about the consequences of actions and by suggestions on how to obtain good performance.

* **Advise on coordination of activities.** This pattern of communication is activated by the instructor any time he notices that performance may be improved and time resource may be optimized. For example the checklist for starting manouvres is a very time demanding task even in the rare cases it has been sensibly designed. So the instructors spend many explanations on modalities that allow to spare time. Advises are then given on the optimal sequence of performance, when to suspend control, how to integrate checks.

Some times this pattern of communication does not take the form of an explicit advise. And the instructor inserts simple marks that have attention getting properties into the current pilot activities. This form of advise is in fact a very keen one. Apparently the pilot has a full control of the situation and in fact he does not require any form of assistance: He is quite satisfied with his level of performance. What seems to be lacking at the expert's eyes is the necessary flexibility to get the maximum profit from the situation and to increase the margins of error tolerance. This flexibility is essential in dynamic conditions to be ready to face, for example, interruptions, error-prone situations, unexpected events, emergencies.

* **Advise on how to interpret events and displayed data.** This pattern of communication has relevant commonalities with the latter one. They are both aimed at improving the pilot's capability to get the maximum of information from the current events. This capability is in fact essential to the generation of sensible hypothesis during the activities of plan selection.

As in the previous pattern of communication the instructors try nicely to modify the threshold of risk acceptance. In other words the aim is to reduce the likelihood that errors might arrive at a point of "non-return". The difficulty for the instructor is to choose the time when advise has to be given and, moreover, the modality to give the advise. In fact, as previously, apparently simulation is under control, therefore error detection process might result extraneous to the current activities and be perceived as a distractor.

On the basis of the episodes that we could observe, the major difficulty in these cases is to choose whether to keep the process going on or to interrupt it. At the moment we are not able to say which of the two modalities is better and is preferred. However, it seems that in some cases advise is practically unsuccessful even if given timely. These situations are characterized by the subjective feeling of being under high load and by the inability to evaluate correct hypothesys and to decide task priorities. And the instructor seems to have definitely lost control of situation or has deliberately left the pilot alone in his exploration. In both cases what shows to be difficult to communicate is the weight that is convenient to give to evident and hidden clues.

On the contrary, stories of success can be described as the result of a sort of "conspiracy effect": The pilot is able to perform his routine activities and to spare time resources, is capable to anticipate events also as a consequence of the fact that the instructor has provided him with information that make possible anticipation of events (usually these information have connotative properties that are not intentional products of design) and the advise is consistent with the context-driven task. Pilot is "taken by hand" from the very beginning of the problem-solving situation and left alone only at the moment of the task execution.

* **Advise for short-cut procedures.** This form of communication reveals more clearly than the other patterns that errors may depend also upon the pedantic application of procedures. Many times procedures are too long for the time available or there are short-cut procedures that fit better to the situation. However, often these shortest procedural forms are not explicitly provided by the system designers and have to be assessed and evalued by the end-users. One of the purpose of the training is also to find out suitable conditions for a plain use of these "hidden" system potentialities.

* **Advise on deviation between simulated and real flight.** Although the flight simulator guarantees high fidelity, yet there are conditions in which the simulation might exceed the limits of the aircraft. It is then essential that the pilot be aware of consequences when these limits are not respected. As for the previous forms of advise, this pattern of communication contributes actively to the generation of an implicit theory of risk. Episodes of errors detected during the simulation are usually not forgotten if they receive a careful examination. They allow to anticipate events so that, on one side, eventuality to reproduce error-prone conditions can be reduced and, on the other side, coping schemata for anomalies can be quickly available.

Without these experiences it seems almost impossible to gain a clear perception of risks associated to decisions and, accordingly, to assess degree of freedom and subjectives limitations in specific situations, the extent at which errors can be tolerated, recovered. Errors experiences and their detection are essential to generating an implicit theory of risks.

* **Explanation of sub-systems' functioning.** This form of communication concerns technical details that might be also ignored by the pilots without any apparent implications on the skill acquisition. So these forms of communication represent only "curiosity" for the average pilot or could testimony "deep knowledge" for the expert.

* **Updating and validation of procedures.** Before procedures can be definitely assessed they have also to be tested in many different conditions that could be not foreseen during system design. It is therefore not a rare event that errors reveal unpracticality of procedures or the need to be updated. Usually these findings do not give source to design modification while they are treated as constraints for flight operations.

CONCLUSIONS

Our investigation about "on-line assistance" in the context of training in a flight simulator has shown ten consistent patterns of communication between pilots and instructors. These patterns are aimed at finding the modalities and the time-context that fit better to the error detection and correction activities. These forms of communication can be considered representative of the majority of error episodes cases we have observed. They can therefore be used as a framework for the design of a data-bank which comprehends both error episodes and modalities of assistance in learning activities. The next step of our research foresees the assessment of the outlined patterns of communication in real world contexts outside learning domains.

REFERENCES

Carroll J.M. and Aaronson A.P. (1988) Learning by doing with simulated intelligent help. **Communication of the ACM.** 31, 1064-1079.

Carroll J.M. and Carrithers C. (1984) Training wheels in a user interface. **Communication of the ACM.** 27, 800-806.

Catrambone R. and Carroll J.M. (1987) Learning a word processing system with training wheels and guided exploration. In J.M. Carroll and P.P. Tanner (Eds) **Human Factors in Computing Systems IV.** Amsterdam: North-Holland.

Chambers A.B. and Nagel D.C. (1985) Pilots of the future: Human or computer ? **Communication of the ACM.** 28, 1187-1199.

Speyer J.J. and Fort A.P. (1984) Communications. Major Human Factor in Cockpit Design. **Society of Automotive Engineering.** 831530.

Visciola M., Armando A., Bagnara S. (1989) Errors While Learning in Flight Simulation. In **Proceedings of the 2nd European Meeting on Cognitive Science Approaches to Process Control.** Siena, Italy; Oct. 24-27, 1989.

Wiener E.L. (1985) Beyond the Sterile Cockpit. **Human Factors.** 27 (1), 75-90.

User Models

Training of Pascal novices' error handling ability

Carl Martin Allwood
Carl-Gustav Björhag

Department of Psychology
University of Göteborg
Sweden

Abstract

The present study reports on the effects of giving novices in Pascal one hour of training in the debugging of their own programming bugs. The training presented subjects with results from previous research on novices' debugging as well as fairly detailed instructions on how to debug a program. Furthermore, the instructions included explanations of common error messages from the computer. The debugging of the eleven subjects receiving training was compared to ten control subjects occupied with programming on their own during the training hour. The results show that the subjects who received training debugged a significantly larger proportion of their errors compared with subjects who received no training. The results suggest that especially the semantic programming errors were easier to debug for the trained group. No differences in detailed debugging behaviour were found between the groups.

Introduction

Debugging of computer programs is important because it can help prevent dangerous accidents and other unfortunate events. Furthermore, previous research shows that debugging occupies a substantial part of programmers' time. For novice programmers, two studies (Allwood & Björhag, 1990; Miller, 1974) indicate that debugging can occupy at least half of the time spent programming. From a theoretical perspective debugging is interesting as an example of evaluative processes. Evaluative processes give the individual an opportunity to stand back, reflect and possibly change his or her understanding or actions from a more deliberated perspective. Considering the practical importance of debugging it is surprising that most textbooks on programming only contain a few pages on how to debug computer programs.

The importance of debugging clearly indicates the desirability of developing methods of training novices to debug computer programs. Research developing methods or aids and/or comparing various methods or aids for debugging appears to be somewhat more common (for example, Myers, 1978; Gilmore & Smith, 1984) compared to research evaluating training of debugging skills. In fact, we have been able to locate only two studies (Carver & Risinger, 1987; Katz & Anderson, 1988, experiment 4) evaluating training in debugging. The study by Carver and Risinger demonstrated the effectiveness of a training regime in debugging developed for sixth graders' programming in LOGO. Half an hour of instruction in debugging resulted in improved debugging for the experimental group as compared to a control group.

The instructions in the training given by Carver and Risinger emphasized the importance of a step-by-step approach when debugging, going from more general questions to more specific ones. When debugging the student was first to ask him- or

herself, What is the problem? and then, What type of bug could cause the problem? Next, the student was instructed to decide if the program to be debugged had subprograms and if so if the bug might be located in a specific subprogram, in a REPEAT-, or in an IF-construction or after a specific command. If none of these questions helped, the student was asked to read every command and decide whether or not it was correct. Finally, after having made the fix, the student was asked to re-test the program.

The students' debugging skills were tested "shortly after the debugging instruction" (p.159). In the test-phase, the students were asked to debug two programs written by somebody else but that they had tried to code themselves earlier. Each of the two programs contained about 20 lines with four clusters of five lines having nearly identical structure. The students were instructed "to fix all the bugs" (p. 158). Each program contained eight planted bugs and the students were allowed "one class period" to debug each program.

Katz and Anderson (1988) used planted bugs and compared the training of three different ways to locate a buggy line in programs on the order of ten lines. Subjects' bug-location strategy was manipulated through the use of different debugging interfaces which forced the subject to use a specific bug-location strategy. The first bug-location strategy was to have subjects go through each line of code from the start in the order the program would be executed by LISP. The second bug-location strategy was to have subjects locate a program line to look at starting from the buggy output of the program. The third bug-location interface, finally, asked the subjects directly which area in the program they would like to look at. For all three conditions, the subjects were asked to judge any line they looked at with respect to whether or not it was correct, and to supply a correct version of the line if the line was judged to be erroneous.

The results showed that forcing the subjects to use a specific bug-location strategy during the debugging of six short programs resulted in the subjects' tending to use the same bug location strategy when debugging six further programs with the exception that the subjects in the first condition read the programs in prose order rather than in execution order. However, there was no clear relation between bug-location strategy and correctness of judgements as to whether a certain line of code was correct or not, nor with respect to replacing a buggy line with a correct new line or not.

In the present study we trained novice students in Pascal to debug their own programs. One hour of debugging instruction and training was given and the testing took place immediately after this.

In common with the study by Carver and Risinger (1987) and by Katz and Anderson (1988, experiment 4), most studies on debugging have dealt with the debugging of bugs planted by the researcher. However, debugging of computer programs may also take place in a context where the program to be debugged is written by the same person who is doing the debugging. Both of these situations are presumably common in programming contexts. As suggested previously, there may be important differences between the cognitive processes involved on each of these two occasions (Allwood, 1986; Allwood & Björhag, 1990; Katz & Anderson, 1988).

For example Katz and Anderson (1988, experiment 3) compared the debugging of own LISP-programs and the debugging of LISP-programs created by others and found differences between these conditions. The results showed that subjects debugging their own programs found more of the bugs and took shorter time to do so as compared with when they debugged programs written by others. Furthermore, the two conditions differed in the debugging strategies employed by the subjects. Debugging of own produced programs involved less time used trying to understand the program. It was also found

to involve more use of backward-reasoning, i.e. a bug location strategy starting from the buggy output of the program, in contrast to forward-reasoning, i.e. reading and evaluating the program line by line from the start.

A comparison between two studies by Gugerty and Olson (1986) and Allwood and Björhag (1990) also indicates that comprehension processes play a smaller role when debugging own produced programs as compared with other produced programs. Gugerty and Olson (1986) used planted bugs and compared the debugging of experts and novices. Their results indicated that the experts' advantage in debugging was due to their better skills in program comprehension. Allwood and Björhag (1990) studied novices' debugging of their own programs and in this context found evidence of limitations to the importance of program comprehension processes in the debugging phase.

Furthermore, studies of debugging of planted bugs usually do not include the stage where the programmer goes from not suspecting a bug to supposing that the program is buggy (Allwood, 1986). It is also the case that studies of debugging using planted bugs most of the time use fairly simple bugs and, moreover, the remaining program, if written by the researcher, tends to have a good structure. In contrast, the bugs in studies using own produced programs can be complex and programs of any length written by novices may have a poor structure.

Debugging is clearly a complex cognitive activity the details of which are difficult to study. In order to structure the situation, nearly all researchers have, when studying debugging, attempted to simplify the situation in various ways. Such simplification often involves putting restrictions on what the subject can do when attempting to debug a program. For example, in many studies subjects were not allowed to interact with the computer when attempting to debug the program (for example, Adelson, 1984; Vessey, 1985; Wiedenbeck, 1985). It appears likely that the same person will use different strategies for debugging with access and without access to a computer.

Other studies have used very small programs (sometimes less than ten lines); have only studied a limited part of the debugging process; have introduced rules for the debugging strategies subjects may use; and have automatically eliminated corrections which did not eliminate the bug, etc (e.g. Katz and Anderson, 1988). The effect of the introduction of such limitations on subjects' debugging is not clear. Moreover, in most studies, subjects' debugging processes have not been followed step by step but instead other types of data have been collected, for example subjects have been asked to give retrospective reports on their debugging.

The present study is an attempt to study the effect of debugging training in a somewhat more realistic situation compared with most earlier studies of debugging. We tested the effect of our debugging training by, for two hours, letting the subjects attempt to create a bug-free program, a correct version of which amounted to approximately 40 lines of code. The content of the training was to a large extent derived from the weaknesses in subjects' debugging processes as evidenced in the study of Allwood and Björhag (1990) and other studies. For example, Allwood and Björhag (1990) found that 32% of the bugs indicated by error messages from the compiler were not eliminated by the subjects. Some reasons found for this were that the subjects sometimes ignored the error messages, that the error messages were occasionally misleading and that the subjects did not always correct the bugs in the order in which they appeared in the program. The last mentioned fact led to the subjects attempting to correct "errors" which were due to the fact that the compiler had parsed the program erroneously as an effect of an earlier bug in the program.

Method

Subjects

Twenty-two undergraduate students from the department of Industrial Economy at the Chalmers University of Technology, Göteborg, Sweden participated in the study. All were novices in programming and had just completed the first seven weeks of an introductory course in Pascal.

No clear differences were found between the subjects with respect to background experience in programming or with respect to their rating on a five-step scale of how much fun they thought it was to program. Three subjects in each group did not have any prior programming experience, most of the remaining subjects had only taken Basic courses in school at the pre-university level. Only four subjects had a computer at home (three in the control group); for the most part the computer was used for games.

All subjects were volunteers and were paid for their participation. Eleven subjects were randomly assigned to an experimental group and 11 to a control group. One of the subjects in the control group was later deleted from the study due to technical problems with the video recording of the subject's screen.

Materials

Programs. The version of Pascal used in the study was a version of Standard Pascal, running on a NORD500 computer under the operative system Sintran. The text-editor used when entering the code into the computer was an editor called Ped.

Training instructions in debugging. The training instructions were five pages long. The first section of the text presented results from previous research (our own and others) on deficiencies in novices' debugging of their own code. Novices' debugging was described as generally being too passive. Common deficiencies mentioned were that novices often do not 1) sufficiently attempt to follow the flow of information in the coded program, 2) attempt to understand the content of the error messages given by the computer, 3) enter the program on the screen in order to find the place in the program referred to in the error message given by the computer and 4) sufficiently attempt to understand the effects of the corrections that they plan to carry out in the program before the corrections are carried out.

The subjects were also warned that the compiler might sometimes parse the program in an incorrect way and because of this give erroneous information about the bugs made. However, it was stated that the error messages usually give very useful information.

Novices' debugging was compared to that of experts and it was concluded that novices more often engage in debugging activities which are less intellectually demanding. Furthermore, research results were described which show that novices often debug their program in an unsystematic way, for example by attempting to debug errors occurring later in the program before errors occurring earlier in the program. This might result in the correction of "errors" which would have disappeared by themselves when the earlier occurring bugs were corrected. The unsystematic and mechanical use of test values by novices when testing the program was also commented upon.

In the next section of the instructions the novices were asked to debug their programs according to the following instructions (in translation from Swedish):

"1. Start to correct the bugs in the order in which they are marked in the program (i.e. not in the order they are given in the compiler's "error list"). Try not to deal with too many bugs at the same time; preferably deal with only one bug at a time (you might get rid of a few sequel bugs at the same time). When you start to correct a bug, first read the error message dealing with the bug and explain the content of the error message.

2. When you have understood the meaning of the error message for a certain bug and you do not immediately realize what the error is, bring up the program on the screen. Thereafter localize the area in the program which the compiler might have indicated. Next, look for syntax errors if the text of the error message gives you reason to suspect this. Examples of syntax errors are: omission of a semicolon (;), one semicolon too many, and the use of loop-constructions which are not grammatically correct.

If the error message indicates other types of bugs try to trace these by carefully following the flow of information from the beginning of the program. If this seems completely unnecessary, start to follow the flow of information a little above the place in the program suggested by the compiler as the place of the bug. For each program line that you read in the program state aloud what is carried out in that line. Then consider if it fits in with what the program did earlier and if it fits with what you planned that the program should do.

3. Before you correct a bug, carefully make it clear for yourself what is the nature of the bug. Before you carry out a correction you should make yourself aware of what consequences the change will have on the rest of your program. Also make sure your change does not violate any of the assumptions which the program is dependent on.

4. If you have corrected a bug which, in turn, might have given rise to other bugs you should immediately investigate the effect of your correction by compiling or running the program in order to find out if your change has had the intended effect."

In the next section the subject was given information about the local compiler. This information included information about bugs which diminished the compiler's chance of interpreting the program. For example, it was pointed out that a semicolon directly preceding an "ELSE" usually leads to the rest of the program being completely misinterpreted by the compiler. Two more such examples were given together with typical error messages indicating the presence of such bugs.

In the final section of the instructions the meanings of 31 common error messages concerning compile-time errors were explained as were the meanings of four common messages concerning run-time errors.

Training problems. In the training session four small programming problems were given to the subjects. The simplest of these problems was to create a program adding five numbers given to the program by the user from the keyboard. The most complicated of the four problems was to construct a program which could read in values and compute the mean and the standard deviation of these values. The formulae for the mean and the standard deviation were given in the instructions in statistical notation.

Test problem. This program was to calculate the sum of a simple mathematical series (the harmonic series) for any given value of n. The task included all stages of programming, including debugging and testing the program. This was a slightly simplified version of the problem used by Allwood and Björhag (1990). The instructions to the problem are given in the Appendix and a possible correct solution to the slightly more difficult problem is given in Allwood and Björhag (1990). A correct solution to the present problem is about 38 lines long. All problem instructions were given in writing.

Procedure

All subjects were first given an overview of the session and were then asked to fill in a brief questionnaire about their programming experience. In the training session which followed next and lasted about one hour, the subjects in the experimental group were asked to read the debugging training instructions, described above.

After having read through the instructions carefully and having been given the opportunity to ask questions, subjects in the experimental condition were presented with the four training problems, one at a time. Each training program had planted bugs and the subjects were asked to retrieve the programs from a prepared file to the screen, compile and debug the program so that it functioned according to the description given in the problem instruction. The subjects were asked to debug the programs according to the debugging instructions just read. The experimenter intervened during the training session to remind the subject to debug according to the training instructions, should he or she forget to do so. The subjects in the experimental condition had free access to the debugging instructions during both the training phase and the test phase.

Subjects in the control condition were in the training session asked to create correct programs for each of the four training programs. In contrast to the task given to the experimental subjects, this task included writing the actual code from the beginning.

In the test phase, which lasted for two hours, all subjects were asked to create a correct version of the test problem described above. All subjects were given the same instructions except that the subjects in the experimental condition were asked to debug according to the training instructions. The instructions in both conditions emphasized that the subject should attempt to solve the problems as independently of the experimenter as possible. However, subjects were also told that the experimenter would ask them if they wanted help if they got stuck and sat passive for a few minutes. An analysis showed that there was no statistical significant difference between the conditions with respect to the number of times they were aided by the experimenter ($p > 0.42$).

All subjects were allowed free use of their text book in Pascal. The subjects in both conditions were asked to verbalize their thoughts aloud ("think-aloud") during the whole test phase. The subjects were also asked to report what commands they entered on the computer. The computer screen was videotaped and the subjects' think-aloud comments were tape-recorded during the test phase. Each subject was seen individually.

Results

The raw data used in the following analysis consisted of the video recordings made of the subjects' screen as they attempted to solve the programming task in the test phase of the experiment. The raw data also included the tape recordings of subjects' think-aloud reports and notes taken by the experimenter during the test phase. The data analyses presented here only relate to events after the subjects had attempted to compile the program for the first time. This means that any bugs debugged before the first compilation of the program are not included in the analyses presented below, all other bugs are included. However, previous research indicates that most of novices' debugging activities usually take place after the subject has attempted to compile the program for the first time.

Bugs made. Three types of programming errors were analyzed: syntax, semantic and logical errors. Syntax errors are defined as program code which does not follow the

grammatical form necessary for the running version of Pascal. Syntax errors also include instances where the subject omits code which is obligatory in the program, such as declarations of variables used in the program. Semantic errors are syntactically correct expressions which are impossible in the context they occur in. Finally, Logical errors are errors where the code does something different than is stated in the problem instruction, but the subject does not realize this.

Table 1 shows that the two conditions do not differ with respect to the total number of bugs made nor with respect to the number of bugs made when the three types of bugs analyzed are considered separately.

Table 1. Mean number of errors made in the two conditions, for four types of errors and for all errors.

Error	Condition		df	p-value
	Experimental	Control		
Syntax	5.24	5.50	19	0.90
Semantic	1.91	1.30	19	0.33
Logical	4.27	3.90	19	0.69
Spelling	0.45	0.40	19	0.88
All errors	11.91	11.10	19	0.74

Bugs eliminated. Table 2 shows the mean proportion corrected errors out of all errors made and out of all syntax, semantic and logical errors made in each of the two conditions. As can be seen in the table, there was a significant difference between the conditions with respect to the proportion of errors corrected by a subject out of all errors made by the same subject. The table also shows that the proportion corrected Semantic errors out of all Semantic errors made approached significance. There was no statistical significant difference between the groups with respect to the number of errors introduced after the first compilation of the program.

Table 2. Mean proportion corrected errors out of all syntax, semantic and logical errors made and out of all errors made for the two conditions.

Error	Condition		df	p-value
	Experimental	Control		
Syntax	0.99	0.95	19	0.36
Semantic	0.96	0.67	19	0.09
Logic	0.87	0.75	19	0.24
All errors	0.94	0.83	19	0.02

Debugging behaviour. The next analysis deals with possible differences between the two conditions with respect to the subjects' debugging behaviour. For this purpose the subjects' behaviour in those parts of the testing-phase where the subjects evaluated their program in a negative way was coded with the following categories.

1) Interprets Meaning of Error Message. The subject interprets the meaning of an error message; for example, the subject rephrases the content of the error message.
2) Follows Flow of Information in the Program. The subject mentally "simulates" or "interprets" some part of the program by trying to imagine the flow of information in that part.
3) Error Hypothesis Localized to Construction. The subject hypothesizes about what might be wrong with the program. The hypothesis is limited to the mention of a specific construction, i.e. a FOR-loop or a part of the program carrying out a specific function such as controlling for overflow.
4) Exact Error Hypothesis. The subject's hypothesis about the error specifies the exact code which is thought to be buggy.
5) Compiles. The subject compiles the program.
6) Tests. The subject enters a value into the program and observes the program's response.
7) Enters program on screen. The subject enters the program on the screen.
8) General Dissatisfaction. The subject expresses general feelings of helplessness and dissatisfaction with respect to the program.
9) Enters evaluative phase. The subject stops producing code for a new section of the program and starts to evaluate previously produced code.

Statistical analyses (using t-tests) showed no statistically significant differences between the two conditions with respect to any of the categories investigated.

Discussion

The present study involves an attempt to improve subjects' debugging of a program created by themselves by giving them training in debugging. The main result of the study was that the experimental group after one hour of training in debugging eliminated a significantly larger proportion of their bugs compared to a control group which had received no such training. There was no significant difference between the conditions with respect to the number of bugs made. This suggests that the groups were equivalent with respect to their knowledge of Pascal.

The result that debugging of own produced programs can be improved after training together with the fact that novices spend much time debugging their programs suggests the desirability of introducing debugging training into the curriculum in programming courses.

An important question in connection with training in debugging concerns which components of the training were effective and which were not. Unfortunately we were not able to locate any systematic differences between the two conditions with respect to debugging behaviour. The lack of differences in this respect between the two conditions in our study possibly suggests that our debugging training did not serve to increase the general level of activity of subjects in the experimental condition.

For some of the categories investigated, the fact that subjects in both groups were asked to verbalize all thoughts aloud may have contributed to the lack of differences

found between the two groups in specific debugging behaviour. For example, the think-aloud instructions may have encouraged the subjects in the control condition to become more explicit about the meaning of error messages, just as was encouraged in the debugging instructions. Furthermore, our analyses only dealt with the number of times a specific category occurred; it may be that there are differences between the conditions with respect to the order in which and quality of how the debugging behaviours were carried out.

A similarity between the debugging training given in the present study and in the study of Carver and Risinger (1987) is that it encourages subjects to be systematic in their debugging behaviour. It may be that the beneficial effects of the training to a large extent were located on this general level.

In order to develop more efficient debugging training programs it is important to find out exactly what training components will produce effective debugging behaviour. In this connection we assume that it will not always prove efficient to attempt to make novices copy the debugging behaviour of experts. For example, Nanja and Cook (1987) found that the experts in their study often corrected many bugs at the same time before verifying the effects of their corrections. In contrast, the novices in the same study tended to correct one bug at a time. When the novices in the study by Allwood and Björhag (1990) attempted to correct many bugs at the same time they occasionally got into difficulties. It seems likely that the greater processing capacity available to experts in connection with programming allows them to successfully use debugging strategies which are too resource demanding for novices. Further research is needed to make clear what training components result in efficient debugging behaviour.

References

Adelson, B., 1984. When novices surpass experts: the difficulty of a task may increase with expertise. Journal of Experimental Psychology: Learning, Memory and Cognition 10, 483-495.

Allwood, C.M. and C.-G. Björhag, 1990. Novices' debugging when programming in Pascal. International Journal of Man-Machine Studies 33 (In press).

Allwood, C.M., 1986. Novices on the computer: a review of the literature. International Journal of Man-Machine Studies 25, 633-658.

Carver, S.M. and S.C. Risinger, 1987. Improving children's debugging skills. In: G. Olson, S. Sheppard and E. Soloway (eds.), Empirical Studies of Programmers: Second Workshop. Norwood, NJ: Ablex. pp. 147-171.

Gilmore, D.J. and H.T. Smith, 1984. An investigation of the utility of flowcharts during computer program debugging. International Journal of Man-Machine Studies 20, 357-372.

Gugerty, L. and G.M. Olson, 1986. Debugging by skilled and novice programmers. In: M. Mantei and P. Orbeton (eds.), Proceedings CHI' 86: Human factors in computing systems. Boston: ACM. pp. 171-174.

Katz, I.R. and J.R. Anderson, 1988. Debugging: An analysis of bug-location strategies. Human-Computer Interaction 3, 351-399.

Miller, L.A. 1974. Programming by non-programmers. International Journal of Man-Machine Studies 6, 237-260.

Myers, G.J., 1978. A controlled experiment in program testing and code walkthroughs/inspections. Communications of the ACM 21, 760-768.

Nanja, M. and C.R. Cook, 1987. An analysis of the on-line debugging process. In: G. Olson, S. Sheppard, and E. Soloway (eds.), Empirical Studies of Programmers: Second Workshop. Norwood, NJ: Ablex. pp. 172 - 184.

Vessey, I., 1985. Expertise in debugging computer programs: A process analysis. International Journal of Man-Machine Studies 23, 459-494.

Wiedenbeck, S., 1985. Novice/expert differences in programming skills. International Journal of Man-Machine Studies 23, 383-390.

Appendix

Programming task given to subjects.

Harmonic series

The task is to write a PASCAL program which calculates the sum H of the so called harmonic series.

$H = 1 + 1/2 + 1/3 + ... + 1/n$, where $n > = 1$

The obvious way to calculate this sum is to add the n terms involved. However, for large values of n this way of doing the calculations will be very time-consuming, even for a fast computer. For large n-values, the sum "H-approx." can be used as an approximation. The formula for this is:

$H\text{-approx.} = \ln n + Y + 1/(2n) - 1/(12n^2) + 1/(120n^4)$,

where the constant Y = 0.5772 1566 4901 5329 (rounded to 16 decimals).

When $n > 200$ the difference between the two formulas is less than 10^{-16} (i.e. 16 decimals, which is the approximate limit for the computer's precision using "longreal"). Thus, for $n > 200$ the approximation formula should be used.

The numerically largest number, "MAXREAL," which can be represented in the computer (when using the compiler relevant here) is approximately 10^{77} and the largest "MAXINTEGER" is 10^9. If one tries to perform a computation which gives a result larger than this figure one will get a so called "arithmetic overflow" and the program run will be terminated.

Overflow can be avoided by excluding one or more terms in "H-approx." for critical values of n. However, one has to make sure that the excluded terms are so small that they are smaller than the precision of the computer. This can be done by a rough estimation, and does not need to be exact, since the limit for overflow is at 77 figures (compared to 16 decimals which is the precision of the computer).

* User instructions should be kept to a minimum (For example, "Give a positive integer").
* The program shall compute and print the sum for the value of n which is read in. ("The sum is:=" is sufficient as a print out and the decimals should not be limited).
* The program shall ask for and read values of n from the terminal an arbitrary number of times.
* If the n-value 0 is given, the program should interpret this as the end of the indata and terminate the run.

Approaches to Error Analysis in Human-Computer Interaction

Paul A. Booth & John Gray

Department of Computing, Faculty of Science and Engineering,
Manchester Polytechnic, John Dalton Building, Chester Street, Manchester, M1 5GD, UK.

This paper concentrates on errors in understanding (the kind that users make when learning to use a system) as opposed to slips or lapses (the kind that experts make when dealing with a situation or task). It is suggested that present schemes for dealing with misunderstandings, or representational mismatches, are inadequate. Many schemes are incomplete or centred around describing just one system, while the ECM scheme, although comprehensive and system independent, produces a piecemeal and theory-less analysis.

The objective was to assess whether two of the theories embodied within HCI grammars, GOMS and TAG, might be of use in interpreting and understanding representational mismatches. To this end, three problems were taken and analysed using ECM, GOMS and TAG. The results of these analyses demonstrated that GOMS and TAG were more difficult to apply as error classification schemes, because their categories could be decomposed past a level that was of use. More importantly, however, their theoretical constructs did not appear to allow any greater understanding of user misunderstandings. It is concluded that current HCI theories are too restricted, and that an appreciation of how users develop their representations of machines is unlikely to be achieved without everyday global theories of cognition.

Introduction

Learning to use computer systems is sometimes viewed as being trivial. Too often there is an artificial distinction drawn between the first few hours or days when a computer system is learnt, and later use when it is falsely supposed that no learning occurs. Of course learning continues as users explore the system, discover shortcuts and find out how to avoid problem areas. The perspective where learning and use are considered as separate is misplaced. For example, users who learn to deal with a computer system, but, nevertheless, develop representations of that system that are inaccurate in some respect, may subsequently act in ways that create problems, particularly when problems 'snowball' (Janosky et al., 1986). In the context of an office system such users may unintentionally destroy, damage or disrupt either their own work or the work of others. One example of such an occurrence is where a person using a spreadsheet application on an Apple Macintosh. This person had used this system for over one year, and so had a representation that was, in terms of his everyday interaction with the system, quite adequate. The system produced a message informing the user that there had been an error when attempting to save the file to disk. The user quit the application and re-initialized the

disk, despite having to use the 'erase disk' command. This, of course, destroyed all of the work on the disk, although the user believed that this would just 'tidy' his work and do no harm at all. This was because he had been informed that initializing the disks 'draws the lines to park the cars, or blocks of data.' Consequently, by re-drawing the lines he would ensure that in future his file was properly 'parked' or saved.

In the context of process control systems, the consequences can be even more serious. The Three-Mile Island accident is a case in point, where plant operators had an inadequate representation of a valve and spent some considerable time attempting to perform an operation that was impossible. The substance of the argument is that users representations of systems are frequently inadequate, and that these unsatisfactory representations lead directly to unhelpful and even dangerous actions.

Given the importance of users developing adequate representations of the technology they use, how can we reach acceptable judgements as to whether systems encourage the development of appropriate or inappropriate representations? One answer is to look at the errors that occur during interaction, particularly during the early stages of learning and using a system. These *errors* might be more appropriately called *dialogue failures* (see Booth, 1990c; Lewis & Norman, 1986). For the moment, however, we will use the more common, if less desirable term, *errors*.

It is sometimes useful to think of errors as being of two types: Firstly, the kind that Norman (1983a) describes with respect to slips and lapses. These are the sorts of errors that people make when they have a good understanding of a sequence of actions. Examples can be found almost anywhere in everyday life, whether it is when we begin to drive along a commonly used road when we meant to take another route, or whether we accidentally make coffee when we intended to make tea. These sorts of errors have been considered in detail, and very interesting and plausible explanations for their occurrence have been advanced (i.e. Rasmussen, 1988; Reason, 1990; 1986; Reason & Mycielska, 1982).

Our interests, however, are in the second type. These are errors in understanding. Such errors have been referred to in Norman's (1983a) scheme as *mistakes*. Mistakes are where there is an error in the formation of an intention. For example, a person might intentionally switch on a boiler to heat water, without realizing that the boiler is empty and has to be filled prior to heating. These errors are *misunderstandings* or *mismatches* in the representations of the task adopted by the user and the system. This, however, is not meant to imply that users have adequate, complete or consistent representations. On the other hand, there is no reason to assume that the representation *implicit* within a piece of software is complete, consistent or reasonable. Nevertheless, labeling such errors as mistakes, however, does not tell us a great deal about how they arise, nor how to identify systems that might encourage such misunderstandings.

Breaking down errors

Having identified our area of concern, there is now the question of how to analyse these errors in understanding. Generally, the distinction between the two types of error

outlined above, those that arise out of automaticity where people who understand a task well and slip or lapse into some other routine, and those that occur when people do not fully understand a machine, has not been made. Misunderstandings and slips and lapses have tended to be categorized under the catch-all heading of *errors*, instead of being distinguished in any fundamental way according to their origins. This may account partly for why the question of misunderstandings, *representational mismatches* or *user-system errors* as they have been previously called (Booth, 1990a & c), has tended to be neglected, although there has been some work in the field of programming (e.g. Brna et al., 1987).

Norman (1983a) suggests that misunderstandings can be thought of as either *mode* errors or *description* errors. Other schemes have been suggested based around syntactic and semantic distinctions (Davis 1983a & b), particularly work aimed at understanding programmers' representational mismatches (Youngs, 1974). Many schemes, however, have tended to consist of *ad hoc* categories that reflect the structure of the system being analysed (e.g. Davis, 1983a & b; Welty, 1985), and although such classifications may have helped in understanding a particular system, it is difficult to see how these schemes have advanced our understanding of representational mismatches. Moreover, those schemes that do not reflect any particular system structure are rarely comprehensive. For example, not all misunderstandings can be dealt with under Norman's *mode* and *description* error distinctions (Booth et al., 1989).

The ECM scheme for classifying errors (Booth et al., 1989; Booth 1990a & c) has certain advantages that can be taken to distinguish it from previous error classification schemes, amongst which are that it is user-oriented; it is comprehensive; it enables and directs creative thinking; and it identifies specific points in the system that are at odds with the user's perspective. However, its two fundamental problems are that it lacks any underlying theory to explain representational mismatches, and that, as a consequence, its analysis tends to be piecemeal. One additional practical problem with the ECM scheme is that it is sometimes possible, in a minority of cases, to argue for more than one classification of a single mismatch. It is the absence of a theoretical framework, however, that appears to be the major obstacle in the path of further understanding representational mismatches.

Consequently, the aim in this paper is to ascertain whether some of the theoretical frameworks already present within human-computer interaction, as embodied in some of the HCI grammars, might be of use in providing a theoretical basis for understanding representational mismatches. Consequently, we have taken a number of problem descriptions and analysed them using the ECM scheme and two other schemes: the Goal-Operators-Methods-Selection Rules (GOMS) scheme (Card et al., 1980; 1983) and Payne & Green's (1986) Task-Action Grammar (TAG). The constructs of these grammars are used to classify misunderstandings in order to assess whether they might provide an overall theoretical framework in which these errors in understanding might be fruitfully

considered. First, however, we need to briefly describe ECM, as well as a little of the background of both GOMS and TAG.

The ECM scheme

The approach adopted within ECM is one where classification is based upon the user's perspective. What is *right* and *wrong* in the system relies entirely on the user's view of the system and task. The four possible classes of mismatch following the identification of a dialogue failure can be seen in figure 1. The classification process entails four stages through which the proposed classification process should proceed (see figure 2).

	Object	Operation
Concept	object-concept mismatch	operation-concept mismatch
Symbol	object-symbol mismatch	operation-symbol mismatch

Figure 1: The four classes into which a mismatched element should fit

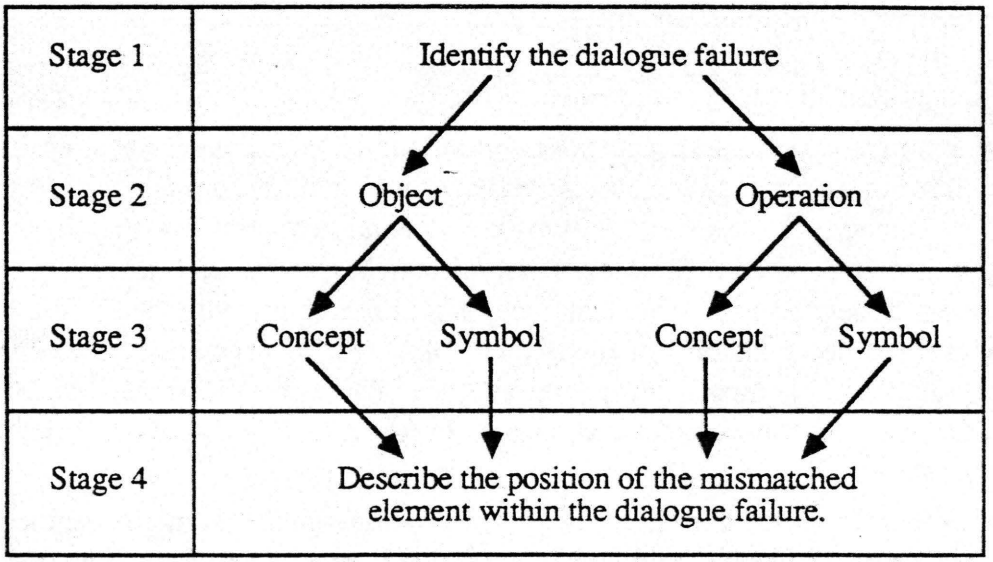


Figure 2: The process of classification

Stage 1: Identifying a Dialogue Failure. The first stage of the classification process involves the identification of a representational mismatch or user-system error. In practical terms, this means the detection of a failure in the dialogue between the system and the user. *Stage 2: Identifying an Object or Operation.* Having identified a dialogue failure then the next stage is to identify the object or operation associated with the failure. *Objects:* A data file in a system, an applications package, a figure, character or window on a screen might all be considered to be objects. *Operations:* An operation *is an action*

which is performed upon an object or objects within the system. Saving a file, deleting a character, changing a shape in a graphics package, creating text in a word processing package are all operations. A general rule of thumb for distinguishing objects from operations is that an operation is something that is done to an object whereas an object has operations performed upon it.

Stage 3: Identifying the Mismatch Type. Once the dialogue failure and the associate (an object or operation) of the dialogue failure has been identified then the next stage is to classify it in terms of its cause. To do this, either the concept or the symbol of the object or operation is identified as being at fault. *Concepts: A concept may be either an object or operation whether represented mentally (the user) or computationally.* A concept mismatch is *a fundamental difference in the understanding and representation of system or task objects or operations. Symbols:* The term *symbol* is taken to mean *a word, character, sign, figure, shape or icon employed by either the user or the system to represent an object or operation within the system.* A symbol mismatch is not one of fundamental understanding, *but occurs where the computer and the user adopt different terms or visual images to represent the same concept.* A rule of thumb for distinguishing symbol from concept mismatches might be whether the user's symbols for the systems objects and operations easily map onto the designer's version of the task or system. *Stage 4: Positioning a Mismatched Element.* Once a dialogue failure has been identified, and its causes classified, these causes (or mismatched elements) need to be described with respect to their position within (or contribution towards) the dialogue failure. The crucial question that is addressed at this stage in the classification process is; *what role did the element play in the dialogue failure?* The important point about distinguishing mismatches is that to properly classify a dialogue failure the observer must not only discover what happened (i.e. which operations were applied to which objects) but must also elicit the user's view of the task and system. The physical actions and consequences may remain constant, but the classification depends upon the user's view of the task and system.

GOMS and TAG

Card et al.'s (1983) approximate theory requires a task to be decomposed into its goals and subgoals, the operations required to achieve these goals, different methods for achieving goals, and rules for choosing between alternative methods. This scheme can be used to predict routes through tasks, as well as the time to complete a task. Although there is some evidence that users do not make the rational choices between alternative methods that GOMS assumes (MacLean et al., 1985), the attraction of this sort of production-rule theory is such that it has been used as the basis for more advanced methods (i.e. Kieras & Polson, 1985). Nevertheless, the drawbacks of GOMS are that it assumes expert and error-free performance, while providing limited information (i.e. time to complete a task and the route through the task).

Payne & Green's (1986) TAG is based on notation very similar to Backus Naur Form (see Reisner, 1981, for a discussion of the potential application of BNF to HCI).

The central notion of TAG is that we, as users, have a library of cognitively automated *simple tasks* that we perform routinely. We can combine these simple tasks in various ways to achieve different effects. The ordering of these simple tasks is determined, in Payne & Green's theory, according to rule schemata (see Booth, 1989; Payne & Green, 1986, for an explanation). To analyse a system using TAG we first identify the system's concepts and then simple tasks. The number of different orderings of these simple tasks, for the different concepts, is represented in the number of rules required to describe the system. Essentially, the more rules that are required then the more complex, and possibly less consistent, the system. TAG, then, provides a measure of complexity. However, although TAG has certain theoretically attractive features (i.e. such as the way in which errors can be explained) it is considered to be difficult to learn, and gives small system inconsistencies as much prominence as large inconsistencies (Green, 1987).

Although a number of researchers have identified both theoretical and practical drawbacks with these methods (Bellotti, 1988; Booth, 1990b; Briggs, 1988; Carroll & Rosson, 1985; Green, 1987; Knowles, 1988) there is no *a priori* reason why the theoretical constructs and terminology provided by these methods might not be used to fruitfully classify and understand errors. Indeed, using these theories in this way (i.e. classifying errors rather than analysing tasks) overcomes some of the theoretical and practical objections that have been previously raised.

Representational mismatches

Here, some of the descriptions of misunderstandings provided by Steve Jones (1989) have been used. We will first analyse them using ECM, and then using GOMS and TAG. Although ECM has a number of drawbacks, as mentioned earlier, it nevertheless provides a baseline against which we can judge the usefulness of GOMS and TAG. Firstly, however, we need to outline some of the background to the observations made by Jones (1989).

Jones asked a number of users, ranging from people who had no computing experience to some who had several years experience of using IBM PCs, to use the Apple Macintosh PC and its manual. Jones observed, and to some extent interpreted, users' actions and the misunderstandings that occurred. Indeed, it is the interpretative aspect of Jones's descriptions that makes them so interesting and useful. We will examine just three of his problem descriptions.

Problem descriptions

Problem 1. In the Macintosh desktop system certain icons (i.e. folders or disks) can be opened to form windows on the screen (see figure 3). The user, however, did not understand the relationship between icons and windows. Nor did this user understand what was meant in the manual when it stated that an icon becomes *hollow*. For those who do not have any experience of the Apple Macintosh, it may be useful to describe this phenomenon. An icon may represent a folder or disk which contains files, applications,

etc. When this icon is opened (usually by placing the mouse pointed over the icon and pressing the mouse button twice) a window appears (see figure 3). The icon, however, remains where it is, but turns from being generally white to a darker shade of grey. It is in this state that the icon is considered to be *hollow*. This hollow icon now has new properties. It cannot be opened of course, because it is already open. If, however, the open operation is performed on it (i.e. the mouse pointed is placed over the icon and the mouse button pressed twice) this will have the effect of bringing the icon's window to the front of any other windows that are open on the desktop.

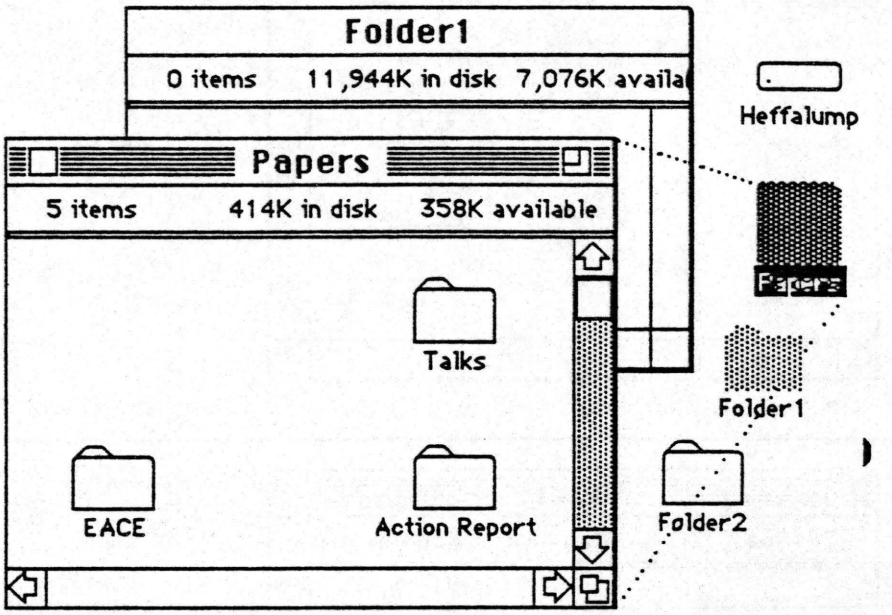


Figure 3: A diagram showing a hollow disk icon and its window (the dotted lines have been drawn in to indicate the relationship), a hollow folder and its window and a second folder that is closed

Problem 2. When a disk is ejected from the computer its icon, and the icons of any of its files on the desktop, do not disappear, but become hollow (see figure 4). In this situation, however, if there is an attempt to open any of the icons this will not immediately cause them to either open nor their windows to come to the front, but will cause the system to ask for the disk to be re-inserted. A user complained that he thought the icons should disappear when the disk was ejected. To get rid of the icons they have to be placed in the wastepaper icon (see figure 4). It is actually possible to eject a disk and remove its icons in one action, and this involves placing the disk icon directly into the wastepaper icon. Many users see this as counter-intuitive, however, and fear that their disk might be erased.

Problem 3. This final problem is related to the operations the Macintosh will perform when changing between applications. As the Macintosh loads an application (actually the wordprocessing application MacWrite in this case) it requires the user to swap a number of disks. The Macintosh will read data from one disk, eject it, and ask for the other disk to be inserted. After reading data from this one it then ejects this disk and asks

for the previous disk to be inserted. This cycle can continue for as many as seventeen or more swaps. Most of Jones's subjects believed that there was something wrong, and that they had acted incorrectly in some way. Subjects did not realize that if they continued to swap the disks the application would eventually appear for them to use. Many of these subjects abandoned the session and gave up when confronted with what they perceived as an insurmountable problem.

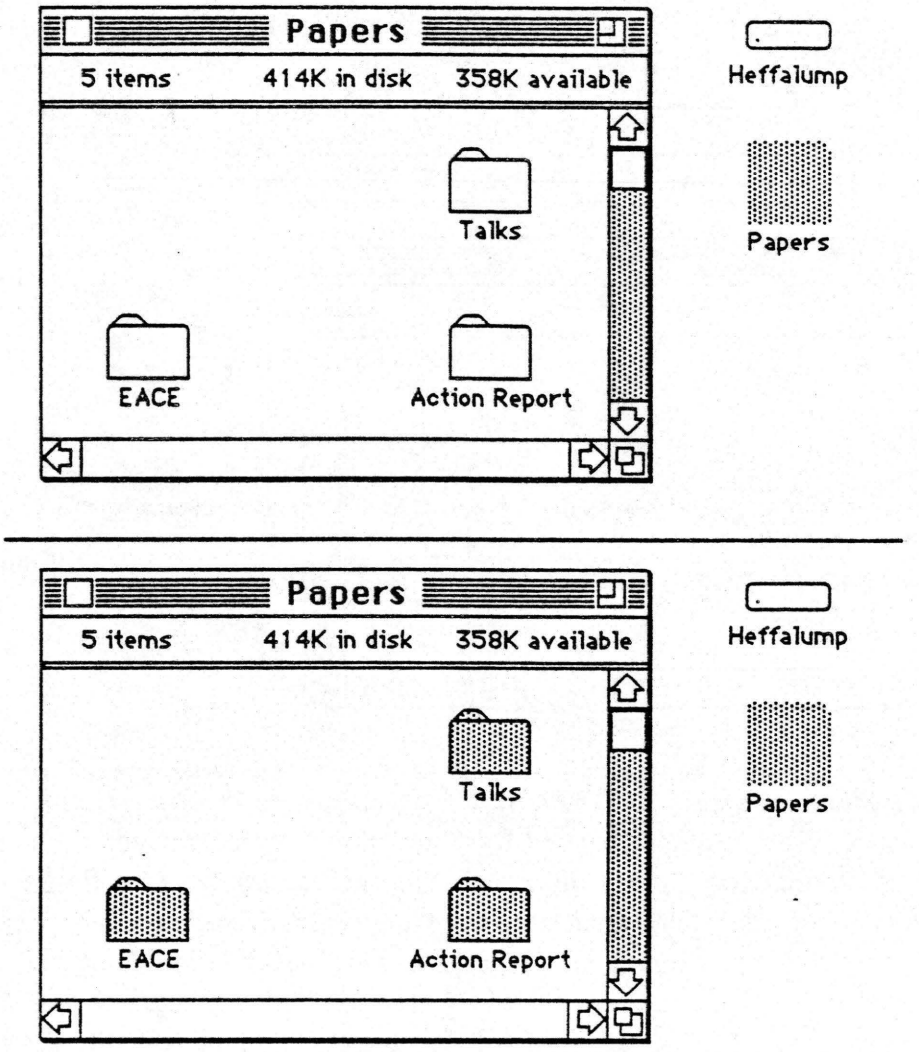


Figure 4: A diagram showing the hollow disk item and its window before ejection, and after ejection. Notice that the folders in the disk window become hollow after the disk has been ejected

Analyses

ECM

Problem 1. This first problem, to recapitulate, was where the user could not understand how an icon could open to become a window. This object, this folder or disk icon that can open into a window, has properties that the user did not expect. It is the object that is mismatched, and it appears to be the concept of the object that is at fault from

the user's perspective. Either the system needs to be changed in a way that is more commensurate with users' expectations or the user needs to be informed in some way of the nature and properties of icons. This, however, is a design question. As we are concerned with only analysing and understanding these problems rather than solving them, we will not pursue these redesign issues.

Problem 2. This second problem was where the icons associated with a disk did not disappear when the disk was ejected, contrary to the user's expectations and wishes. We could argue that this is clearly a problem with the operation of the system, as the system does not operate as the user expects, and that it is the concept of the operation that is mismatched with the user's representation. This, then, is an operation-concept mismatch. On the other hand, as it is possible to eject a disk and remove its icons from the desktop by placing the icon in the wastepaper basket (i.e. as the operation the user wants is possible) then we might characterize this as an operation-symbol mismatch, as the operation the user requires is present, but not in a form the user recognizes.

The practical problems of agreeing classifications using ECM, evident here and mentioned earlier, are typical of the dilemmas that faced the design team who used ECM to analyse problems and guide their redesign discussions (see Booth et al., 1989). It is possible to argue for more than one classification of a problem depending upon the perspective that is adopted when considering the problem. Although such problems tend to occur in a minority of cases, they are, nevertheless, significant.

Problem 3. In the final problem, where the Macintosh required users to swap disks a large number of times and users thought that had made some mistake, we can classify this as an operation-concept mismatch. This is because the notion of the operation that is being performed, that of loading the data needed for the application in blocks from different disks, is not shared by the user and the system.

GOMS

In attempting to classify the problems in accordance with structure of the GOMS scheme we will attempt to slot representational mismatches into the four central constructs of Card et al.'s (1983) approximate theory. To do this we will identify whether it was a goal that was mismatched between the representations held by the user and the system; whether it was an operator that was mismatched; whether it was a method (series of operators) that was mismatched; or a selection rule that was not mutually shared.

Problem 1. In the case of the first problem, where icons open to become windows and the user did not understand this, we might argue that it is a goal mismatch. This is because the user did not want objects on the screen to open into windows. The problem is, of course, *the goals at which level?* Overall goals of producing a piece of text, etc., are most probably well-matched. Consequently, we might argue that it is not a goal mismatch, but the way in which the goal is achieved. It is the operator or method that is mismatched? It appears to be difficult to decide which of these is incorrect from the user's perspective.

The only thing we can be clear about is that it probably is not a disagreement with respect to a selection rule.

Problem 2. For the second problem, where icons remain on the screen and become hollow when a disk is ejected, instead of disappearing, we could suggest that it is an operator mismatch. This is because the operator within the system was not what the user expected (of course, it might argued, at this point, that the terms operator and operation are being used as synonyms). Again, however, we could argue that this is a goal mismatch, as the goal stack the user and the system hold at this particular point are not identical, as one party (the user) has a goal to eject the disk and all its icons, and the other (the system) has a goal to eject the disk but retain some information regarding the disk.

Problem 3. In this 'disk swap problem', where users mistakenly believed that they had acted incorrectly in some way, we could characterize this as a goal mismatch because the subjects were unaware of the system goal. Once again, however, it is possible to suggest that this is a method or operator mismatch, as the goals were the same (to open the MacWrite application), but the expected methods were not agreed.

It is possible to argue that any mismatch is a goal mismatch or an operator mismatch, depending on the perspective we adopt. Moreover, if we decide that goals are agreed, then there is still a problem with the question of whether it is an operator or method that is at fault, given that a method is a sequence of operators and an operator, depending upon the grain of analysis, can be broken down into further operators. As a result, it seems as though a decompositional system such as GOMS, with goals and subgoals, and operators that can, on analysis, be broken into further operators and called a method, cannot work as a classification scheme.

Overall. We can say two things about this analysis. The most obvious is that a categorization of representational mismatches using GOMS is particularly difficult. It seems impossible to argue for a classification of any problem using this scheme that is not equivocal in some respect. The second point is that even if we agree a classification, deciding that a mismatch is a goal, operator or method mismatch does not seem to provide us with further useful information about the mismatch. With respect to errors then, the GOMS approximate theory appears to be of little use; it allows us no greater understanding of representational mismatches than ECM, and is more difficult to apply. Nevertheless, to be fair to Card, Moran and Newell, it is possibly worth mentioning that their production system theory was never intended for the analysis of errors.

TAG

Payne & Green's (1986) Task-Action Grammar has constructs that are similar to ECM. For example, concepts and symbols are identified. TAG, however, goes further and distinguishes between simple tasks and rule schemata. During the analysis of the three problem descriptions mismatches will be assigned, not only to particular concepts and symbols, but also to these latter categories.

Problem 1. In the icons and windows problem (see earlier description) there

appears to be a concept mismatch, as it is the concept of the relationship between the icon and its window that is unclear to the user. However, to take this classification further using the constructs provided by TAG, and hopefully gain a greater understanding of it, we can identify the problem as being with either a simple task or a rule schema. In this case we could suggest that it is a simple task, as the opening from an icon to a window is one of the systems basic operations, and does not involve the ordering, for the user, of a number of operations.

Problem 2. The second problem, with icons not disappearing when a disk was ejected, serves to demonstrate that TAG is likely to run into the same practical problems as ECM—that a problem can sometimes be classified in more than one way depending upon the perspective adopted by the classifier. In this case it is not clear as to whether it is a symbol or a concept that is mismatched (see earlier ECM analysis). If we decide, for the sake of argument, that it is a concept problem, then we could classify it as a problem with the rule schemata. This is because there are a number of simple tasks performed here. One simple task is selecting the disk to be ejected, and the other selecting the 'eject' command.

Alternatively, we might suggest that the problem is with the simple task, in that the simple task to eject a disk is not mutually agreed, in that the icons are left on the screen. Once again, we might retort that the simple task to eject a disk can be broken down into two simple tasks: one to eject the disk from the machine and the other to remove the icons (as this is possible if the disk icon is placed directly in the wastepaper basket icon.) Consequently, it is a problem with the rule schema in ordering the appropriate simple tasks. Of course, we could pursue this argument further, but it is doubtful whether much of use might be gained from it.

Problem 3. The final 'disk swap' problem can be characterized as a concept mismatch. To recapitulate, this is because the notion of the operation that is being performed, that of loading the data needed for the application in blocks from different disks, is not shared by the user and the system. With respect to TAG, the simple task of swapping a disk seems to be shared by both the user and the system. However, there is no understanding on the part of the user as to the need, on the part of the machine, for a large number of disk swaps. Consequently, we might argue that the rule schemata are mismatched. On the other hand, we might suggest that the notion of swapping disks is shared , but the need for repeating this operation is not understood, and so it is not a problem with either a simple task or a rule schema.

Overall. Given that it is possible to argue that some simple tasks can be decomposed into smaller simple tasks, TAG appears to be a decompositional scheme in the same way that GOMS is. In other words, whether a problem is classified as a simple task or rule schema mismatch depends upon the extent to which the problem has been broken down.

More important, however, is the question of whether the process of classification using TAG really advances our understanding of representational mismatches. Ironically,

thinking of slips and lapses in terms of simple tasks and rule schemata that organize this simple tasks has definite attractions: TAG's theory seems to be better at explaining the errors that an expert might make (i.e. lapsing from one routine into another) than those that a user learning a system is likely to experience.

Discussion

It might be possible to argue that the 'shoehorning' of problems into categories provided by theories that were never intended for this sort of use, does not show us a great deal that is useful. What is apparent from this exercise is that it is more difficult to classify the problems using TAG or GOMS than it is using the ECM scheme. One explanation of this could be that ECM has definitions specifically geared towards classification, whereas TAG and GOMS do not. Consequently, we might expect classification using ECM to be easier.

Another explanation is that both TAG and GOMS have problems in that their categories can be decomposed to a point where analysis is no longer useful. For example, goals can be divided up into subgoals and yet more subgoals, and operators divided up into smaller operators. Likewise, TAG's simple tasks can be divided up into smaller simple tasks. ECM does not appear to possess this problem. Although it is possible to decompose concepts into smaller concepts, etc., in reality this has not proved to be a problem.

Nevertheless, ECM still has the practical problem of multiple classification; where it is possible to classify a mismatch in more than one way depending upon the perspective that is adopted. In essence then, although ECM is not afflicted by the decompositional problems evident when using GOMS or TAG, it has other drawbacks that have similar practical consequences.

The need for theory

Although the practical difficulties in fruitfully applying the HCI grammars as classification schemes are of interest, the central issue of concern here is the question of theory. The objective of the exercise described in this paper was to assess whether either GOMS or TAG might provide a theoretical framework for understanding representational mismatches; the kind of errors that occur when users are learning to understand and use a machine. The results of this qualitative analysis suggest that these two schemes, GOMS and TAG, do not allow us any greater understanding of representational mismatches.

The central problem seems to be that the limited theories present within these two methods cover such small areas of concern. For example, GOMS can only predict routes and times through tasks for expert performance, while TAG is aimed at explaining how we convert or map our intentions into actions. However important these areas might be, it seems probable that the kinds of misunderstandings that users experience when learning to operate a new device are only likely to be explained by global theories which have

something to say about the everyday nature of large aspects of cognition. But how can we justify such a claim?

To account for misunderstandings we need to know something about how users form their representations of systems (of computers generally, or any machine or device for that matter). We know that user's representations have been characterized as 'unstable, parsimonious, confusable and incomplete' (Norman, 1983b), but this is just a list of attributes, it still provides no useful way of conceptualizing why our representations are, by nature, like this. A global theory, or framework as Reason (1990) calls it, would be able to provide a conceptual structure for answering broader questions that relate to the way users learn and sometimes misconstrue systems. For example:

- How can we act so quickly in so many very different situations, some of which we have not encountered before?
- How do we generate novel solutions, hypotheses and ideas?
- Why are we so quick to formulate hypotheses and so slow to relinquish them?
- Why are metaphors and analogies so useful, important and sometimes problematic?
- Why is context so important?
- What is the role of intuition?
- How do we learn, and build our representations of machines or any sort?

This is by no means intended as a balanced or comprehensive list, but provides a feel for the kinds of issues a global or general theory or framework would need to address if we are to understand everyday behaviour such as understanding and misunderstanding machines.

Conclusion

In terms of classifying representational mismatches, both GOMS and TAG have practical problems. More importantly, it has been demonstrated that their theoretical constructs do not shed any greater light on the misunderstandings that occur between users and systems. This was because they dealt with relatively small areas of theoretical concern, whereas the process of building a representation of a machine appears to touch upon a wide variety of issues. Consequently, it has been suggested that we need general or global theories of everyday cognition if we are to gain a better understanding of the problems of users learning to use machines.

Currently, there are no theories of this nature that seem to adequately explain users' understandings, misunderstandings and misinterpretations of computer systems. Although a small number of global theories have been suggested (i.e. Anderson's (1983) ACT; Reason's (1990) Underspecification Theory), even these do not provide the kind of comprehensive explanations and conceptual schemes to understand anything more than one or two important issues.

It is not envisaged that scheme global theories need be exact in every detail, indeed it

seems highly unlikely that such frameworks could account for every aspect of the empirical literature while explaining everyday cognition. But why do we lack such theories? There appear to be two possible answers. The first is that our knowledge has not advanced to a level where we are capable of developing even first attempts at such theories, and the second is that we are not bold enough to attempt them (McCann, 1990).

Acknowledgements

We would like to thank Steve Jones for forwarding his excellent reports of users' misunderstandings of the Macintosh Finder and MacWrite wordprocessing software, and Gill Brown for her help and comments on an earlier draft.

References

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

Bellotti, V. (1988) Implications of current design practice for the use of HCI techniques. In: D. M. Jones & R. Winder (Eds.), *People and Computers IV, proceedings of the fourth conference on human-computer interaction.* Cambridge, UK: Cambridge University Press.

Booth, P. A. (1990a) ECM: A Scheme for Analysing User-System Errors In: D. Diaper et al. (Eds.), *Human-Computer Interaction - Interact '90: Proceedings of the third IFIP conference on human-computer interaction, Cambridge.* Amsterdam: North-Holland.

Booth, P. A. (1990b) Modelling the User: User-system errors and predictive grammars. In: G. A. Weir (Ed) *HCI and Complex Systems.* London, UK: Academic Press.

Booth, P. A. (1990c) Using Errors to Direct Design. *Knowledge-Based Systems, 3(2),* 67-76.

Booth, P. A. (1989) *An Introduction to Human-Computer Interaction.* Hove, Sussex: Lawrence Erlbaum Associates.

Booth, P. A., Wright, M. & Oakey, S. (1989) Improving the Human-Machine Interface: Analysing User-System Errors. European Space Research and Technology Centre workshop. *Human Factors Engineering: A task-orientated approach.* ESTEC, European Space Agency, Noordwijk, The Netherlands, November 1989.

Briggs, P. (1989) What we know and what we need to know: The user model versus the user's model in human-computer interaction. *Behaviour and Information Technology. 7 (4),* 431-442.

Brna, P., Bundy, A., Pain, H., & Lynch, L. (1987) Programming tools for prolog environments. In: Hallam & Mellish (Eds.), *Advances in Artificial Intelligence.* New York, USA: Wiley.

Card, S. K., Moran, T. P. & Newell, A. (1983) *The psychology of human-computer interaction.* Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates.

Card, S. K., Moran, T. P. & Newell, A. (1980) Computer text-editing: An information-processing analysis of a routine cognitive skill. *Cognitive Psychology, 12,* 32-74.

Carroll, J. M. & Rosson, M. B. (1985) Usability specifications as a tool in iterative development. In: H. R. Hartson (Ed.), *Advances in Human-Computer Interaction.* Norwood, New Jersey, USA: Ablex.

Davis, R. (1983a) User error or computer error? Observations on a statistics package. *International Journal of Man-Machine Studies, 19,* 359-376.

Davis, R. (1983b) Task analysis and user errors: a methodology for assessing interactions. *International Journal of Man-Machine Studies, 19,* 561-574.

Green, T. R. G. (1987) Limited theories as a framework for human-computer interaction. Invited address to the Austrian Computer Society's 6th Annual Interdisciplinary workshop. *Mental models and human-computer interaction.* Schårding, Austria, 9th-12th June, 1987.

Janosky, B., Smith, P. J., & Hildreth, C. (1986) Online library catalog systems: An analysis of user errors. *International Jouirnal of Man-Machine Studies, 25,* 573-592.

Jones, S. (1989) *Preliminary observations of Macintosh Plus users.* Notes forwarded by the author, Department of Computer Science, Stirling University.

Kieras, D. E. & Polson, P. G. (1985) An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22 (4),* 365-394.

Knowles, C. (1988) Can Cognitive Complexity Theory (CCT) produce an adequate measure of system usability? In: D. M. Jones & R. Winder (Eds.), *People and Computers IV, proceedings of the fourth conference on human-computer interaction.* Cambridge, UK: Cambridge University Press.

Lewis, C. & Norman, D. A. (1986) Designing for error. In: D. A. Norman & S. W. Draper (Eds.), *User centred system design: New perspectives on human-computer interaction.* Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates.

MacLean, A., Barnard, P. J. & Wilson, M. D. (1985) Evaluating the human interface of a data entry system: User choice and performance measures yield different tradeoff functions. In: P. Johnson & S. Cook (Eds.), *People and Computers: Designing the interface. Proceedings of the First Conference of the BCS HCI specialist group.* Cambridge, UK: Cambridge University Press.

McCann, G. (1990) Indiscretion as a virtue. *Times Higher Education Supplement, 15.6.90.*

Norman, D. A. (1983a) Design rules based on analyses of human error. *Communications of the ACM, 26 (4),* 254-258.

Norman, D. A. (1983b) Some observations on mental models. In: D. Gentner & A. L. Stevens (Eds.), *Mental Models.* Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates.

Payne, S. J. & Green, T. R. G. (1986) Task-action grammars: A model of the mental representation of task languages. *Human-computer interaction, 2,* 93-133.

Rasmussen, J. (1988) Human error mechanisms in complex work environments. *Reliability Engineering and System Safety, 22,* 155-167.

Reason, J. T. (1990) *Human Error.* Cambridge, UK: Cambridge University Press.

Reason, J. T. (1986) Cognitive underspecification: Its varieties and consequences. In: B. Baars (Ed.), *The psychology of error.* London, UK: Plenum.

Reason, J. & Mycielska, K. (1982) *Absent-minded? The psychology of mental lapses and everyday errors.* Englewood Cliffs, New Jersey, USA: Prentice-Hall.

Reisner, P. (1981) Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions on Software Engineering, SE7 (2),* 229-240.

Welty, C. (1985) Correcting user errors in SQL. *International Journal, of Man-Machine Studies, 22,* 463-477.

Youngs, E. A. (1974) Human errors in programming. *International Journal of Man-Machine Studies, 6,* 361-376.

Predictive Tasks to Build User Models

Lisa Neal
Aiken Computation Laboratory
Harvard University
Cambridge, Massachusetts 02138 USA
tel: 617-495-8848
fax: 617-495-9837
email: lisa@harvard.harvard.edu

Abstract

The design of systems should be based on adequate user models which reflect the cognitive styles and skills of users. We propose a user model which incorporates expertise and learning and decision-making styles. Predictive tasks are used to acquire information about the user to construct these user models. Specifically, we use a number of computer games as rich tasks which allow us to derive information about users. The task structure is described, and the results and system implications of this work are presented.

1. Introduction

There is abundant evidence that software is often designed so that it does not lend itself to successful use: the underlying model of the user is too limited and based on the subjective perception of the designer. The inadequacy of designer's models of users has led to difficulties with the use of and the rejection of systems [Dagwell and Weber 83]. System design should be based on user models which reflect the cognitive styles and skills of users. Hence, we formulate a multi-dimensional user model, of which two dimensions are related to learning and decision-making styles and additional dimensions are related to the plurality of expertises involved in the use of a system [Neal 87a].

Once the components of a user model are defined, information about the user must be acquired to construct representative user models. An approach to the acquisition of information about the user which allows the building of models prior to a user's interaction with a system is presented which uses simple, structured tasks to measure these models. Specifically, we use a number of computer games as rich tasks which allow us to derive information about users. The games we selected for our experimental research are discussed and our task structure and methodology are described. Our results show that the primary goal chosen by the player is a very reliable predictor for the learning strategy adopted by the player. Implications, conclusions, and future directions are presented.

2. Multi-Dimensional User Models

A user model can incorporate the aspects of a user that are relevant to the behavior of a system [Kass 87]; we determine what these characteristics are and how to acquire information about them. User models can characterize a user's knowledge, expertise,

cognitive styles and skills, capturing the aspects of the user that are relevant to a system and making explicit the user's role as part of a system [Neal 87b].

In Neal [87a], user models are extended to examine expertise as a multi-dimensional entity and to incorporate learning and decision-making strategies. Expertise is most often viewed as mono-dimensional and binary, and is related to the differences between novices and experts which affect their interactions with systems. The multi-dimensional view of expertise is required by the multiple types of expertise involved in the use of a system. For instance, three distinct types of expertise must be considered in the design of a language-based tool: computer operation expertise, programming expertise, and expertise in the programming language supported by the tool.

In Neal [87a], it was shown that significant differences in system use are linked to the user's decision-making strategy and learning strategy. Decision-making strategies range from risk-averse to risk-accepting. Learning strategies range from operational learning to comprehensive learning. Users with an operational strategy build a sparse model containing procedural rules, while those using a comprehensive strategy emphasize learning at a deeper conceptual level, acquiring a larger command set than necessary, and developing an understanding of a system and its features and their relationship.

3. Formulation of User Models

Once the components of a user model are defined, information about the user must be observed and measured in order to construct user models which provide accurate cognitive representations of the user. User models can be built through user, system, or observer's measures. Self-placement of users can be problematic as users are often not very accurately introspective and lack knowledge about themselves [Rich 83b]. For instance, in competitive tasks requiring skill, people tend to overestimate their abilities [Hink and Woods 87]. Information about a user can be acquired prior to the use of a system or through interactions with a system. In the former case, techniques such as questionnaires are currently used, for which there are problems with accuracy. In the latter case, typically, a stereotype or generic model is activated and refined.

Information about the user is typically not available with the immediacy necessary to impact the interface to a system. Establishing a model before a user's first interaction with a system eliminates the need for rapid adaptation of the model to the user [Kass and Finin 87] and allows the model to be used to guide personalization of a system to the needs of different users. In addition, user models which will be used for multiple applications must be built independently of a particular domain. Such models allow consistency of system response, function key bindings, and screen formats, and allow the model to condition systems to a user's pattern of use [Fried 82].

An approach to the acquisition of information about the user which allows the building of models prior to a user's interaction with a system is the use of predictive tasks. We faced the decision of whether to formulate or synthesize a set of structured, controlled tasks or use prefabricated tasks. We opted for the use of a number of highly diverse computer games since they are readily available.

The rationale for our decision is based on a number of reasons. Games "hold an inexplicable fascination for many people" [Rich 83a] and they often provide striking examples of highly motivating activities [Malone 80]. Games are well-defined environments which are useful for studying the nature and structure of complex problem-solving processes [Feigenbaum and Feldman 63]. Computer game playing involves similar operational tasks to those necessary for the use of the system which will incorporate the user model. The user is interacting with a computer, providing a context which is consistent with that of the eventual use of the user model, as opposed to other methods such as questionnaires. We find corroboration for the usefulness of games in establishing user models as far back as antiquity, as quoted in Malone [81]. In *"The Republic,"* Plato states, "In teaching children, train them by a kind of game, and you will be able to see more clearly the natural bent of each."

4. Selection and Descriptions of Computer Games

The Macintosh computer was selected for this study because of the wide range of games available on it, many of which incorporate extensive visual and auditory feedback, and because of the multiple entry mechanisms. A number of games were considered; role-playing games were avoided because they tend to be very time-consuming to play and because they often do not have a well-defined ending, and games involving specialized knowledge, such as chess and solitaire, were ruled out, as were games for which the interface presented significant problems.

The games we used involved fundamentally different skills and a diverse set of non-trivial tasks. In Neal [89], a characterization is presented which is based on the elements that a game can include and the skills or abilities that are needed to play a game, specifically: the built-in goal of the game; the degree of complexity; existence of competition; the amount and type of randomness; time and speed; the amount and type of feedback, including scores and visual and auditory effects; the help system; options; levels of difficulty; the amount and type of control needed to play; the amount and type of strategy needed to play; the discovery and understanding of information; and fantasy.

We briefly describe each of the games we used. In Mystery Box, the computer "hides" between two and eight balls, where the number of balls is selected by the player, in an 8×8 grid and the player tries to find the balls by firing rays into the grid. The behavior of the ray depends on the locations of the hidden balls. and the player tries to find the balls by firing rays into the grid. The behavior of the ray depends on the locations of the hidden balls. The player places balls on the squares in which a ball is thought to be hidden, and gets points for correctly placed balls while losing points for incorrectly placed ones.

ThinkAhead is a number game in which a player competes against the computer to obtain the highest score. It is played on an 8×8 board, of which the initial configuration contains a starting marker and 63 numbers ranging from -9 to 15. The first player, who is chosen randomly, makes a vertical move in the same column as the starting marker, and the second player makes a horizontal move in the same row as the box selected by the first player, and the game continues in this manner with each number selected at most

once. The score is the sum of the numbers in the boxes selected by a player. The game ends when there are no numbers left, or when a player has no possible moves, and the winner is the player with the highest score.

In Crystal Quest, the user moves a "ship" through space, accumulating points by gathering crystals and by firing at a wide variety of "nasties," all of whom have different appearances, sounds, and behaviors. "Smart bombs," which are accumulated along with the crystals on each screen, can be used to remove all the "nasties" on the screen The graphics are sophisticated and the sounds and behaviors of the "nasties" often quite humorous. There are multiple levels, which increase in difficulty due to the larger number of increasing more complex "nasties" and the resulting increase in difficulty of avoiding or shooting at them.

Tetris is a game involving eye-hand coordination and spatial reasoning, in which different arrangements of four blocks fall from the top of the screen and can be moved left or right, rotated, or dropped. Filled-in rows disappear, and the game continues until there is no space left at the top for shapes to fall. There are multiple levels at which a player can start, and each time a player fills in ten rows the level increases automatically, increasing the speed at which shapes fall. The score is based on the number of shapes that are placed, the height from which they are dropped, and the speed at which they are falling.

5. Experimental Participants and Description of Task Structure and Methodology

Twenty-five subjects participated in the experiment; the subjects were undergraduate and graduate students at Harvard University, non-computer professionals, and software engineers. An effort was made to have as wide a variety of subjects as possible, with the range based on computer and game-playing experience; subjects ranged from infrequent users of word processors to heavy computer users with over ten years of computer experience. Likewise, subjects ranged from infrequent players of computer games to people who play games for hours daily.

Subjects were given the task of playing four Macintosh games: Mystery Box, ThinkAhead, Crystal Quest, and Tetris. Subjects were told that they could play each game for as long as they wished, playing each game at least once. Subjects who had played one game for an hour were asked to go on to the next. In addition to being observed while engaged in the task, subjects were asked to think aloud about what they were doing and why they were doing it, what were their goals, what they were learning or understanding about the game, what made them feel they were progressing and/or made them feel successful, why they stopped playing and what they thought what would happen if they continued to play, and what they liked and disliked about each game. Each participant was asked a set of standard questions during and after the playing of each game.

Data is in two forms: videotape and annotated transcripts of the videotape. Subjects were videotaped while they performed the task, resulting in fifty-four hours of videotape. Transcripts were produced from the tapes containing the verbal data, the visual data necessary to understand the verbal data, and supplemental information, such as counter

information from the VCR in order to provide pointers to the videotape if it was necessary to review a portion of tape. The transcripts were used to answer the following questions: what were the goals the subject was trying to achieve; what were the subject's measures of progress and success; did the subject knew the goals and measures of success built-in to the game; how did the subject react to and use feedback; how were levels of difficulty and options used; how were help and analogies used; why the subject continued to play or quit playing a game; what the subject thought would happen if the game was continued; and how the games were ranked in terms of preferences and why.

6. Classification of Subjects

Subjects were classified based on measurable aspects of system use, verbalizations during the experiment, and post-experiment interviews. The data produced included computer expertise, game playing expertise, learning strategy, and level of risk-aversion. A binary ranking was used for expertise. A more fine-grained categorization was inappropriate given the sample size. Also, most of the subjects had used computers over many years, so that a categorization based solely on the length of time computers had been used would not accurately capture subjects' level of expertise.

A number of criteria to aid in the identification of a user's learning strategy were used, primarily the verbalizations of subjects and their use of help. In general, the identification of a comprehensive strategy was well-defined, while an operational strategy was detected by the lack of indications of a comprehensive strategy. The criteria were used consistently over the multiple games, given differences in the structure of the games and the availability of help. Indications of an operational learning strategy include when a player had developed a sparse set of rules for playing a game, and would not try to expand or refine the rules even when there was concrete evidence, such as from the solution, that rules were erroneous. Strong indications of a comprehensive strategy include reading help to gain a far more thorough knowledge and understanding of a game than is needed to actually play it, backtracking through the help system, and rereading help after playing the game to confirm and extend knowledge. Use of pull-down menus to explore the system rather than to locate a command or option was also an indication.

Explicit behavior is connected to risk-acceptance more strongly than to risk-aversion. We used the criteria of cautious behavior, a dislike of doing something "stupid", such as wasting a move, and hesitation to try out something, to indicate risk-aversion. Risk acceptance was indicated by a subject clicking around on the screen and not being especially concerned about the potential ramifications of a move or a mouse click. This behavior was especially apparent before a game was understood.

7. Goals

Computer games are typically designed with built-in goals. Malone [80] states that "without a clear goal the game is not really a game at all." The goals built into a game include beating the computer, in the case of a two-player game where the computer is the opponent, obtaining a high score, in the case of single-player games, and solving a puzzle. However, we found that although games are typically designed with a clearly-

defined, built-in goal, game players frequently generate their own goals, which can upstage the built-in goal of the game. Turkle [84] looked at how children use microprocessor-based toys such as "Speak and Spell" and similarly found that children often use them in other than the ways they were intended, generating their own goals for the use of the toy.

A goal categorization is presented which is based on the purpose of the goal. We distinguish between goals related to learning and understanding how to play a game and goals related to playing a game and the outcome of playing the game. The goals that players try to achieve can be quite different from the game's built-in goal; they include reaching a certain level within a game, which is determined by what a player feels is achievable, developing the eye-hand coordination or spatial reasoning necessary to play a game, understanding a game or an aspect of a game, such as understanding what the computer's strategy is, or generating pleasing sounds or visual displays.

The set of goals which players tried to achieve when playing Mystery Box, Think-Ahead, Crystal Quest, and Tetris is presented. The variety of goals players try to achieve is extensive, especially given the apparent built-in goals of the games. The goals are categorized as follows: *L goals* are goals related to learning and understanding how to play a game and *P goals* are related to playing a game and the outcome of playing the game. These two categories are further refined as follows: for L goals, we consider *LP goals*, which relate to learning and understanding the rules and elements of a game, *LS goals*, which relate to learning and developing the strategy needed to play a game or understanding the strategy used by the computer to play a game, and *LC goals*, which relate to understanding and developing control to play a game. For P goals, we consider *PS goals*, which are goals related to the final score and the high score lists, *PO goals*, which are goals related to the outcome of the game other than the score, including the winner, the solution, and the level reached, *PC goals*, which are goals related to control in playing a game, such as getting through a level quickly and forcing the computer's moves, and *PE goals*, which are goals related to entertainment and enjoyment of playing a game, such as the visual effects. A further refinement of goal types is presented in Neal [89].

The primary goals of players for each of the four games, broken down into goal categories, is given below.

Primary Goals - Breakdown by Goal Category

Goal	Mystery Box	Think-Ahead	Crystal Quest	Tetris
LP	11	2	3	6
LS	4	7	0	5
LC	0	0	9	5
PS	2	4	5	5
PO	7	7	5	3
PC	1	4	2	1
PE	0	1	1	0

The appearance of the largest populations for each game in different positions in the table reveals some of the differences between the games. For instance, the predominance of learning versus playing goals for a game indicates whether a game is primarily one involving dexterity, computational, or inferencing skills [Neal 89].

8. Learning Strategy and Primary Goals of Players

We examine the observed results from the experiment relating to primary goal types and learning strategies to determine if the data gives evidence of a significant interaction effect between these factors. To accomplish this we compare the observed data to the expectation under a hypothesis of independence between the factors L/P (the primary goals of players that pertain to learning and the primary goals of players that pertain to playing) and O/C (operational and comprehensive learning strategies).

Our goal is to establish an assertion, that the factors L/P and O/C are not independent, with substantive support obtained from our data. We will do a proof by negation by assuming a hypothesis of independence, calculating the probability of the observed data occurring under the assumed condition, and showing that this probability is small enough that it is unlikely that this hypothesis holds true.

Consider the hypotheses

$$H_0\text{: } \textit{observed data shows no relationship between the 2 factors, L/P and O/C}$$

and

$$H_A\text{: } \textit{L/P and O/C are not independent.}$$

We will refer to H_0 as our null hypothesis and H_A as our alternative hypothesis.

We are comparing expected results under H_0 to our observed data and finding the statistical significance of the discrepancy between the two sets of data. To do this we use Pearson's Chi-Square statistic as our measure of discrepancy. If H_0 holds, K. Pearson showed that this test statistic follows the theoretical $\chi^2_{(df)}$ distribution [Snedecor and Cochran 67], where *df* is given by

$$\textit{degrees of freedom } (df) = (\textit{number of rows} - 1) \times (\textit{number of columns} - 1).$$

We set up our data as a 2×2 table, cross-classified by L/P and O/C, and refer to each element of the table as a cell. The null hypothesis of independence states that each cell probability equals the product of the corresponding row and column marginal probabilities.

Thus, the expected value for cell(a,b) is given by

$$\exp_{ab} = \frac{\sum_j x_{aj} \times \sum_i x_{ib}}{\sum_{i,j} x_{ij}} \qquad (1)$$

where x_{ab} is the observed count in cell(a,b) and the χ^2 statistic is given by

$$\chi^2 \text{ statistic} = \sum_{i,j} \frac{(x_{ij} - \exp_{ij})^2}{\exp_{ij}} \qquad (2)$$

8.1. Mystery Box

For Mystery Box, the observed data and the expected data under the null hypothesis of no independence is:

	Observed Data for Primary Goal		Expected Data for Primary Goal	
goal	operational	comprehensive	operational	comprehensive
L	1	14	6.0	9.0
P	9	1	4.0	6.0

For Mystery Box,

$$\chi^2 = 17.361 \text{ and } df = 1.$$

From the χ^2 table we find that

$$Pr(\chi^2 \geq 17.361 \mid H_0) = .0001.$$

Therefore, the data is consistent with H_A and, barring the presence of a hidden cofactor (or any disturbing influences), the data suggests that an association exists. Thus the data supports our hypothesis that Mystery Box is a strong discriminator for learning strategy.

8.2. ThinkAhead

For ThinkAhead, the observed data and the expected data under the null hypothesis of no independence is:

	Observed Data for Primary Goal		Expected Data for Primary Goal	
goal	operational	comprehensive	operational	comprehensive
L	1	8	3.6	5.4
P	9	7	5.2	9.6

For ThinkAhead,

$$\chi^2 = 4.890 \text{ and } df = 1.$$

From the χ^2 table we find that

$$Pr(\chi^2 \geq 4.890 \mid H_0) = .0280.$$

Therefore, the data is consistent with H_A and, barring the presence of a hidden cofactor (or any disturbing influences), the data suggests that an association exists. Thus the data supports our hypothesis that ThinkAhead is a discriminator for learning strategy.

8.3. Crystal Quest

For Crystal Quest, the observed data and the expected data under the null hypothesis of no independence is:

	Observed Data for Primary Goal		Expected Data for Primary Goal	
goal	operational	comprehensive	operational	comprehensive
L	3	9	4.8	7.2
P	7	6	5.2	7.8

For Crystal Quest,

$$\chi^2 = 2.1635 \text{ and } df = 1.$$

From the χ^2 table we find that

$$Pr(\chi^2 \geq 2.1635 \mid H_0) = .1413.$$

The data is not strongly consistent with H_A and therefore the test is inconclusive for Crystal Quest. The data suggests that an association may exist, thus Crystal Quest is a not a reliable discriminator for learning strategy.

8.4. Tetris

For Tetris, the observed data and the expected data under the null hypothesis of no independence is:

	Observed Data for Primary Goal		Expected Data for Primary Goal	
goal	operational	comprehensive	operational	comprehensive
L	4	12	6.4	9.6
P	6	3	3.6	5.4

For Tetris,

$$\chi^2 = 4.1667 \text{ and } df = 1.$$

From the χ^2 table we find that

$$Pr(\chi^2 \geq 4.1667 \mid H_0) = .0270.$$

Therefore, the data is consistent with H_A and, barring the presence of a hidden cofactor (or any disturbing influences), the data suggests that an association exists. Thus the data supports our hypothesis that Tetris is a discriminator for learning strategy.

8.5. Mystery Box and ThinkAhead Considered Together

We next consider the games Mystery Box and ThinkAhead together, to determine if the pair of games can be used as predictors of learning strategy. We show the primary goals for the individual games followed by goal pairs we construct from the primary

goals for each game.

For Mystery Box and ThinkAhead, the observed data and the expected data under the null hypothesis of no independence follows, where the goal pair (L,L) means an L goal for Mystery Box and an L goal for ThinkAhead, (L,P) means an L goal for Mystery Box and a P goal for ThinkAhead, etc.

	Observed Data for Primary Goals		Expected Data for Primary Goals	
goal pairs	operational	comprehensive	operational	comprehensive
(L,L)	0	7	2.8	4.2
(L,P)	1	7	3.2	4.8
(P,L)	1	1	0.8	1.2
(P,P)	8	0	3.2	4.8

For Mystery Box and ThinkAhead together,

$$\chi^2 = 19.27 \text{ and } df = 3.$$

From the χ^2 table we find that

$$Pr(\chi^2 \geq 19.27 \mid H_0) = .0002.$$

8.6. Discussion

In examining the observed results from the experiment relating to primary goal types and learning strategies, we find evidence of a significant interaction effect between these factors. The tests support the hypothesis that learning strategy can be predicted by observing the primary goal a user is trying to achieve. The strongest evidence of this is with Mystery Box, and with Mystery Box and ThinkAhead when the primary goals are considered as a pair. The weakest evidence of this is with Crystal Quest, which does not provide as reliable a discriminator for learning strategy as do the other three games.

Games are not designed for this purpose, yet they provide reliable detectors for learning strategy. However, similar analyses yield no results for decision-making strategies. There was some evidence that risk was a factor in when Mystery Box players chose to find out if their solution was correct. A player of Mystery Box, when describing what he liked about the game, said, "The thing that makes this interesting is you make the guesses but without thinking much about whether the guess is absolutely correct. So there's some kind of risk; you want to see if you have made the right guess." Another Mystery Box player said, "I have negative points now. I'm starting to feel that I wanted to just guess with this one, just to find out whether I guessed right with that one so I could get some feedback. But then the other thing was I wanted to do better than the last time, so maybe I'd want to get another ball just in case I was taking too much of a risk with that one and I'm wrong... Let's see if I did any better." He had one ball close out of two balls, and started another game after carefully examining the solution. After firing a few rays, he said, "If the ball was here it would have gotten absorbed. I can't wait. I'm just

going to guess." This time he only tried to place one ball, and put the other one in a corner. However, other players were far more cautious, and delayed making a guess even to the detriment of their score.

9. Implications

Implications of our work for systems design, which are related to the cognitive variation of users, include command design, measures of progress and success, help and system response, and graduated challenge [Neal 90]. Systems need to be different to accommodate the range of learning and decision-making strategies. For instance, responses to queries made to a help, advising, or tutoring system can be tailored by providing procedural information to operational learners and in-depth explanations to comprehensive learners. This will extend the capabilities of systems which do not tailor responses or which exclusively use domain knowledge to determine the kind of information and the amount of detail to provide in a response.

Understandable feedback for all commands and actions should be provided to risk-averse users, increasing their ability to comfortably use a system. It is especially important that there always be a system response to erroneous user input, where the nature of the response should be dependent on the class of error and on the user. The level of risk-aversion can be expected to correlate with the appropriate level of intervention when a user is experiencing difficulties. In addition, the level of risk-aversion can be expected to indicate the type feedback to provide and when it is most important to avoid meaningless or confusing feedback [Carroll 82]. An uncluttered screen with extensive use of fonts, highlighting and color helps a risk-averse user focus attention and provides a greater sense of control, and the availability of an undo command alleviates fears and encourages experimentation of a system [Neal 87b]. While these features aid a risk-accepting user as well, they are necessary for reducing a real or perceived risk in system use for a risk-averse user.

In our analysis of game features, we found that some are more important than others in the achievement of objective success and some are significantly correlated with game popularity [Malone and Lepper 87]. Many of these features are important to consider as part of general system design as well [Neal 90]. Games can be viewed as providing a testbed for interface design, demonstrating elements of an interface that are effectively or poorly designed, implemented, and integrated into an environment. "Games provide an extremely compelling interface whose advantages should be considered for more standard applications" [Krueger *et al.* 85]. The types of features that were problematic in games are likely to problematic in any software, and the incorporation of successful features from games could contribute to improving system design.

10. Conclusions and Future Directions

It is expected that an approach to design that takes cognitive variation into account can support the construction of systems which are successfully used. The incorporation of a user model characterizing relevant variables allows the tailoring of a system to the needs and abilities of its users. Predictive tasks can be used to build multi-dimensional user models, which can be used to provide different versions of a system to users who differ in ways that are relevant to their interaction with the system.

In examining the results from the experiment relating to primary goal types and learning strategies, we find evidence of a significant interaction effect between these factors. The tests support the hypothesis that learning strategy can be predicted by observing the primary goal a user is trying to achieve. Games that are not designed for the purpose of being discriminators of learning strategy do provide reliable discriminators.

A number of new questions emerge from our work. In particular, we would like to determine if there are patterns of goals that are characteristic of risk-averse or risk-accepting users. A possible direction to establish a risk-aversion detector is suggested by preliminary evidence that risk-aversion was a factor in when Mystery Box players chose to find out if their solution was correct. In future work, we would like to determine if when a guess is made in Mystery Box provides a consistent and reliable indication of level of risk-aversion or risk-acceptance. It is also possible that some other aspect of game use, such as a more complex configuration of goals, provides a discriminator for risk-aversion. In future work, we would also like to as fully as possible automate the building of user models through predictive tasks.

Bibliography

Carroll, J. M. The Adventure of Getting to Know a Computer. *Computer Magazine.* (1982), pp. 49-58.

Dagwell, R. and Weber, R. System designers' user models: a comparative study and methodological critique. *Communications of the ACM* 26(11), 1983, pp. 987-997.

Feigenbaum, E. A. and Feldman, J. *Computers and Thought.* New York: McGraw-Hill, 1963.

Fried, L. Nine Principles for Ergonomic Software. *DATAMATION,* November 1982.

Hink, R. F. and Woods, D. L. How Humans Process Uncertain Knowledge: An Introduction for Knowledge Engineers. *AI Magazine,* V. 8, N. 3, Fall 1987, pp. 41-53.

Kass, R. *Implicit Acquisition User Models in Cooperative Advisory Systems.* Technical Report MS-CIS-87-05, Department of Computer and Information Science, University of Pennsylvania, 1987.

Kass, R, and Finin, T. Rules for the Implicit Acquisition of Knowledge about the User. In *Proceedings of AAAI'87,* pp. 295-300, 1987.

Krueger, M. W., Gionfriddo, T., and Hinrichsen, K. VIDEOPLACE -- An artificial reality. In *Proceedings of CHI'85 Conference on Human Factors in Computing Systems* (San Francisco, April 14-18, 1985), ACM, New York, pp. 35-40.

Malone, T. W. *What Makes Things Fun to Learn? A Study of Intrinsically Motivating Computer Games*. Technical Report CIS-7, XEROX PARC, 1980.

Malone, T. W. Toward a Theory of Intrinsically Motivating Instruction. *Cognitive Science,* 4 (333-369) 1981.

Malone, T. W. and Lepper, M. R. Making Learning Fun: A Taxonomy of Intrinsic Motivations for Learning, In Snow, R. E. and Farr, M. J. (editors), *Aptitude, learning and instruction: III. Conative and affective process analysis*. Hillsdale, N.J.: Erlbaum, 1987.

Neal, L. R. Cognition-Sensitive Design and User Modeling for Syntax-Directed Editors. In *Proceedings of CHI+GI'87 Conference on Human Factors in Computing Systems and Graphics Interface* (Toronto, April 5-9). ACM, New York, 1987a, pp. 99-102.

Neal, L. R. User Modeling for Syntax-Directed Editors. In Bullinger, H.-J. and Shackel, B. (editors), *Human-Computer Interaction - INTERACT'87*. Amsterdam: North-Holland, 1987b.

Neal, L. R. *The Role of User Models in System Design*. TR-18-89, Harvard University, October 1989. Ph.D. Dissertation.

Neal, L. R. Implications of Computer Games for System Design. Accepted for publication in *Proceedings of INTERACT'90 3rd IFIP Conference on Human-Computer Interaction,* Cambridge, England, August 27-31, 1990.

Rich, E. *Artificial Intelligence*. New York: McGraw-Hill, 1983a.

Rich, E. Users are Individuals: Individualizing User Models. *International Journal of Man-Machine Studies* (1983b) 18, pp. 199-214.

Snedecor, G. W. and Cochran, W. G. *Statistical Methods,* The Iowa State University Press, 1967.

Turkle, S. *The Second Self: Computers and the Human Spirit*. New York: Simon and Schuster, 1984.

Users' Models of Computer Systems

Martina-Angela Sasse[1]
School of Computer Science
The University of Birmingham
Birmingham B15 2TT
UK

Abstract

The paper describes a series of experimental observations of users' models of computer systems. The observations were devised to overcome two methodological shortcomings prevalent in experimental studies to date: over-interpretation of performance data and lack of ecological validity. The five observations cover a range of experimental scenarios, ranging from highly structured benchmark tests to relatively unstructured scenarios based on constructive interaction technique. The sessions were recorded on videotape, and subsequently analysed for a range of quantitative and qualitative data. These were used to attempt a reconstruction of the models held by the subjects of the systems they were interacting with.

1. Introduction

The term '*mental model*' is widely used in the field of Human-Computer Interaction today, encompassing quite a variety of models and representations. This paper deals with *users' models* (a term introduced by NORMAN, 1986); to be precise: actual conceptual models the user holds of a computer system (UC/ACM in the taxonomy proposed by NIELSEN, 1987). It is now a generally accepted truism in HCI that users form such a model when interacting with a complex device, and that the success of a user's interaction with a system depends on whether this model is appropriate.

There is less agreement in the literature, and the HCI community in general, as to what makes a user's model an appropriate one. Most users' models (certainly those of novice and casual users) have been found to be imcomplete and 'unscientific' (NORMAN, 1983). Users do not, however, necessarily need a complete model representing every detail and accurate structure (a *surrogate model* - YOUNG, 1983) to be able to use a system successfully: a reduced model that correctly accounts for the functionality (a *mapping model* - YOUNG, 1983) can be sufficient to operate a device. There are manifold examples for this in everyday life - only a small number of car drivers have a structural model of how an internal combustion engine works.

---

1. Current address: Philips CID, Building SX, Postbus 218, 5600 MD Eindhoven, The Netherlands. Email: angelas@ica.philips.nl

The very existence of the discipline HCI proves that most computer users do not spontaneously develop models that lead to successful interaction. The aim of the research described was to obtain examples of representations that users hold of computer systems and other complex devices, and to investigate the relationship between those models and users' performance. Knowledge of how users construct those models, and which models lead to successful interaction, could make an important contribution to cognitive ergonomics research, and act as a basis for a new approach to the design of user interfaces and training materials.

2. Research to date

Studies on users' models in HCI have been published since the mid-eighties. This section gives a brief review, based on four experimental investigations of users' models: KIERAS & BOVAIR, (1984); BORGMAN, (1986); BRIGGS (1988) and FRESE et al. (1988), and discusses the problems with traditional approaches and the measurements taken.

"*Let me warn the nonpsychologists that discovering what a person's mental model is like is not easily accomplished.*" A review of the publications on users' models proves the prophetic value of NORMAN's (1983) footnote. As with other cognitive phenomena, users' models are not directly observable. Therefore, we can only try to infer what models users hold, for instance by watching users interacting with a system.

One of the first large-scale experimental investigations published was that by KIERAS & BOVAIR (1984). They reported a series of three experiments, in which two groups of students were trained to use a control panel device. One group was trained by what is described as 'rote learning': the subjects acquired a set of procedures enabling them to operate the device. The other group was taught a surrogate model of the device (a drawing and verbal description), but was not trained in the procedures. The subjects in this group had to pass a knowledge test after the training session, to ensure that they did understand how the device worked.

After the initial training session, KIERAS & BOVAIR (1984) performed a series of experiments on both groups, asking the subjects to operate the device, infer new procedures, and tried to elicit the user's model through an interview with the subjects. They found that the subjects in the 'model' condition learned the procedures faster, executed them faster, and managed to simplify ineffective procedures more often than the subjects in the 'no model' condition. The subjects in the 'model' group were capable of inferring new procedures needed to perform a transfer task, and showed a better retention rate in a second session, which took place a week after the training. In the final interview with the subjects, KIERAS & BOVAIR (1984) found that the better performance of the 'model' group was due to specific knowledge of the workings of the device, rather than differences in motivation.

The study performed by BORGMAN (1986) follows a similar approach: two groups of students were trained to use an on-line catalogue. The content of the instruction material given to both groups was identical, except the 'model' group was taught on the basis of a

conceptual model (the system was explained in terms of an analogy to a card catalogue). The subjects were asked to perform a range of benchmark tasks, to attend an interview, and to perform one task while talking aloud to an experimenter.

As far as performance on the benchmark tasks was concerned, BORGMAN (1986) found no differences on simple tasks, but the 'model' group performed better on complex tasks. The surprising result of the interview was that the descriptions of the systems did not vary between the two groups: BORGMAN (1986) expected the subjects in the 'model' condition to describe the system in analogy with a card catalogue, and the subjects in the 'no model' condition to state less complete and correct models. She found, however, that most subjects held some sort of model of the system, regardless of the training condition. Only 4 out of 28 subjects specifically referred to the system in analogy to a card catalogue - and one of them had been trained in the 'no model' condition. BORGMAN (1986) attributed this to the methodological difficulty of capturing a user's model, and concluded that the interview technique had captured a difference in ability to articulate a user's model, rather than ability to build one. Furthermore, she thought that the above-average intelligence of the subjects (Stanford undergraduates) enabled them to generate their own models of the system, rather than relying on externally provided models.

I am not fully convinced that this is the strongest possible explanation of the results: from the review of the theoretical literature, we know if is difficult to find analogies that are worth using as a conceptual model. After all, BORGMAN (1986) says herself that some features of the system were difficult to explain in terms of the analogy:

This study illustrates the *importance of qualitative data for research on user's models*: on the basis of the performance data alone, we could not reject BORGMAN's (1986) claim that the analogy provided an effective conceptual model. The relationship between performance data and the user's model is not straighforward: differences in performance can indicate that some users hold more appropriate models than others, but this is not necessarily the case. People often do the right things for the wrong reasons, and users performing well on a benchmark task can hold the most appalling misconceptions.

The problems with using performance data to test hypotheses about users' models are aggravated in studies with small samples: the reliabilty of statistical evidence is generally reduced, and the tests that have to be employed tend not to be robust. In the study by FRESE et al. (1988), a set of performance data was collected to compare the effect of three different types of training on users' models on a total of 15 subjects (5 in each group). The authors concede that the groups are rather small, and that the the tests employed are not very 'conservative'. This does, however, not prevent them from employing the results to 'prove' their own ideas on user's models. The general interpretation in question is that a 'active' acquisition of a user's model, through exploration and making mistakes, leads to a better operative user's model. The subjects in the two control groups were supposed to acquire a 'passive' user's model, either through a procedural approach or through a 'conceptual model' (a manual and a hierarchical diagram of the tasks). Apart from the fact that a distinction between 'active' and 'passive' user's models is quite inappropriate, since model-building is always an active process, the authors ignore an obvious alternative explana-

tion originating from the design of the training for the different groups: the level of interaction between subjects and experimenters. In the control groups, subjects followed written instructions. The experimenter intervened and interacted with the subjects only when they made a mistake. In the 'active' condition, subjects were given no written material. They were encouraged to generate and test hypotheses - which then were "... elaborated with the guidance of the experimenter down to the keystroke level". It seems to me that, essentially, the authors are comparing the effect of personal tuition with learning through written instruction.

The paper by BRIGGS (1988) gives a good example of how a study with a small number of users can contribute valuable data to research on user's models. She investigated how well novice users are able to identify the kinds of knowledge they need to interact with the system. The technique employed by BRIGGS (1988) was to let subjects ask questions about the system before embarking on editing a document, and compare the questions asked beforehand with those the users had to ask when they performed the task. The conclusion was that, for lack of a user's model, the users were not able to identify what they needed to know about the system.

2.1 Current methods

This section lists and briefly discusses the methods used in experimental studies on users' models to date.

Thinking-aloud techniques and verbal protocols

'Thinking aloud' techniques have been extensively employed in HCI . Users are encouraged to verbalise their thoughts and conclusions while they are interacting with a system. This 'encouragement' is often given by the experimenter thoughout the session in form of questions or prompts: subjects may be asked, for instance, to explain their own behaviour, reason about the system's behaviour , or reason about errors. The sessions are usually recorded on audiotapes or videotapes, and a written protocol of the interaction is generated on the basis of these. The scientist can then attempt to derive a conceptualisation of a user's model from the explanations.

NORMAN (1983) states that these protocols may be informative, but are always incomplete: people cannot give complete verbal accounts of their thought processes. The cueing or construction of a user's model is a subconscious process, and we cannot expect verbal accounts of subconscious processes. If we ask a user to verbalise these processes, the very shift from subconscious to conscious processing can change nature and content of the processes we are trying to tap. Such accounts can actually yield erroneous information, since asking subjects to think aloud changes the *demand structure* of the situation: users probably tend to rationalise their behaviour if they have to explain it, or tend to tell the experimenter what they think s/he wants to hear, or might be guided by subtle verbal and non-verbal cues given by the experimenter.

'Thinking aloud' is obviously highly artificial behaviour, but we can remove some of this artificiality by making the explanation of models a necessary part of a higher-level task.

On-line protocols

NORMAN (1983) advocates on-line protocols which record the behaviour of users and system during the interaction, because they are more 'reliable' than explanations gathered from the users. It is quite obvious, however, that a conceptualisation derived from an on-line protocol of user-system interaction will not yield much information about a user's model. After all, on-line protocols record the user's and the system's behaviour - the experimenter will have a record of what the user did, but know nothing about why s/he did it. Conceptualisations constructed on this basis reflect guesswork on behalf of the experimenter about a user's model rather than a good approximation of the model.

Having some record of system behaviour is essential for the interpretation of a user's verbal protocols. But a record of the screen display which can be linked to the user's behaviour (verbal and otherwise) is, to my mind, sufficient. Such accounts help to illuminate utterances like: *'I' ll try to move this thing over here'* . Videotaped accounts of user-system interaction are a good way to capture essential, higher-level data, and are easier to produce and analyse than on-line protocols.

Performance data

Most performance data in research on users' models are collected through benchmark tests: after some initial training (on the basis of which the user is supposed to have constructed a model) the subjects are asked to perform a set of tasks. The most common measurements taken are: time taken to complete a task and/or subtasks, frequency and type of errors, efficiency of task completion (rating by experimenter), efficiency of training (time/error ratio, transfer task), and retention rate.

The tasks presented to the subjects in *benchmark tests* do not require any knowledge beyond that presented in the training. Such tests are mainly applied to compare the performance of users that have been trained by different methods, and are assumed to have constructed different models as a result of the training. However, performance alone cannot be seen as proof that the user does hold a certain model.

Performance time may be an 'objective' measurement and easy to obtain, but tells us very little about a user's model. There is no strong theoretical link between a user's model and the time it takes him/her to complete a task. There is, however, a strong correlation between time and errors, since incorrect actions and revocery obviously cost time.

Errors are among the most informative data about users' models, if some analyis is performed on them. Unfortunately, most of the experimental studies on users' models in HCI count errors rather than analysing them. Possible types of analyses are qualitative-explanatory analysis , or a classification of errors. We can apply a general classification (e.g. NORMAN, 1988), or devise a classification suitable to describe the system used. It makes sense to adapt some of the categories from a general classification, and distinguish between slips and errors, and apply categories well-known in HCI - such as mode-errors - if appropriate. On the basis of error analysis, we can try to identify missing representations or mis-representations that the user holds, and try to build a conceptualisation of his/her model. It is, however, not always easy to determine exactly which inappropriate represen-

tation caused a particular error. Scenarios where the experimenter can try to prompt the user to comment on the error (e.g. 'What did you think the system would do?) can yield valuable insight in this respect. One of the drawbacks of error analysis is that, while we can learn a lot about models constructed by users who make errors, there is very little information to be gained about models held by competent users.

Some experimenters (KIERAS & BOVAIR 1984; FRESE et al., 1988) ask subjects to perform a *transfer task*. The procedures that have to be applied to solve a transfer task are not explicitly presented to the users, but can be 'inferred' from the material presented during training. The ability of users to infer new procedures is seen as a crucial test of the appropriateness of the models they hold.

Retention of information over time is a measurement that has been employed by KIERAS \& BOVAIR (1984) and FRESE et al. (1988). The argument for retention rate as a performance measure is that it should be easier to remember information that has been integrated in single, coherent, plausible models. FRESE et al. (1988) simply counted the number of commands remembered, whereas KIERAS & BOVAIR (1984) asked subjects to perform another benchmark test on the device they had been trained on before.

2.2 Subjects

The users in the studies reviewed were all 'novice' or 'naive' users, and -- apart from those in the study by BRIGGS (1988) -- undergraduate students. The experimenters not only established that none of the subjects had ever used a system similar to the one in the study, but recruited subjects with little or no experience of computer use. It is certainly easier to standardise the previous knowledge and experience of users when the assumed level of relevant knowledge is little or none. But even when subjects are selected very carefully (FRESE et al. 1988, seem to have gone to great lenghts in this respect), there is no way we can standardise the knowledge and experiences that users bring to the task.

Studies with novice users focus on a particular stage of a particular type of user-system interaction. While design should facilitate access for novice and casual users, theories of users' models have to cover all groups of users: even novice users are novice users only for a certain period of time. Some of the difficulties users have in their very first interaction with a particular system are simply orientation problems, which will disappear with practice. It is important to separate these initial difficulties from those that will persist with prolonged use of the system.

2.3 Experimental scenarios

When conducting experimental work on user's models, HCI researchers have to face the same dilemma as researchers in psychology: the tradeoff between *internal* and *external validity*. Internal validity can be increased in empirical studies on users' models by structuring user-system interaction tightly, and trying to control or eliminate all possible other influences. This is difficult to say the least. Even when user-system interaction is tightly structured, users will create different situations in the course of the interaction, and different interactions and responses will be required from the experimenter. There can be no doubt that a tightly controlled experimental scenario is not very conducive to users inter-

acting with the system in a 'natural' way.

Research on users' models should, in my opinion, aim to achieve a high degree of external, or ecological, validity: the results should be applicable to user-system interaction outside the laboratory. HCI is, after all, an applied science, and the research conducted should look at systems, users and interaction situations as they can be found in the workplace. One-off experiments with novice users learning a simple device will not contribute very much to an applicable theory of users' models.

2.4 Summary: Problems with current methodologies

The main problems with current experimental methodology in research on user's models in HCI are over-reliance on performance data and lack of ecological validity. Field studies could increase ecologival valididty - they are, however, are virtually unknown in HCI. They require a lot of resources and manpower, and pose considerable organisational problems. In research on users' models, we collect information from a set of individual user-system interactions. It is not unreasonable to try to elicit users' models in a laboratory environment, provided the following points are observed:

Experimental scenarios should not be too artificial or restrictive. While it is desirable to structure the interaction to some extent -- in form of tasks -- in order to be able to compare the information elicited from different subjects, users should be controlling the interaction. If we want to observe how users select and process information, we have to place them in an active role. Real systems and meaningful tasks should be used. To elicit users' models, the experimenter has to encourage verbalisation of their thought processes, either through prompting or by employing a scenario that makes communication of users' thought processes part of the task.

A range of qualitative and quantitative data should be collected. Videotapes of user-system interaction can be used to provide a range of quantitative and qualitative data. An additional advantage is that an analogue record of the interaction is available, and that several investigators can rate users' performance and interpret their verbal behaviour.

Samples should be neither too small, nor concentrate on novice users only. If users are split into different groups to test the effect of an independent variable, the groups should be big enough to allow for the application of robust statistical tests. There is a lack of studies investigating users' models on groups other than novice users.

3. Five scenarios for investigating users' models

The aim of the research described here was to devise a set of experimental scenarios to elicit users' models which avoided the problems with current techniques as presented in the previous section. In designing these studies, I have also tried to put into practice a suggestion by YOUNG (1983) that models could be elicited by observing users *using* the system, users *explaining* the system, users *predicting* the behaviour of the system, users *learning* the system.. After a brief description of subjects and the general setup, an account

of each study is given, commenting on the strenghts and weaknesses of the scenario employed.

Subjects: The users were students at the University of Birmingham, taking a subsidiary course in computing. Out of a total of 32, between 21 and 26 students volunteered to take part in the individual sessions. The samples were nearly balanced with respect to gender; the age of the students was between 18 and 34. Most of the students had no computing experience whatsoever when they started the course.

General procedure: Each of the five studies observed the users interacting with a computer system - three applications and a programming language on an Apricot microcomputer, and one application on an Apple Macintosh. Prior to the experimental sessions (with the exception of the last) the users were taught to use the applications in hands-on tutorials and workshops (2 hours per week). All tutorials and workshops were run by the author, who also designed all training materials and exercises, following the idea of minimalist instruction (CARROLL, 1985). At the end of the training for each application, the users were asked to attend individual experimental sessions. All experimental sessions were recorded on videotape. One camera recorded the user, a second camera recorded the computer screen. A microphone recorded the users' verbal comments. The pictures from the two cameras were mixed and recorded on the same tape.

3. 1 Observing users using a system

The first study observed 22 users editing a document on a wordprocessor (Wordstar). The experimenter (the author) sat next to the user, answering questions the user asked, and prompting them throughout the course of the interaction. The experimenter tried to follow a certain 'script' throughout the interaction by applying the following rules:

When a user asked the experimenter how to solve a certain task, the experimenter would not answer by telling the user the procedure in the first instance. Instead, the experimenter would ask a question like; '*How did you delete* (insert, underline) *a word* (sentence, paragraph) *in the exercises in class?*'. Only when it was clear that the user could not remember the procedure, the experimenter would tell the user how to solve the task.

Whenever a user made an error, the experimenter would wait to see if the user would comment on the error, try to recover from it, or try again. If the user did not say or do anything following the error, the experimenter would try to prompt the user to comment on the error, and how to recover and perform the task. Whenever the user made a comment that indicated s/he was refering to some kind of user's model, the experimenter would encourage the subject to elaborate their statement.

This experimental scenario led to a fairly similar structure across the individual user-system interactions, and seemed to elicit a lot of detailed information. The disadvantage of the scenario was that some users clearly felt under pressure to perform. This was partly due to the fact that the experimenter and the course tutor were the same person - even though assurances were given that the sessions would in no way be used to assess the users performance for the practical credits. The disadvantage of prompting from the experi-

menter was highlighted in this study: it occasionally seemed to interfere with the users' problem solving processes. Furthermore, it was difficult to elicit knowledge from the more competent users. While errors provided the experimenter with good opportunities for prompting, prompting without errors seemed to be 'unnatural', and made users more self-conscious -- an observation in line with NORMAN's (1983) assertion about the change of demand structure.

3.2 Users explaining a system

In the second study, the 26 users were asked by the experimenter to teach another person (a co-experimenter) how to use the spreadsheet software (Open Access) they had been learning in the previous weeks. The experimenter suggested that the 'learner' (co-experimenter) should operate the keyboard, and that they should try to teach him what knowledge they thought was necessary to use the spreadsheet. The experimenter would then leave the room and watch the session on a monitor in the adjacent control room. The users were told that they could call for the experimenter if they got into a situation from which they were unable to recover and continue. This *teaching back* scenario is based on an observational technique called *constructive interaction* (MIYAKE, 1982) and a variant of the *thinking aloud* approach. The idea is to study two people who jointly try to solve a problem, capitalising on the fact that communication of knowledge takes place naturally between the two participants. Therefore, the techique avoids the change in demand structure (O'MALLEY et al., 1984). The success or failure of this scenario obviously depends to a large extent on the skills and behaviour of the co-experimenter. The subjects have to be fully convinced that the 'learner' does not know anything about the system - otherwise they are not motivated to communicate their knowledge to the co-experimenter. The co-experimenter very cautiously tried to prompt the users by demanding explanations whenever appropriate. Some users tended to give entirely procedural instructions . Demanding explanation for those procedures is a good way of encouraging them to verbalise the models that the procedures are derived from -- or indeed check if they do hold a model at all.

This *teaching back* scenario worked extremely well -- in terms of conceptual knowledge elicited, probably best of all the approaches described here. Most of the users enjoyed the task, and communicated a great deal with the co-experimenter. This illustrates the benefit of a 'natural' context for the interaction. A great deal of knowledge can be elicited from the more competent users. The disadvantage of this scenario is that when less competent subjects do get into a situation from which they cannot recover and have to call for the experimenter, the result is a considerable disruption in the interaction.

3.4 Users predicting behaviour of a system

26 users participated in the third study, which asked them to predict the behaviour of a Prolog program. The experimenter presented them with a short Prolog program on the computer, consisting of a knowledge base and one rule. The users could also refer to a printout of the program. The study consisted of three sets of tasks: To start with, the exper-

imenter asked the user to type in a number of Prolog queries, and predict the answer the system would return before entering the question. The experimenter then presented a few questions in natural language, and asked the user to construct Prolog queries to find the answer to these questions. The user then proceeded with queries. If the query did not return the predicted answer, the experimenter again asked the user to work out what was wrong with the query. Finally, the experimenter asked the user to constuct a rule and add it to the program. The user was then asked to explain what the rule would do, and run a few queries to check if the rule worked as intended. If the rule did not work, the experimenter encouraged the user to find out why.

The study yielded quite a good insight in the users' understanding of Prolog. The first task elicited more from the less competent users, since - like in the first scenario - errors made by the user provided a good opportunity to discuss the workings of the system. The second and third tasks provided a better chance to review the models held by the more competent users, since it allowed observing problem-solving behaviour. The interaction was fairly structured, but less artificial than in the first study. Predicting the behaviour of a system, and asking users to make changes to the system, is a good way of checking whether the user holds a representation which could be described as a *surrogate model* of the system.

3.4 Users describing and using a system

The fourth study was not directly based on YOUNG's (1983) suggestion, but tried to illuminate the relationship between knowledge that users hold and their performance. 21 users participated in this study, and the system in question was a database (Open Access). First, the experimenter asked the user to explain what a database system is. She checked whether the users knew the basic terminology and commands of the SQL type query language. Then, presented the user with a database file (a customer record) and three tasks .

This experimental study yielded some interesting observations: while most of the users had no problem whatsoever to perform the three tasks, very few of them knew the terminology or could describe the basic concepts accurately. They remembered the commands of the query language and knew how to apply them, but could not explain what the commands actually did.

The interaction process in this scenario is highly structured, but does not provide the experimenter with many opportunities to elicit users' models. It might be more appropriate for testing specific hypotheses about users' models and performance. The observations made in this study suggest that the users certainly did not hold a surrogate model of the system. The fact that they were, nevertheless, able to operate the system and solve the tasks indicates that they might have developed something akin to a mapping model.

3.5 Observing users learning a system

In the final study, 22 users were asked to explore a word processor (Macwrite) in a one hour session. The 'observing users learning a system' approach was left for the last of the

five studies, since the six month course should have allowed the users to acquire some relevant background knowledge, i.e. a set of relevant models that could be applied to the learning of a new system. Since all studies on users' models so far concentrated on novice users learning a system, it seemd more interesting to observe users who had some experience with computer systems learning a system.

The experimenter introduced the user to a *co-learner* (co-experimenter) and asked them to work together to find out how the word processor worked. This scenario is a further variation of the *constructive interation technique* which was also employed in the second study. Applying a straightforward *constructive interaction* approach would have paired two users to explore the system. In view of the fact that all the other studies observed one user per session, I decided to vary the approach and pair each user with a contrived co-learner. This change in the approach counteracts the possible danger of one partner taking charge and dominating the interaction, in which case there would be little knowledge elicited from the less dominant partner.

The experimenter suggested that the user should operate the keyboard, while the *co-learner* should consult the manual whenever the user felt that s/he needed some specific information about the system. As with the *teaching back* scenario in study 2, the success or failure in eliciting users' models depends largely on the skill and discretion of the co-experimenter. If the co-experimenter is perceived as having knowledge about the system, the user will rely on the more expert 'co-learner' to feed them the procedures which have to be employed to solve the task, and not contribute very much to the interaction. In a bid to avoid such a situation arising, the co-experimenters were briefed to be interested, but not over-helpful: they were not to respond too quickly when the user asked a question, and always had to look the answer up in the manual. When they had located the relevant text, they were to give a vague answer rather than an exact procedure in the first answer to a question (e.g. they answered *It's got something to do with the ruler* when the user could not work out how to change the margin). .

The scenario worked quite well in terms of verbal responses elicited from the users. The interaction was, through the provision of a series of tasks, more tightly structured than in the other constructive interaction scenario (study 2). Most of the users enjoyed the sessions, and were quite confident in exploring the system. A detailed analysis should give information about how the use of the previous word processing system influenced the way they approached this one.

4. Conclusions: Scenarios for empirical studies on users' models

The scenarios based on the *constructive interaction* technique (studies 2 and 5) seem to be most effective for eliciting users' models, but require more resources and preparation than the highly structured approaches. It is, however, effort well spent if structure and content of the knowledge elicited are to be used for a conceptualisation of the users' models.

The structured approaches seem to be more suited to testing specific hypotheses about users' models, rather than eliciting them. Highly structured scenarios like the one in study 1 are restrictive and artificial; while they are easier to set up and analyse, researchers using

this approach have to be aware that they might be trapping users' models rather than tapping them. Predicting the behaviour of a system and asking the user to make changes (study 3) is a good method to test whether the user holds a surrogate model of the system. Contrasting users' conceptual knowledge about a system, and their ability to use the system to perform tasks (study 4) is a method that could be employed to test for mapping models.

In the analysis, it turned out that concentrating on users' errors and their recovery procedures was most fruitful for a reconstronction of users' models. This leaves us with a problems as to how to elicit models from competetent users. Here, the contructive interaction technique seems the only promising approach to elicit data for reconstruction of the users' model.

References

BORGMAN, C. L. (1986): The user's mental model of an information retrieval system: an experiment on a prototype online catalog. INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES, 24, 47-64.

BRIGGS, P. (1988): What we know and what we need to know: the user model versus the user's model in human-computer interaction. BEHAVIOUR AND INFORMATION TECHNOLOGY, 7, 431-442.

CARROLL, J. M. ; \& MACK, R. L. (1985): Metaphor, computer systems, and active learning. INTERNATIONAL JOURNAL OF MAN-MACHINE STUDIES, 22, 39-57.

FRESE, M. et al. (1988): The effects of an active development of the mental model in the training process: experimental results in a word processing system. BEHAVIOUR AND INFORMATION TECHNOLOGY, 7, 295-304.

GENTNER, S. & STEVENS, A. L. [Eds.] (1983): Mental models. Hillsdale, NJ: Erlbaum.

KIERAS, D. E. \& BOVAIR, S. (1984): The role of a mental model in learning to operate a device. COGNITIVE SCIENCE, 8, 255-273.

MIYAKE, N. (1982): Constructive interaction. Tech. Rep. No. 113. San Diego: University of California, Centre for Human Information Processing.

NIELSEN, J. (1987): The spectrum of models in software ergonomics. Proceedings of the Fifth Symposion on Empirical Foudations of Information and Software Science, Roskilde, Denmark, 23-25 Nov. 1987.

NORMAN, D. A. (1988): The Psychology of Everyday Things. New York: Basic Books.

NORMAN, D. A. (1986): Cognitive engineering; in: NORMAN & DRAPER.

NORMAN, D. A. (1983): Some observations on mental models; in: GENTNER, D. A. & STEVENS, A. A.: Mental Models.

NORMAN, D. A. \& DRAPER, S. W. (1986): User-centered system design: New perspectives in human-computer interaction. Hillsdale, NJ: LEA.

O'MALLEY, C. E.; DRAPER, S. W. \& RILEY, M. (1984): Constructive interaction: a method for studying user-computer-user interaction. Tech. Rep. C-015.San Diego: University of California, Institute for Cognitive Science.

YOUNG, R. M. (1983): Surrogates and mappings: two kinds of conceptual models for interactive devices; in: GENTNER, D. \& STEVENS, A. L.: Mental models.

Computer Supported Cooperation

COORDINATION THROUGH MULTIPLE MEDIA

Bignoli C., De Michelis G.
Dept. of Information Science
University of Milano
Via Moretto da Brescia 9
20133 Milano

tel. ++39 - 2 - 7575-221/266
fax. ++39 - 2 - 76110556
E-mail: gdemich@imiucca.bitnet
bignoli@imiucca.bitnet

ABSTRACT

Recent developments within the area of Computer Supported Cooperative Work have proposed new tools to aid their users in cooperating with their colleagues, based on new important insights in the way human beings work together.

Most of the proposed tools are bounded to a particular communication medium. But human experience can not be partitioned on the basis of the media being used. Human beings characterize their affairs on the basis of their contents, not on the basis of the communication media they are using. We believe that a cooperation support becomes much more effective as it helps its users independently of the communication medium they are using.

This paper proposes a new coordination support, designed in the language/action perspective introduced by Fernando Flores and Terry Winograd. Its kernel is an Information Basis, accessible through various Interfaces (electronic mail, face-to-face colloquia support, meeting facilitator support, and finally telephone colloquia support, when the technology will make it possible in a realistic way) which supports conversations carried on through different communication media.

INTRODUCTION

Recent developments within the area of Computer Supported Cooperative Work (or in other terms Groupware) have proposed new tools which help their users to cooperate with their colleagues, based on new important insights in the way human beings work together [CSCW'86], [CSCW'88].

Most of the proposed tools (independently of the fact that they are worked products or still academical prototypes), are bounded to a particular communication medium: either to asynchronous electronic mail (e.g., The Coordinator [Action Tech. 1987], [Flores & Ludlow 1981], [Winograd & Flores 1986], Information- and Object Lens [Malone & others 1986a], [Malone & others 1986b], [Malone & others 1987], [Lai & Malone 1988], as well as the CHAOS System [De Cindio & others 1986], [De Cindio & others 1987], [De Cindio & others 1988], [Bignoli & Simone 1988] in development in our Department) or to

synchronous meetings (e.g., the system under development at the DEC Laboratory on Man-Machine Interfaces, [Whiteside & Wixon 1988], [Marca 1989]), etc.

But human experience can not be partitioned on the basis of the media being used. Human beings characterize their affairs on the basis of their contents, not on the basis of the communication media they are using.

When a person (a professional, a manager, a researcher, etc.) is managing his/her commitments, coordinating his/her activities with his/her colleagues, he/she is generally involved in complex conversations carried on through different communication media. For example I ask a colleague by phone to prepare a document for me. After that we meet at my office to discuss the critical aspects of the document. Afterward we both participate in a meeting where the deadline for the completion of the document is changed. During the preparation phase I exchange messages frequently with my collegue through electronic mail to discuss the first release of the document, etc.

Stephen Reder and Robert Schwab analyzed many workgroups in the last years [Reder & Schwab 1989], [Reder & Schwab 1990], and confirm in a definitive way our belief. They claim that technological support for workgroups will not likely be useful if it will not offer an integrated support to asynchronous as well as to synchronous communication channels.

On the basis of the above considerations,we believe that a cooperation support becomes much more effective as it helps its users independently of the communication medium they are using.

This paper proposes a new coordination support, designed in the language/action perspective opened by Fernando Flores and Terry Winograd [Flores & Ludlow 1981], [Flores 1982], [Winograd & Flores 1986], [Winograd 1988], [Whiteside & Wixon 1988], [De Cindio & others 1988], [De Michelis & others 1989]. Its kernel is an Information Basis, accessible through various Interfaces (electronic mail, face-to-face colloquia support, meeting facilitator support, and finally telephone colloquia support, when the technology will make it possible in a realistic way) wich supports conversations (defined in accordance with Austin and Searle Speech Act Theory [Austin 1962], [Searle 1969], [Searle 1975] and Wittgenstein Linguistic Game notion [Wittgenstein 1953], [Wittgenstein 1969]) carried on through different communication media.

1. THE ARCHITECTURE OF UTUCS

UTUCS (User to User Communication Support) aims to be a support system for a group of people handling conversations carried on through different communication media.

Its global architecture is depicted in Figure 1. Its kernel is an Information Basis, named Current Relation Records, in which the relevant information about commitments are stored. It is shared by various modules, each of which supports the user in communicating through a different medium: E-mail Support, Face-to-face Colloquia Support, Meeting Group Support. When ISDN makes available integration of voice and data communication a new module for Telephone Colloquia Support will be added to UTUCS.

The E-mail Support interacts with a Mail Box, where arriving and departing messages are temporarily stored before (respectively) being linked to the proper conversations or being sent to their destination. A Message Switching System is in charge of all that concerns the physical message exchange. It can be a standard message handling system (e.g. MHS of Novell).

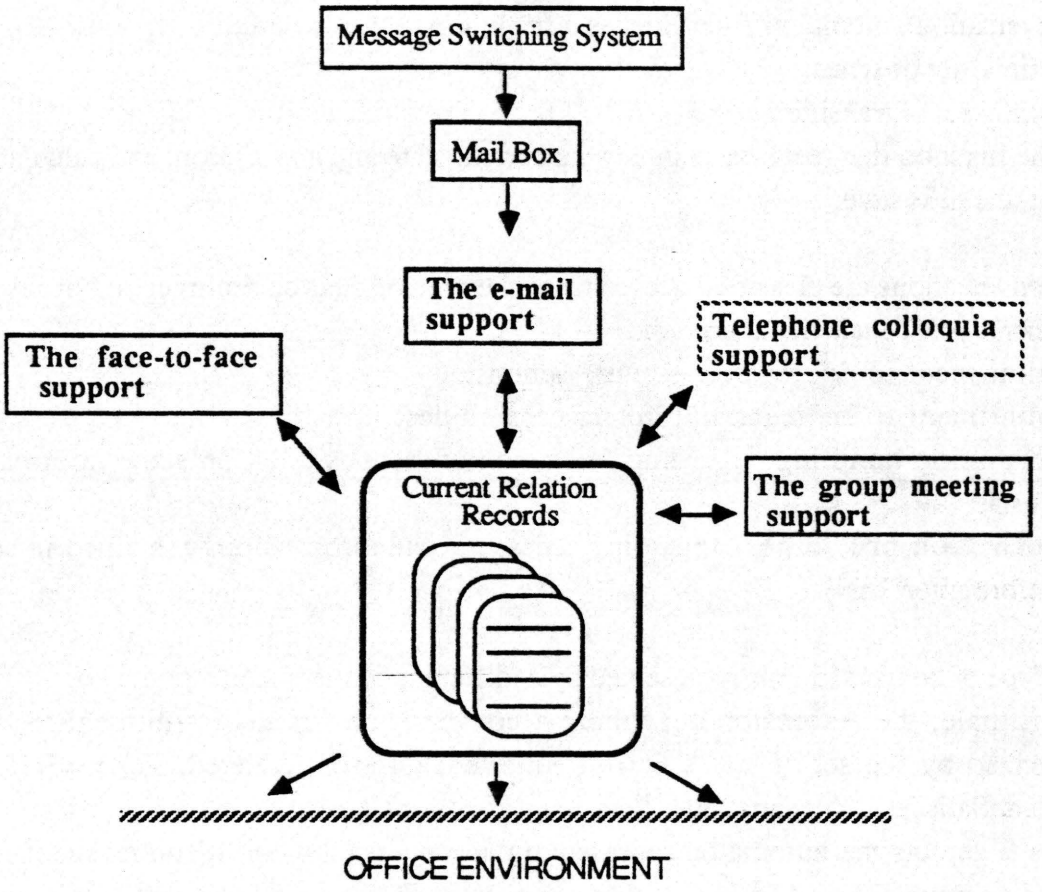


Figure 1

In the following sections, the MSS and to the Mail Box are not discussed. Details of the Current Relations Records and of the various communication supports are given.

2 THE CONVERSATIONS

A conversation is the basic entity in the communication process. Each conversation involves: the actor who opens the conversation, the partner with whom the conversation is going on, and, optionally, a set of users that the actor can designate to have visibility in the conversation (through a distribution list).

The state of any conversation is characterized as the state of the commitment it refers to. Then each conversation is modeled by the possible states of the commitment and by its possible transitions from one state to another one. Formally, we can describe it by an automaton, in line with [Winograd & Flores1986] and [De Michelis and others 1989].

A conversation automaton is defined as a triple **< S, I, Ω>** where:

S is a finite set of states;

I is a finite set of transitions;

Ω is the function that receives as inputs the current state and a transition, and calculates as an output the next state.

The conversations are classified according to the type of their commitments. Four type of commitments will be taken into account:

- **commitment to do**, regarding future actions;
- **commitment to be**, regarding future possibilities;
- **information handling**, regarding delegation of responsibility on some information basis;
- **information providing**, regarding accessing the information contained in some information basis.

Each type of conversation is represented by a different conversation automaton.

For example, the automaton describing a conversation for a 'commitment to do' is characterized by the set of states S = {Requested, Offered, Deleted, Taken, Refused, Concluded, Satisfied, Not satisfied}.

Figure 2 depicts the automaton representing a conversation for a 'commitment to do' between two people A and B. If A makes a request to B, there are many possible states in which the commitment under discussion can be moved from the state 'Requested', depending on the response of B:

- 'Taken' if B accepts doing the action;
- 'Refused' if B refuses the commitment;
- 'Offered' if B counter-offers to A a modified commitment;
- 'Concluded' if B immediately satisfies the request.

Also A can determine a state transition of the commitment from the state 'Requested' to the state 'Deleted', if after the request he changes opinion for some reason and withdraws.

Notice that A keeps the commitment in the state 'Requested' if, while waiting for an answer, he solicits his interlocutor.

In the state 'Concluded' the commitment is evaluated by A either as satisfactory or not. Then the possible final states are "Satisfied" or "Not satisfied".

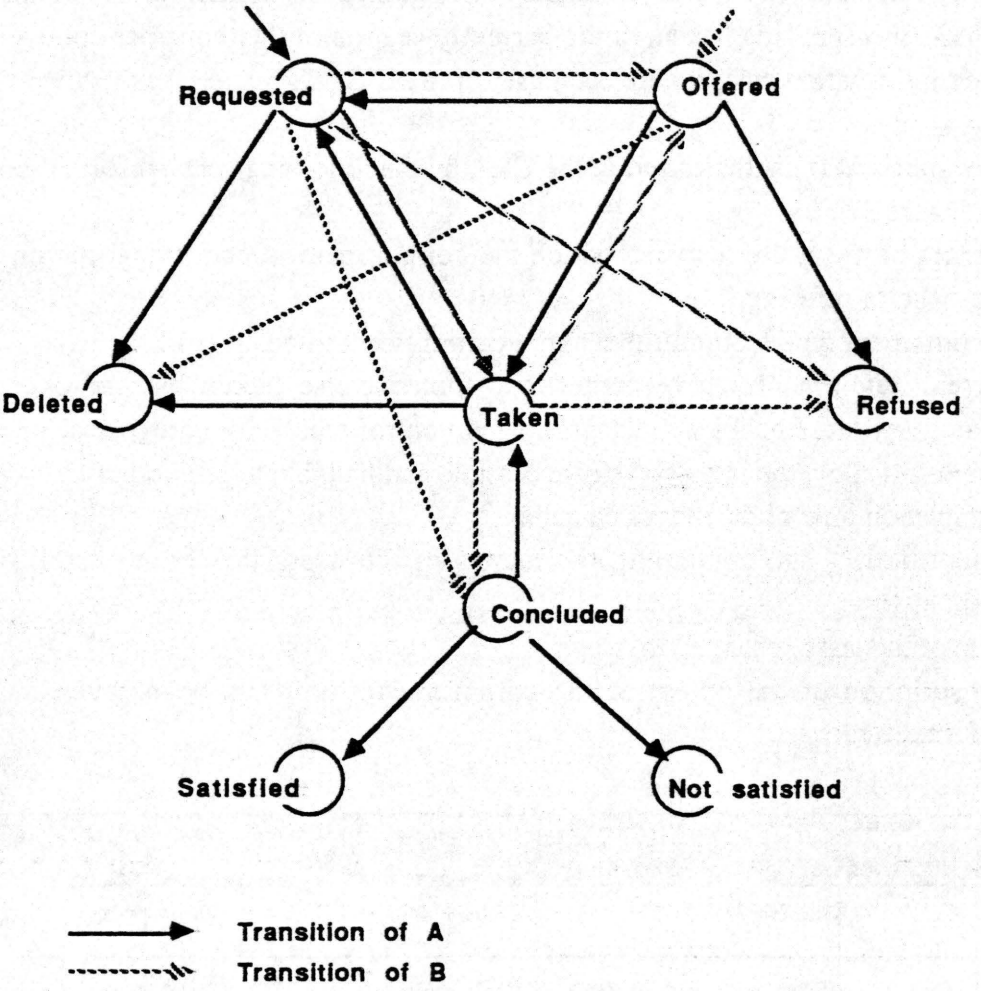


Figure 2

The conversation automaton of Figure 2 also depicts all the flow of information for the case of a conversation opened by B making an offer to A to do something.

This conversation automaton is independent of the communicative medium used for making the transitions between states, i.e. the automaton describes a conversation, independently of whether it is carried on through a synchronous communication medium (as

a face-to-face meeting) where speaking and hearing, or sending and receiving, are simulateneous, or through an asynchronous one (as an electronic mail) where speaking and hearing follow each other in an independent way.

3. THE CURRENT RELATION RECORD

The Current Relation Record (CRR) is the information basis containing the commitments that the UTUCS's users have taken, or are currently negotiating with his/her colleagues. It is a distributed information basis where each user of the UTUCS System has its own CRR.

Each commitment is characterized in the CRR by the following set of information (as can be seen in Figure 3):

- the names of the users between which the commitment is under negotiation (who is requesting, who is offering, ...);
- the commitment type (commitment to do, commitment to be, ...);
- its current state ('requested' by someone to someone else, 'taken' by someone , ...).
- its category set. Each user can group his commitments by categories, in order to classify them and allow them to be easily accessible and linkable to each other;
- the completion time of the requested action;
- the 'distribution list' containing the names of the users that have visibility in the commitment history;
- the date of the last exchanged message;
- the description of the object of the commitment, as it has been given in the last exchanged message;

Object	Addressee	State	Category	Completion time	Distribution list		Date	Support
Writing an offer...	Paul	Requested by	...	January 30 1990	0	●	10/06/90	Face-to-face
Preparing a meeting about ITHACA	John	Taken by	Esprit-Project	December 1989	(Ann, Mary, Paul)	●	07/02/90	e-mail

(list containing the whole commitment history)

(list containing the whole commitment history)

Figure 3: The Current Relations Structure

- the support through which the last exchanged message was created (the e-mail support, the face-to-face support, ...);
- a pointer to a list containing the history of the conversation, in terms of the records characterizing all the previously exchanged messages.

The CRR is the kernel of the UTUCS System. It is the the information basis integrating the various communication supports the system provides. All the CRR of the UTUCS's users are maintained mutually consistent .

4. THE COMMUNICATION SUPPORTS

How communication is supported depends on the media through which it is carried on. In fact , when we use an e-mail system, it is required that the system takes care of the transfer of the message from its source to its destination. During a group meeting, or during a face-to-face colloquium, the conversations do not need any support to be carried on.

What is common to all the communication supports of the UTUCS System is their integration at the level of the CRR modules, where all of the supports can update and access the commitments of each user.

But each communication support of the UTUCS System is designed to help its users in the communication situation in which it is used. The functionalities of each communication support are therefore a combination of a common component using the CRR module, and other components that are tailored for the communication situation.

4.1 The e-mail support

The e-mail support is an e-mail system structured in terms of conversations. It can be considered as a 'The Coordinator'-like system [Action Tech. 1987], fulfilling the requirement indicated by Christin Bullen and John Bennett, "Message linking is a key improvement provided by electronic communication", [Bullen & Bennett 1990].

The automaton representing a 'commitment to do - conversation' between A and B in Figure 2 can be considered as the diagram of the state transitions of a conversation carried on through the e-mail support of UTUCS.

Figure 4 shows the complete scheme of the possible state transitions. Figure 5 shows the speech acts that don't change the states.

Present state	Input value	Next state
Requested	B counter-offers	Offered
	B declines	*Declined
	B accepts	Taken
	B concludes	Concluded
	A cancels	Deleted
Offered	A counter-requests	Requested
	A accepts	Taken
	A declines	Refused
	B withdraws	Deleted
Taken	A cancels	*Deleted
	B withdraws	*Refused
	B concludes	Concluded
Concluded	A refuses	Taken
	A evaluates	*Satisfied
	A evaluates	*Not satisfied

* Terminal states

Figure 4

Present state	Input value	Next state
Proposed	A solicits	Proposed
	B acknowledges	Proposed
Suggested	B solicits	Suggested
	A acknowledges	Suggested

Figure 5

When a user sends a message, the message is temporarily stored in a memory register. When the message is read at its destination, the CRR of the two participants to the conversation are updated accordingly. Due to this updating protocol, that preserves the consistency of the CRR modules of all the users of the UTUCS System, the e-mail support has the capability of solving the conflicts that can occur because of the asynchronicity of the communciation. If both the participants in a conversation send each other a message changing the state of the commitment, then the e-mail support avoids any inconsistency, canceling one of the two messages.

4.2 The face-to-face support

The main module of the face-to-face support is a 'minutes' component. It can be opened

by both the participants in the colloquium with their passwords, and allows them to record the commitments that they take on during the colloquium itself. At the end of the session (at the end of the colloquium) the minutes component updates the CRR of both the participants.

The minutes of any face-to-face colloquium are also stored in a file, together with the links to the CRR component. Any user of the UTUCS System can recall the minutes of a previous meeting, having access also to the successive state transformations of the commitments negotiated within that meeting.

The minutes-component does not support directly the conversations of the participants in a face-to-face colloquium. It is used only at the end of the colloquium. Therefore no restriction is imposed on its users at the level of the state transitions of a commitment.

For example, during a face-to-face colloquium the users can record in the 'minutes' component any new commitment in the 'Taken' state, without simulating or representing their whole conversation.

4.3 The group meeting support

The main module of the group meeting support is also a 'minutes' component. It has the same functionalities of the correspondent component of the face-to-face support. It allows to record, in real time, the commitments taken during the meeeting. It updates the CRR module of each participant, adding the new commitments taken within the meeting or modifying commitments already existing. It allows to recall the minutes of any previous meeting, and so to access to all the successive modifications occurred to the commitments taken within that meeting.

Following the guidelines proposed by John Whiteside, Dennis Wixon and David Marca of Digital Equipment Corporation, who designed some prototypes of electronic meeting support [Whiteside & Wixon 1988], [Marca 1989], our group meeting support is based on two assumptions:

i) it is a support of the meeting coordinator (or,better of the meeting facilitator) Because that electronic support gives the best performances when the meeting is conducted by a person dedicating his/her best attention to the success of the meeting itself. It does not provide any individual electronic support to the meeting participants;

ii) an easy and cheap integration with an overhead projector allows to display on-line to the participants the current minutes of the meeting So they can discuss them carefully, and modify them so that they correspond in the best way to the reached agreements, increasing the ability of the group to focus on critic issues, and to interact in a collaborative way .

Furthermore, the integration of this module within the UTUCS System makes possible a more effective conduct of the intervals occurring between one meeting and the following one. UTUCS integrates in fact in the best way the collaboration within the group meeting

with the conversations occurring via electronic mail or face-to-face meetings during those intervals.

5. CONCLUSION

The UTUCS System will be a component of the office environment that is under development within the ESPRIT-2 Research Project , Nr. 2121 ITHACA, of the European Economical Community .

Its integration within that environment open some interesting possibilities in the perspective of offering a common and well integrated support to routine and exceptional work within offices, as well as for what concerns the enrichment of the offered functionalities on the basis of the office model underlying the office environment itself.

The exploitation of these possibilities is our major concern within the deveopment of UTUCS.

ACKNOWLEDGMENTS

The authors wish to thank Carla Simone, Renata Tinini, Giuseppe Omodei Sale' and Fiorella De Cindio, that contributed greatly to the development of the UTUCS System.

Many thanks are also due to Michel Tueni and Janzhong Li of Bull, for the assistance in using the AMS prototype.

REFERENCES

Action Technologies Inc., *The Coordinator Workbook & Tutorial Guide*, AT, Emeryville, 1987

Austin J.L., *How to do Things with Words*, Harvard University Press Cambridge, 1962

Bignoli C., Simone C., *AI Techniques for supporting human to human communication in CHAOS*, in: Proc. EC-CSCW'89, CSC, London, 1989

Bullen C. V., Bennett, J. L., *Groupware in Practice: An Interpretation of Work Experiences*, MIT Center for Information Systems Research, Cambridge, MA, 1990

CSCW'86, *Proceedings of the Conference on Computer-Supported Cooperative Work*, MCC, Austin, 1986

CSCW'88, *Proceedings of the Conference on Computer-Supported Cooperative Work*, ACM, New York, 1988

De Cindio F., De Michelis G., Simone C., *GAMERU, a Language for the Analysis and*

Design of Human Communication Pragmatics within Organizational Systems, in Advances on Petri Nets'86, LNCS 266, Springer Verlag, Berlin, 1987

De Cindio F., De Michelis G., Simone C., *The Communication Disciplines of CHAOS*, in: K. Voss, H. J. Genrich, G. Rozenberg (eds.), Concurrency and Nets, Springer Verlag, Berlin, 1987

De Cindio F., De Michelis G., Simone C., *Computer-based Tools in a Language/Action Perspective*, in: Proc. EUTECO'88, North Holland, Amsterdam, 1988

De Cindio F., De Michelis G., Simone C., Vassallo R., Zanaboni A., *CHAOS as a Coordination Technology*, in: CSCW'86

De Cindio F., Simone C., Vassallo R., Zanaboni A., *CHAOS: a Knowledge-based System for Conversing inside Offices*, in: Office Knowledge: Representation, Management and Utilization, North Holland, Amsterdam, 1988

De Michelis G., *La complessita' delle organizzazioni*, Sviluppo e Organizzazione 110, 1988

De Michelis G., *Sistemi sociali come giochi linguistici*, in: A. Ardigo', G. Mazzoli (eds.), L'ipercomplessita' tra socio-sistemica e cibernetiche, Angeli, Milano, 1990

De Michelis G.,Bignoli E., Simone C., Tinini R., *User-To-User Communication Support: Updating the State of the Art and (1st Release) Specifications of the System*; ITHACA Tech. Report, 1990

De Michelis G., De Cindio F., Simone C., *Groups in the Language/Action Perspective*, in: Proc. II European Meeting on Cognitive Science Approaches to Process control, JRC, Ispra, 1989

Flores C.F, *Management and Communication in the Office of the Future*, Hermenet, San Francisco, 1982

Flores C.F., Ludlow J.J., *Doing and Speaking in the Office*, in: F. Fick, R. Sprague (eds.), DSS: Issues and Challenges, Pergamon, New York,1981

Lai K., Malone T., *Object Lens: a "Spreadsheet" for the Cooperative Work*, in: CSCW'88

Maturana H., Varela F., *Autopoiesis and Cognition*, Reidel Dordrecht, 1980

Malone T.W., Grant K., Turbak F., *The Information LENS: an intelligent system for information sharing in organizations*, in: Proc.of the CHI 86 Conference on Human Factors in Computing Systems, Boston, 1986

Malone T.W., Grant K.R., Lai K-Y, Rao R., Rosenblitt D., *Semi-structured messages are surprisingly useful for Computer-supported Coordination*, in: CSCW'86

Malone T.W., Grant K., Turbak F., Brobst S.A., Cohen M.D, *Intelligent Information-Sharing Systems*, Comm. ACM 30.5, may 1987

Marca D., *Experiences In Building Usable Meeting Support Software*, Proc. IFIP Groupware Technology Workshop, Palo Alto, CA, 1989

Reder S., Schwab R. G., *The Communicative economy of the work group: Multi-channel genres of communication*, Office: Technology & People 4.3, 1989

Reder S., Schwab R. G., *Selection and Effects of Channels in Distributed Communication and Decision Making Tasks: Final report to the US Army Research Institute*, Northwest Regional Educational Laboratory, Portland, OR, 1990

Searle J.R., *Speech Acts: an Essay in the Philosophy of Language*, Cambridge University Press, Cambridge, 1969

Searle J.R., *A Taxonomy of Illocutionary Acts*, in: K. Gunderson (ed.) Language, Mind and Knowledge, University of Minnesota, Minneapolis, 1975

Tueni M., Li J., Fares P., *AMS: a knowledge based approach to tasks representation manipulation and organization*, in: Proc. COIS-88, Palo Alto, 1988

Whiteside J., Wixon W., *Contextualism as a World View for the Reformation of Meetings*, in: CSCW'88

Winograd T., *A Language/Action Perspective on the Design of Cooperative Work*, Human Computer Interaction, 3.1, 1988

Winograd T.,Flores C.F., *Understanding Computers and Cognition*, Ablex, Norwood, 1986

Wittgenstein L., *Philosophischen Untersuchungen*, Blackwell, Oxford, 1953

Wittgenstein L., *On Certainty*, Blackwell, Oxford, 1969

A Model of Visual Language for Cooperative Problem Solving(*)

P.Dell'Olmo, E. Nardelli

Istituto di Analisi dei Sistemi ed Informatica, C.N.R.,
V.le Manzoni 30, 00185 Rome, Italy
Phone & Fax +39-6-770031
E-mail dellolmo@irmiasi.bitnet

M. Talamo, P. Vocca

Dipartimento di Informatica e Sistemistica,
Università di Roma "La Sapienza",
Via Buonarroti 12, 000185 Rome, Italy
Phone +39-6-7312367

Abstract

In this paper a model of a visual language for cooperative problem solving is presented. A *game metaphor* is introduced in order to provide at user level an high degree of abstraction from the complex representation of the decision making process internal to the computer system. In the game metaphor domain the experts can apply in a direct and synthetic way differents *problem resolution strategies* by means of a *visual language* which completely hides the underlying heuristic search procedures mechanisms. The whole resolution process can be so driven by the experts quite independently from each other, but in a common synthetic visual representation that allows an effective cooperation to be performed.

The completeness of the program generated by the experts' commands is mantained by a set of constraints defined on the internal state space representation of the decision-making problem (AND-OR tree), so ensuring the convergence of the whole search process until a feasible solution is reached. None of such internal constraints are visible at metaphor level and whatever choice of *play strategy* is made possible for the expert, being care of the system performing the suitable internal application of experts' choices.

Such a high-level cooperation between experts and a computer system during the execution of a decision making process seems to be a promising approach for obtaining "good" solutions of intractable problems of practical

---

(*) This work has been partially supported by "Obiettivo MULTIDATA" of Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of C.N.R.

size. Note that the dynamic graphical presentation of the complex problems instances and of the resolution strategies behavior presented in the game metaphor, can increase the user knowledge of both the problem and the resolution techniques. Moreover the well-known decision-making mental models based on initial, intermediate and finale states can be effectively implemented in the such a metaphor and augmented by the user driven selections of successive intermediate states.

The formal model of the man-machine interface supporting the metaphor and the language introduced is described and a schema of the architecture of the global system is given. Finally we discuss the experimental results obtained in an environment planning context, where the problem of optimal location of a set of services in an area subject to a number of constraints is used as testbed application.

1. Introduction

Many complex systems consist of relatively simple components but with a complex behaviour (time constants spanning several orders of magnitude and multiple causal interconnections) [Hop87]. Thus it is difficult to predict system behaviour from a detailed knowledge of its parts; some integrative understanding of the system is required. The consequence is a class of problem solving domains called by Simon [Sim81, BhS77] *semantically rich problem solving domains*. These are contexts in which prior semantic knowledge and subproblems specific information play crucial roles in effective problem solving. Examples of these kind of problems can be found in [HCP87, MUB83, BNG89, Bur89].

The possibility of using the above specific knowledge depends on user's understanding of the complex problem solving systems behaviour and on the possibility of interacting with the system itself.

In [DNTV89] the architecture of a multimedia interactive environment (a *Cooperative Problem Solver*) is proposed which allows to incorporate the human in the problem solving process and, as a consequence, to improve the overall system [JMW89].

In this paper the model of a visual language for cooperative problem solving is presented which maps a set of primitives for manipulating both problem and resolution strategies onto a sequence of user actions at the interface level. This set of primitives allows to describe and manipulate, by means of suitable visual metaphors the internal state space representation of the problem solver and, as a consequence, to ensure the convergence of the whole search process until a feasible solution is reached.

Such a high-level cooperation between the domain experts and a computer system during the execution of a problem resolution process seems to be a

promising approach for obtaining "good" solutions of intractable problems of practical size [GaJ79]. Note that the dynamic graphical presentation of the complex problems instances and of the resolution strategies behaviour presented in the interface metaphors, can increase the user knowledge of both the problem and the resolution techniques [MUB83].

The paper is organized as follows: section 2 described the problem formal model which constitutes the starting point for the primitives definition; metaphors for representing at interface level the underlying concepts are sketched in section 3 and discussed in section 4; in section 5 problem solving primitives are defined and the completeness of the proposed visual language is shown; in section 6 experimental results are presented and discussed; finally, section 7 presents conclusion and future research.

2. Problem Model

The class of problems examined in this paper from the point of view of the strategies for cooperative, interactive solution is that of problems solvable by reduction to subproblems. Such problems (called reduction-problems in the following) are solved by an iterative reduction process which uses specific operators to transform a given (sub)problem to (sub)subproblems which have a lower degree of complexity. This process terminates when the original problem is completely decomposed in primitive problems, i.e. problems whose solution is known.

Reduction-problems can be suitably represented by a structure called AND/OR graph [Nil71, Pea84]. To search for a solution of a reduction-problem means to search for a particular subgraph of the AND/OR graph.

Definition: A *directed hypergraph* G=(V,E) is defined by an ordered pair (V,E), where V={1,...,n} is the set of *nodes* and $E \subseteq V \times P(V)$ (P(V) is the powerset of V) is the set of *hyperlinks*. An hyperlink is defined by an ordered pair $\langle s, t \rangle$ where s is the *source node* and t is the set of *target nodes*.

An *AND/OR graph* is a particular hypergraph where we distinguish between two kinds of hyperlinks, the *OR hyperlinks* and the *AND hyperlinks*. An OR hyperlink is an ordered pair $\langle s, t \rangle \in V \times V$ and the other hyperlinks are the AND hyperlinks. Therefore, if we remove the AND hyperlinks we obtain an ordinary directed graph *(OR graph)*.

In reduction-problem the subproblems are represented by nodes and a special node, the *start node*, is the original problem. Each OR hyperlink represents an alternative approach in handling the problem node from which it emanates, while each AND hyperlink connects a parent problem to the subproblems which compose it.

A complete solution of the original problem is represented by an AND/OR subgraph G', called *Solution Subgraph*, which has the following properties:

1) G' is connected;
2) G' contains the start node;
3) all its terminal nodes (nodes without successors) are primitive subproblems, i.e. problems whose solution is known.

More formally, the solution of the original problem is an hyperpath between the start node and the set of primitive terminal nodes.

In general, the AND/OR graphs are too large to be explicitly represented in computer's memory and to be searched for finding a solution, and moreover the generation of the whole graphs can be too expensive. Therefore, the graph can be generated incrementally from the start node and *expanding* (i.e. *generating* all the successors) some suitable chosen nodes. A node expansion is performed by applying particular *operators* which allow to transform the problem represented by the node to be expanded into a set of subproblems. The set of operators available depends on the specific class of problems the system is dealing with.

3. Metaphors and Language

In order to improve user's understanding of the complex problem resolution process performed by the system, and as a consequence to allow experts to interact between them and with the system itself, a cooperative problem solver system should provide an high-level man-machine interface giving a synthetic but expressive representation of the resolution process while hiding the underlying resolution mechanism [CaT82].

In [DNTV89] two kinds of metaphors were sketched; namely *Problem Representation Metaphor* (PRM) and *Problem Perception Metaphor* (PPM). PRM defines a way of representing and rendering to the user the entities, the relations among entities and the operations defining the static structure of the class of problems the system is dealing with (*structural mechanisms*).

On the other hand, by means of PPM it is possible to define the dynamic relations between the entities (*evolution process*) while describing the strategical aspects of the problem's domain (i.e. the non predictable evolution of the complex problem during the resolution process).

The information underlying the PPM can be obtained, for example, by means of a set of resolution process estimators [Voc89, NTV89], which allow to evaluate the resolution process behaviour and as a consequence to individuate relevant strategies.

By means of a set of multimedia functionalities the above introduced metaphors can be presented, at user level, in a way satisfying the criteria of syntheticity, analogy, effectiveness [DNTV89].

In order to allow the user to efficiently interact with the problem solver at the metaphor level, a visual language has to be defined which introduces a set of primitives for manipulating objects at the interface and maps the set of introduced visual primitives for manipulating both problem and resolution strategies onto a sequences of manipulation applied at the level of problem and strategy representation.

In the next section a set of primitives of the visual languages is defined, and the completeness of the introduced primitives, with respect to the need of identifying a solution in the state space representing the problem, is proven.

4. Description of the game metaphor

Let us denote with $\mathbb{X}$ the state space of the problem and of the heuristic which is trying to solve it, that is the space spawned by the vector $X=(x_1, x_2, ..., x_m)$ whose m components completely describe with their current values the current state of the problem solving process. Let us denote with $\mathbb{S}$ the physical space used to represent, according to chosen Problem Representation and Perception Metaphores, the state space of the problem solving process.

A point S in the space $\mathbb{S}$ correspond to a point X in the space $\mathbb{X}$ according to a monitor function $S = \mathbb{F}(X)$ [DNTV89], whose purpose is to sinthesize and aggregate raw data regarding the problem solving process. Thus, the sequence of points $X_1, X_2, ..., X_n$ which describes in the space $\mathbb{X}$ the evolution of the problem solving process is seen by the user as a sequence of point $S_1= \mathbb{F}(X_1), S_2= \mathbb{F}(X_2), ..., S_n= \mathbb{F}(X_n)$ in the physical space used for representing to the user this evolution.

Points S are physically rendered to the user by means of a set of attribute value(s) T, which is computed using another monitor function $T = \mathbb{G}(X)$. Such a function $\mathbb{G}$ computes points T in the evaluation space $\mathbb{T}$, which is used to measure the quality of points X as solution for the problem. Thus, the complete definition of monitor functions $\mathbb{F}$ and $\mathbb{G}$ completely defines the Problem Representation Metaphore. In our approach we have used as $\mathbb{S}$ the bidimensional space corresponding to the monitor screen and as $\mathbb{T}$ the tri-dimensional space corresponding to color definition according to HSV model (i.e. hue, saturation, and value).

Our Problem Perception Metaphore assumes that, in each moment a line ("highway line") connects in $\mathbb{S}$ S_0 to S* (where S* corresponds to the state

with the higher quality in terms of problem solution). Highway line is calculated by modulating a base straight line from S_0 to S* with gaussian noise, whose standard deviation is larger the bigger has been the "difficulty" for heuristic which is guiding the solution process in reaching S* from S_0. When a new state S' is reached with a quality higher than S*, then the extreme of the base straight line is moved from the old S* to S' (which now becomes the new S*), the value of standard deviation is again computed, and a new highway line is displayed.

We can now define the *game metaphor* as it follows. Let use assume that heuristic which is working on problem instance is driven by k predicates P_1, P_2, ..., P_k. We shall partition physical representation space (i.e. the screen) in two sections.

The first section is partitioned in Q "basic physical presentation spaces" (BSs). In each one of this spaces the evolution of the problem is represented when the heuristic is under the influence of one single predicate. Thus we can observe, for Q different predicates, the effect they singularly have on the evolution of the problem.

In the second section we define R "combined physical presentation spaces" (CSs). In each CS the evolution of the problem when the heuristic is under the influence of two or more parameters is represented.

At the beginning user selects among the k predicates the Q he/she wants to observe. The evolution is shown in parallel on the Q spaces. Then the user begin to assign to CSs combination of predicates, chosen among the Q selected in the first section. Each time a combination is chosen and assigned to one CS, the joint effect of predicates is shown and the user can thus compare the effect of combined predicates to that of single one and evaluate how well the combination is performing.

This process can also be iterated. Suppose that predicates P_a and P_b have been combined, and user is observing a BS for P_a, a BS for P_b, and a CS for $P_{a,b}$. Now a further parameter can be added, let us say P_c, and a new CS is activated. The user is thus being able to compare the behaviour of the heuristic being jointly affected by P_a, P_b, and P_c to previous ones.

The effect of combining two basic spaces can be formally described as it follows. Let us denote with S_a and S_b the temporal sequences of states in the representation space $\mathbb{S}$ generated by heuristic being affected by only predicates P_a and P_b, respectively. Sequence S_a is now reexamined by taking into account predicate P_b and only states which are feasible with respect to P_b are retained. Simmetrically, sequence S_b also is reexamined and only states which are feasible with respect to P_a are retained. Furthermore, a new best solution S* is computed and the new highway line is displayed.

The objective of the *game metaphor* is to obtain an highway line as straight as possible.

By looking at shapes, relative and absolute positions, and quality, of the two basic spaces and of the combined one user can thus, by means of its high expertise in terms of graphical patterns which emerge, evaluate in a synthetic and analogic way if the combination he/she is trying leads to a better highway line or not.

5. Language primitives and their power

At the interface level users therefore interact with a set of visual object (called *monitors* in [DNTV89]) which represent the current state of the problem. The state of the problem is hence represented at the interface level by a set of *visual objects* $O_1, O_2, ..., O_n$.

Given a primitive X of the visual language, the effect of its application to a certain state $G = (O_1, O_2, ..., O_n)$ of the interface is the new state of the interface defined by the set of visual objects $X(G)=X(O_1), X(O_2), ..., X(O_n)$.

Given a sequence of primitives $X_1, X_2, ... X_m$ the effect of its application to a certain state $G = (O_1, O_2, ..., O_n)$ is defined by $(X_m(X_{m-1}(...(X_1(G)...)))$.

A sequence of primitives is said to be *acyclic* if does not exist two indexes i and j of the sequence such that $X_i(G') = X_j(G'')$.

A set of primitives is said to be *acyclic*(G',G'') if it does admit at least one acyclic sequence transforming state G' in state G''.

A set of primitives is said to be *acyclic* if, given the initial state G°, it is acyclic(G°,G'') for each possible choice of of G'' among the solution states.

According with the formal model chosen for problem representation, the following primitives of the visual language have been defined:

A: **ChooseObject(O_i)**: given a set of visual objects $O_1, O_2, ..., O_n$ at the interface level this primitive allows to make it the subproblem corresponding to O_i the one which has to be analyzed next.

B: **ChooseOperator(P)**: given the set of operators available for the problem under consideration, this primitive allows to choose operator corresponding to P as the one to be applied for the next visual object's expansion.

C: **ExpandObject(O_i)**: given a set of visual objects $O_1, O_2, ..., O_n$ at the interface level this primitive allows to make it the subproblem corresponding to O_i the one which has to be expanded next, that is that one which next has to be decomposed in subproblems.

The following theorem is proved in [DNTV90].

Theorem 1:

The set of primitives A, B, C, above defined is acyclic with respect to the class of reduction problems defined in section 2.

Theorem 1 ensures us that there exist "safe" sequences of applications of the primitives of the visual language. That is not enough, since we want to be guaranteed of their power. We have therefore to introduce some additional definitions, as follows.

If a problem instance Q admits a feasible solution, we shall say that a sequence of primitives $X_1, X_2, \ldots X_m$ is *complete with respect to a state* G' *of* Q if $(X_m(X_{m-1}(\ldots(X_1(G')\ldots))) \supseteq G^*$ where G^* is the set of visual objects corresponding to a solution of the problem under examination.

We shall say that a set of primitives $X_1, X_2, \ldots X_l$ is *complete* if for each initial state G° relative to a problem instance Q' admitting a solution there exists at least one finite sequence $X_{i1}, X_{i2}, \ldots X_{ik}$ such that $X_{i1}, X_{i2}, \ldots X_{ik}$ is complete with respect to G°.

The following theorem, also proved in [DNTV90] establishes the desired result in terms of power of the primitives of the visual language.

Theorem 2:

The set of primitives A, B, C above defined is complete with respect to the class of problems defined in section 2.

6. Experimental results

6.1. Problem definition

The problem is to find the optimal location for a set of facilities in a region subject to many constraints. It is usually modeled as an integer linear programming problem, that is a linear programming problem where only integer-valued solutions are accepted. Such problems are difficult to solve because general efficient algorithms do not exist.

A practical approach for real life instances of integer linear programming problems is first to find and approximate real-valued solution (for which efficient algorithms exist) working only on a suitable subset of the

variables, then to truncate all obtained values to integer numbers, and finally to try to optimize the overall solution by analyzing the subset of variables not considered in previous real-valued approximation (free variables). This analysis is usually done by enumerating possible different combination of integer values of free variables according to a number of strategies. These strategies decide if certain combinations of variable values are taken into account or not and can be characterized in terms of predicates specifying combination of variables not to be examined.

6.2. Software and hardware environment

A prototype of the user-interface for a system for visual interactive problem solving [DNTV89, NTV89] based on the game metaphor above described has been implemented on an APOLLO 10000 workstation.

An implementation of the above described mathematical model as a batch system with an alphanumeric interface was used for simulating the underlying problem solving system.

A set of eight postgraduate students in environmental planning which knew used mathematical model were asked to test the implemented prototype.

The main objective of such a testing was to verify that the defined framework for visual interactive problem solving is consistent, that is it is able to guarantee at least the same performances as a more traditional alphanumeric interface.

Each student went through a set of interactions with five different instances of the problem, randomly generated, using the system with alphanumeric interface in a batch mode. Histories of interaction with the batch system were then used to simulate the behaviour of an interactive system. Each student then went through another set of interactions with five instances, different from the five ones he/she previously examined, this time using the prototyped visual interface.

6.3. Description of the experiment

The interface used for testing was done as it follows. Two CSs were defined, each one with a size of 15x12 cm, located in the upper part of the screen. In the lower part eight BS were defined, each one with a size of 8x8 cm.

From the point of view of physical rendering the quality of solutions a color scale from dark greyish green (worse quality solutions - low values of saturation and lightness) to bright yellow (best quality solution - high values of saturation and lightness) was used. Thus points S were rendered on the screen according to such a color scale as a function of their quality.

6.4. Measurement parameters

Overall quality of found solutions for the forty instances of the problem in batch and interactive mode was then compared. Overall quality of a solution is measured by a function which takes into account both the number of states examined by the problem solving strategy to find the solution and the distance of the found solution from the optimal one.

It was found that in 73% of cases overall quality of solutions, measured with the previous described function, was significantly better (more than 10%) in the interactive case than in the batch one, in 19% of cases solutions of the same quality (i.e. with a 10% higher or lower quality) were found, while in the remaining 8% of cases interactive interface performed worse.

7. Conclusions

In this paper a visual language for cooperative problem solving has been presented. The concept of interface metaphors has been introduced in order to provide at the user level an high degree of abstraction from the complex representation of the problem resolution which is internal to the computer system. The completeness of the introduced visual language with respect to the resolution process was shown.

The formal model of the man-machine interface supporting the metaphor and the language introduced is described and a schema of the architecture of the global system is given. Finally the experimental results obtained in an environment planning context were discussed, where the problem of optimal location of a set of services in an area subject to a number of constraints is used as testbed application.

Bibliography

[BhS77] R. Bhaskar and H.A.Simon, "*Problem-Solving in Semantically Rich Domains: An Example from Engineering Thermodynamics*", Cognitive Science, vol 1, 1977.

[BNG89] M. L. Brown, S. L. Newsome, E. P. Glinert, "*An Experiment into the Use of Auditory Cues to Reduce Visual Workload*", Proc. of CHI'89, May 1989.

[Bur89] S. A. Burns, "*Graphical Representation of Design Optimization Process*", Computer Aided Design, 21, 1, January 1989.

[CaT82] J. M. Carroll, J. C. Thomas, "*Metaphor and the Cognitive Representation of Computing System*, IEEE Trans. on System, Man and Cybernetics, 12,1982.

[DNTV89] P. Dell'Olmo, E. Nardelli, M. Talamo and P. Vocca, "*A Mixed-Mode Man Machine Interface for Interactive Problem Solving*", international Working Conference on Workstation for Experiments, IFIP WG 5.10, to be published on Lecture Notes on Computer Science, Springer Verlag, 1989.

[DNTV90] P. Dell'Olmo, E. Nardelli, M. Talamo and P. Vocca, "*Formal properties of a visual language for cooperative problem solving*", Tech. Rep. R., Dipartimento di Informatica e Sistemistica, Università di Roma, 1990

[GaJ79] M. R. Garey and D. S. Johnson, *"Computer and Intractability"*, San Francisco: W. H. Freeman, 1979.

[HCP87] R.H. Hopkins, K.B. Campbell, N.S. Peterson, "*Representation of perceived relations among the properties and variables of a complex system*", IEEE Trans. on System, Man and Cybernetics, SCM 17, 1987.

[Hop87] J. Hopcroft, "*Algorithmic Problems in Modeling and Electronic Prototyping*", Discrete Algorithms and Complexity,1987.

[JMW89] V.S. Jacob, J.C. Moore, A.B. Whinston, "*An Analysis of Human and Computer Decision-Making Capabilities*", Information & Management, 16, 1989.

[MUB83] T. M. Mitchell, P. E. Utgoff, R. Banerji, "*Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics*", in Machine Learning: An Artificial Intelligence Approach, R. S. Michalski, J. G. Carbonell and T. M. Mitchell ed., Tioga Pub. Comp., Palo Alto, CA,1983.

[Nil71] N. J. Nilsson, "*Problem Solving Methods in Artificial Intelligence*", MacGrow-Hill, 1971.

[NTV89] E. Nardelli, M. Talamo, P. Vocca, "*Problem solving by interactive guide of heuristics*", in Proc. of IV International Symposium on Computer and Information Sciences, Turkey, 1989.

[Pea84] J. Pearl, "*HEURISTICS: Intelligent Search Strategies for Computer Problem Solving* ", Addison-Wesley,1984.

[Sim81] H.A.Simon, "*The Science of the Artificial*", 2nd ed. Cambridge, MA: MIT Press, 1981.

[Voc89] P. Vocca, "*Animazione di algoritmi di visita guidata per grafi AND/OR*", in Proc. of 1989 National Conference of Italian Association for Automatic Calculus (AICA), Trieste, 1989.

Developing usability integration principles for the design of IBCN systems

Anthony Esgate, Andy Whitefield, and Andrew Life

Ergonomics Unit, University College London, 26 Bedford Way, London, W. C. 1. H. 0. A. P., United Kingdom.

This paper arises from work performed on the Guidance (Generic Usability Information for the Design of Advanced Network Communications in Europe) project, funded by the European Community's RACE (Research in Advanced Communication technologies in Europe) programme. The first section introduces the Guidance project. Since Guidance is primarily concerned with the provision of principles to designers, the nature of extant, non-Guidance principles is discussed in Section 2 followed, in Section 3, by a discussion of Guidance principles, their development, and their application. Future work is outlined in Section 4.

1. Introduction

The Guidance project is concerned with the provision of usability information, in the form of integration principles, to the developers of interactive IBCN (integrated broadband commmunication network) systems.

1.1 Objectives

The principal objective of the project is to develop and evaluate sets of generic and specific usability integration principles for IBCN systems. The task domain addressed is that of multi-media, multi-author document production (MMMADP). This is instantiated as a concrete example in the reading and writing of travel agency documents providing information about holidays. The principles will be both evaluated by means of empirical investigation and demonstrated in practice during the development of a user interface to a multi-media, multi-author, travel agency document system. It is a further aim of the project to present the principles in a delivery vehicle appropriate for use by designers.

1.2 The Guidance design process

Fig. 1 illustrates a simplified form of the design cycle employed within Guidance. In the requirements stage, an initial description of the task (reading and writing multi-media, multi-authored documents) is produced in terms of goal and enabling tasks (see Byerley et al. (1990) for a discussion of this distinction). Briefly, a goal task is one that has the user as its agent and the user's goal state as its outcome. An enabling task is one that creates the necessary conditions for execution of a goal task, termed an enabling state. A tenet of the Guidance design philosophy is that, where possible, enabling tasks should be performed automatically by the system, leaving the user free to concentrate upon goal tasks.

This task description is used in the production of the enabling state parameters, which are essentially a set of known characteristics of the tasks, users, devices and environments. These are then used to identify operational requirements of the system. An interface specification stage follows which consists of a top-down design employing four levels, termed here the conceptual, functional, interface, and device levels. The resulting specification is then implemented and tested in prototype systems.

1.3 Relationship between principles and design process.

The stages in the design process (Fig. 1) marked by asterisks are those at which principles are thought to be applicable. The Guidance approach is to produce both generic and specific principles. It is intended that generic principles will be applicable at the operational requirements stage whilst specific principles will be applicable at each of the levels of the specification stage.

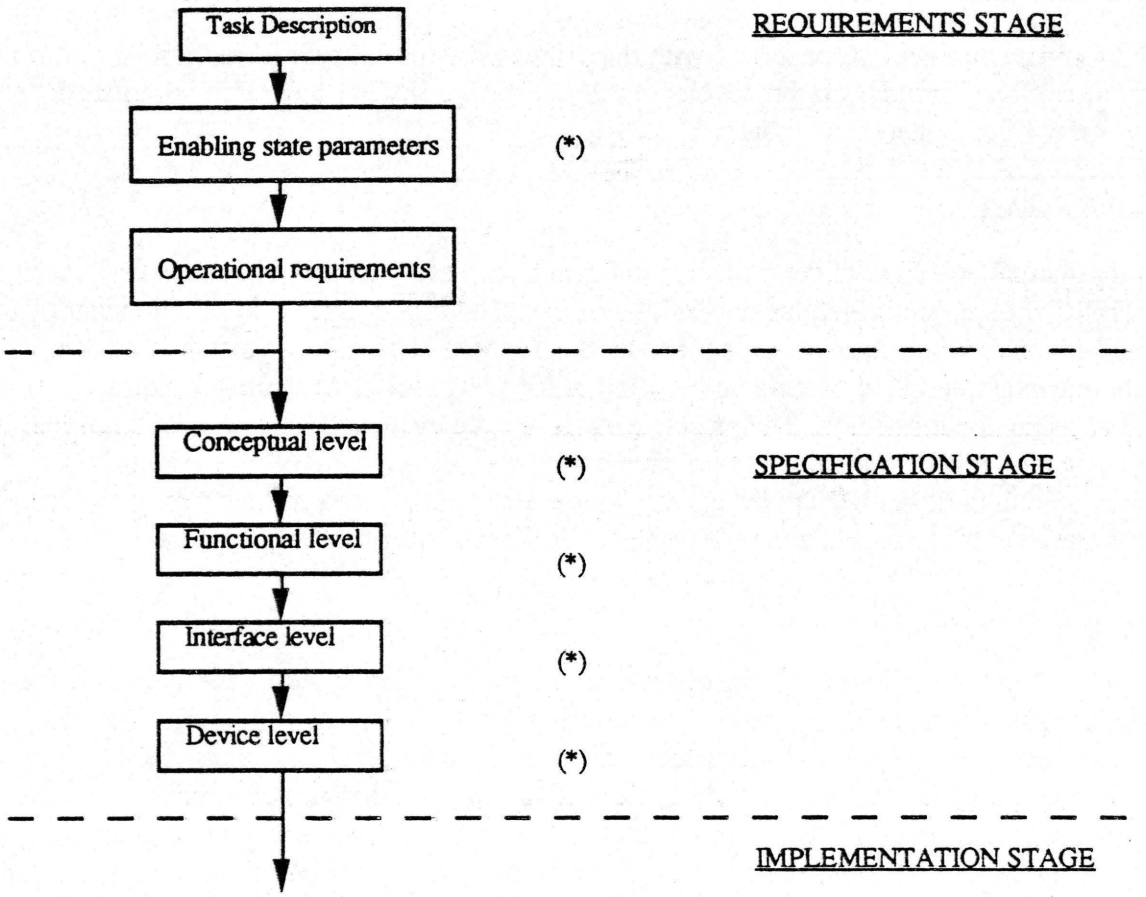


Fig. 1 Basic Design Process (simplified)

2 Non-Guidance principles

Before considering principles as they are understood within Guidance, it is useful to examine how others have used the term. Long and Dowell (1989) have identified three different approaches to human computer interaction (HCI): the *craft, applied science,* and *engineering* approaches. Each of these approaches produces a form of principle and these will now be considered in turn.

Within the *craft* approach to HCI, practices are supported by knowledge that is typically in the form of heuristics or rules of thumb. Such knowledge is acquired by practice and example, and is neither explicit nor formal. Within the *applied science* approach, however, HCI uses theories and models to support hypothesis and test. In contrast to heuristics, scientific knowledge is explicit and formal, operational, testable and generalisable. It is usually expressed in the form of guidelines for designers.

Long and Dowell (1989) claim that current human factors (HF) practices within HCI typically have the character of a craft, where success depends primarily upon the expertise, guided intuition and accumulated experience of the practitioner. An example of successful application of such an approach may be found in Bornat & Thimbleby (1989) who describe the development of the text editor 'ded'. Numerous heuristic principles were applied and tested and those found to be beneficial included simplicity of operation, consistency, and modelessness.

Long and Dowell (1989) assert that the craft form of HCI is inadequate. In particular, human factors activities within a craft approach do not transfer easily between applications, have unknown scope, carry no guarantees as to their effectiveness, and often require reinvention so that no systematic, cumulative progress is made.

The applied science approach, on the other hand, may be exemplified by the work of Gardner & Christie (1987) or Smith & Mosier (1986). As one of over two hundred guidelines offerred by Gardner & Christie (1987), the following may be taken as an example: "In the design of tasks, task elements may be combined in the following ways - *simple + simple*, *complex + simple*, but not *complex + complex*. "

With an applied science conceptualisation, Long & Dowell (1989) cite a further set of objections. Echoing Smith & Mosier's (1986) finding that designers typically find guidelines difficult to locate, anachronistic, and ambiguous in their application, Long & Dowell note that guidelines are poorly integrated into system development, that they do not

adequately describe behaviour, and that they carry no guarantee of effectiveness. Moreover, science is in any case explanatory and predictive rather than prescriptive. The successful application of science-based guidelines is often in fact craft-based, being dependent upon experience.

An *enginering* conception of HCI is favoured by Long & Dowell (1989). Although currently only a hypothetical aproach to interface design, Long & Dowell contend that human-computer interaction engineering (HCIE) as a codified body of engineering principles is the way forward for the discipline. Engineering principles are intended to relate prescriptions to consequences with a known probability or guarantee where this specifically relates to the consequences of following the prescription in terms of the resulting performance of the system. Additionally, activating conditions for the application of a principle are systematic and explicit. The framework presented by HCIE has been elaborated and recruited for the specification of principles in Guidance (Life & Long, 1990). These principles cannot claim to be true engineering principles as they do not derive from an engineering theory of interaction between people and machines linked by an IBCN. However, the framework offers an explicit conceptualisation of the elements of interactive IBCN work systems (IIWSs) and their performance. It supports the development of craft and applied science based principles with standardised and explicit scope and content.

3 Guidance principles

Within the Guidance project, we have tried to develop principles oriented towards an engineering rather than a craft or an applied science approach. Although not truly engineering principles in Long & Dowell's (1989) sense, they are intended to provide design support as prescriptive information for designers of IBC systems so as to enable usable service integration.

3.1 Strategy for principle development within Guidance

There were two parts to the strategy for principle development. Firstly, a top-down approach sought to determine the characteristics of engineering principles as these would apply to the IIWS. The resulting definition explicitly stated the scope of a principle as a set of *triggering conditions* expressed in terms of a *tasks, users, machines,* and *performance*. A *prescription* is predicated upon these triggering conditions. This pairing has associated with it a *rationale*, explaining why the principle applies, and a *guarantee*, indicating the confidence held in it. This definition may be expressed symbolically as follows:

IF X=f(T, U, M, P)

THEN Y

BECAUSE R(Y=g(X)), with

GUARANTEE G>threshold

where T=task, U=user, M=machine, P=performance, X=triggering conditions, Y=prescription, R=rationale, G=guarantee.

The second approach to principle development was a bottom-up one. This involved a wide literature search in order to identify a broad set of recommendations made for designers, whether heuristics, guidelines or possible engineering principles. The resulting corpus of recommendations comprised some 300 candidates (Esgate, 1989). These were expressed at varying levels of specificity and addressed both substantive and methodological issues. Examination of the set revealed that few if any principles have been or can be expressed in the required form for application to IIWSs since some elements of the definition are usually lacking.

3.2 Selection of candidate principles

Intermediate forms of expression of engineering principles can be produced by successively weakening the requirements on the principle in terms of explicitly stated information. If the full definition is termed the 'strong' form, then a 'weak' form may be defined as one in which none of the information is required. 'Intermediate' definitions requre one or more of the slots in the definition to be filled.

Application of the intermediate definition as a selection device reduced the candidate set to 27 principles having at least one of the required pieces of information present. In the case of craft based principles this was usually a guarantee deriving from actual use. In the case of applied science based principles, a rationale was usually implicit within the principle but in fact it was rare for this to be explicitly stated. Both types of principle were amenable to expression as an IF-THEN construction with explicitly stated triggering conditions, though this form of expression was rarely used.

The second criterion for selection from the candidate pool was applicability of the principle to the problems of integration. Integration within Guidance is conceived in two ways, as integration of services within a single machine (intra-workstation integration) and integration of a distributed workgroup (inter-workstation integration). The principle must, then, either arise directly from, or take on new significance when applied to, either or both forms of integration. As examples from the candidate pool of 27 principles, 'modelessness' appears to be applicable to intra-workstation issues and 'what-you-see-is-what-I-see (WYSIWIS)' to inter-workstation issues.

Example - The principle of compatibility

To illustrate how a principle may pass the test for inclusion, consider the case of the principle of compatibility. This principle, which is assumed to mean cognitive rather than stimulus-response compatibility, is generally seen as a relation between the user and machine, expressed as compatibility between the knowledge possessed by the user and that required to perform a task using the machine (Morton et al., 1979). Such knowledge, which the user may be assumed to possess, includes knowledge of natural language, of extant tasks, and of other computer systems. A command syntax that reverses the natural language order is clearly incompatible with that knowledge and therefore has low cognitive compatibility. This principle gains admission to a candidate pool under the first criterion since there is much evidence for its utility (*guarantee)* both from actual application (Buckley & Long, 1985) and applied science (Barnard et al., 1981), the latter also providing much of the *rationale* for the principle.

As to its applicability to integration, cognitive compatibility would be maximised in a multi-service environment, by making available the services with which the user is most familiar (word-processing, video control, camera control etc.) with few, if any, modifications. It is clear, then, that cognitive compatibility is an important candidate principle for evaluation in the design of integrated systems.

This selection procedure left 11 principles as candidates for elaboration and testing within Guidance. They were as follows (only a terse expression of each is given here): automatic tasks should have only one method of use; avoidance of memory overload; avoid branched or nested loops in task elements; compatibility; complete one task at a time; consistency; indicate lack of consistency by different visual environments and controls; modelessness/modularity; reversibility; simplicity; what you see is what I see.

A final selection criterion relates to evaluation. Those principles chosen for evaluation must be capable of test within the prototype IBC system available to the project. Principles such as adaptation of the system to its user, which are difficult to realise technically, cannot therefore be at the initial focus of evaluation. Additionally, the set of candidates for evaluation is further reduced since the early evaluation work concentrates upon principles adddressing intra rather than inter workstation issues.

3.3 Relation between generic and specific principles

The candidate set of 11 principles are expressed at the most general level. In terms of the definition, the slots are filled with a default value of 'all'. Although this claim is clearly extravagant, the principles nonetheless can be hypothesised to hold over a wide range of situations. In order to be of use to Guidance, however, their generic form must be made tighter and their specific forms must be made available.

A generic principle will express a prescription applicable to a range of IIWS configurations and tasks. Its preconditions will be such as to describe the subset of task, machine and user attribute values which must exist for the prescription to hold, and the prescription will describe a set of interventions which will have the desired effect on performance. Specific principles constitute instantiations of a generic principle with respect to a particular configuration of task, machine and user.

The Guidance project is using, as a vehicle for its research, systems to support MMMADP. Consequently, the specific principles developed by the project will be restricted to multi-media document production tasks (implicitly including the reading of such documents), users who are authors/readers (multiple and distributed) and machines running integrated services to support document production/reading.

3.4 Relations between generic principles

Conflicts may arise between generic principles. For example, the meanings given to similar interaction objects in different services will depend upon whether they are defined according to the principles of compatibility or of consistency. The latter principle would dictate that specifcation of objects must be the same in all services whilst the former would advocate retention of the characteristic features of services, possibly as system modes or modules.

The consistency principle, in fact, gives rise to a range of conflicts since there are many ways in which the principle may be usefully contravened, for example as modularity or as adaptation of the system to a user. Initial evaluation work in Guidance has actually shown that modularity may well be preferable to strict consistency in an integrated system (Guidance internal report).

Such conflicts illustrate a considerable requirement for empirical investigation in order to determine the exact circumstances (or triggering conditions) under which a particular principle is applicable. A set of meta-principles, defined over a space of principles, may be required for conflict- resolution between specific principles.

3.5 Example principles

In this section examples of generic and specific principles will be presented as representatives of a set to be evaluated within Guidance.

Example generic principle 1

IF the *task* is an occasional search task AND the *users* are untrained adults with experience of the task using existing technology AND the *machine* offers a multi-media database service AND the *performance* must be better than is achieved using existing technology

THEN (*prescription*) the services should enable the user to record and take away information in a familiar manner

BECAUSE (*rationale*) users' knowledge of the existing task may be recruited to the task

WITH a *guarantee* of highly probable.

From this generic principle, which is an expression of the general compatibility principle applied to an IIWS, a specific principle may be constructed by filling the slots in the definition with lower level information.

Example specfic principle 1

IF the *task* is to find a suitable family summer holiday package from a large database AND the *users* are untrained adults with experience of finding a holiday using conventional brochures and travel agents AND the *machine* is a multi-media holiday database with output as text, images and hard copy AND the *performance* must be a suitable holiday found with minimal interaction errors and few affective costs

THEN (*prescription*) make provision for an online notepad for text and images, with a hard copy facility

BECAUSE (*rationale*) users' search and notetaking behaviour in the existing task may be recruited

WITH a *guarantee* of highly probable.

A further example of a generic IIWS principle is now given where this is based upon the general consistency principle.

Example generic principle 2

IF the *task* requires manipulating information in several media AND the *users* are familiar with manipulating information in some of the media AND the *machine* supports integrated multi-media services AND the *performance* must be at least equivalent to that of systems supporting single services

THEN (*prescription*) common functional and logical relationships should be maintained in the dialogue language

BECAUSE (*rationale*) users will be able to use the same knowledge for interacting with each of the component services

WITH a *guarantee* of probable.

This may again be instantiated as a specific principle.

Example specific principle 2

IF the *task* requires copying of information about holidays presented in text and images AND the *users* are adults familiar with using travel agency brochures AND the *machine* supports text and image user services AND the *performance* must be accurate editing in minimal time

THEN *(prescription)* ensure that the interactive dialogue to copy images is the same as that to copy text

BECAUSE *(rationale)* maintenance of common logical and functional relationships in the dialogue will enable users to use the same knowledge for interacting with each of the component services

WITH a *guarantee* of probable.

Using Fig. 1 the points of application of these two examples of generic and two examples of specific principles may be indicated. The generic principles 1 and 2 find their application at the requirements stage, especially in the production of operational requirements, whilst specific principles are employed in the layered interface specification stage. Specific principles 1 and 2 find their applications in the specification of the conceptual layer and of the interface layer respectively.

4 Future work

The project is in its early stages at the time of writing and the above ideas on principles need to be developed and tested. The main strands of future work, therefore, are evaluation and further refinement of the principles, development of the delivery vehicle for principles, and development of new principles.

4.1 Evaluation of principles

Because IBCN is a novel and developing technology, it is premature to seek to develop principles claiming to have been 'validated' in the scientific sense. At least initially, empirical work will be exploratory, aiming to elaborate, refine, or reject candidate principles, rather than determining their 'truth'. In terms of the Guidance framework, experiments will enable tighter specification of the pre-conditions and rationale for particular prescriptions and of their impact on IIWS behaviour. To specify quantitative impact on IIWS performance would seem to be ambitious, and principle guarantees could only be expressed approximately and tentatively. Evaluations will tend, therefore, to be diagnostic rather than criterial (Whitefield et al., 1990) concentrating on the identification and interpretation of interaction errors and user difficulties.

4.2 Delivery vehicle for principles

It has been decided to design and implement a small system as a feasibility study. This is termed the Concept Prover (CP). The CP can be thought of as comprising three major components: the user interface (UI) prototyping tool, the knowledge based system (KBS), and the human factors (HF) principles in hypermedia form. Users will be able to create simulated user interfaces using the prototyping tool.

On the assumption that the CP is representative of the delivery vehicle for the principles, the principles produced by Guidance should be expressed in a format compatible with representation in a KBS. Most currently available KBS shells assume a rule-based knowledge representation so that Guidance pinciples will be portable to such a system with little further modification.

4.3 Further development of principles

Further work on the Guidance principles will seek to produce specific principles for the MMMADP task from the current generic principle set, refine those specific principes by empirical evaluation using the prototype, and produce new sets of generic and specific principles.

Acknowledgements

The Guidance (Generic Usability Information for the Design of Advanced Network Communications in Europe) project is funded by the European Community's RACE (Research in Advanced Communication Technologies in Europe) initiative (project 1067). The lead partner is Standard Elektrik Lorenz (Germany) and four other partners are involved: British Telecom (UK), Plessey (UK), Swedish Telecom (Sweden), and University College London (UK).

The authors would like to thank the directors of RACE for the support the project has received. They would also like to thank project partners for their contribution to the research. The views expressed here are, however, those of the authors.

References

Barnard P.J., Hammond N.V., Morton J., and Long J.B. (1981) Consistency and compatibility in human-computer dialogues. Int. J. Man-Machine Studies, 15, 87-134.

Bornat R. and Thimbleby H., (1989). The Life and Times of ded, Text Display Editor. In J.B. Long and A.D. Whitefield (ed.s), **Cognitive Ergonomics and Human Computer Interaction**. Cambridge: Cambridge University Press.

Buckley P.K. and Long J.B. (1985): Effects of system and knowledge variables on a task component of "teleshopping". In: P. Johnson and S. Cook (eds): **People And Computers: Designing The Interface.** Proceedings of HCI'85. Cambridge: Cambridge University Press.

Byerley P., May J., Brooks P., Keil K., Whitefield A., and Denley I. (1990) Enabling states: a new approach to usability. Proceedings of the 13'th International Symposium on Human Factors in Telecommunications. Turin, September, 1990.

Esgate, A. (1989). RACE 1067 Guidance project deliverable 8.

Gardner M. and Christie B. (1987). **Applying cognitive psychology to user-interface design**. Chichester, Wiley.

Life M.A. and Long J.B. (in preparation) Machine supported co-operative work systems: a conception of usability and an illustration of its use.

Long, J. and Dowell, J. (1989). Conceptions of the discipline of HCI: craft, applied science and engineering. In A. Sutcliffe and L. Macaulay (eds.) **People and Computers V.** Cambridge: C.U.P., 9-31.

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.

Smith and Mosier (1986). **Guidelines for designing user interface software,** Bedford, Mass., Mitre.

Whitefield A., Wilson F., and Dowell J. (in press) A framework for human factors evaluation. Behaviour & Information Technology.

OPERATIVE COMMUNICATION AND DECISION MAKING IN EMERGENCY MANAGEMENT

Janine Rogalski

Directeur de Recherche, CNRS-Université Paris VIII

URA1297, "Psychologie Cognitive du traitement de l'information symbolique"

Equipe de Psychologie Cognitive Ergonomique

2, rue de la Liberté, F-93526 Saint-Denis Cedex2.

Abstract

This paper presents an empirical study[1] set within the framework of "distributed decision making" in dynamic environment control. It describes hierarchical operative communication centered around the decision-maker whichis modelled as an operational device with three supervisory levels. Methodology is presented for analysing collective verbal protocols in real time. Comparison of two actual situations (forest firest in the South-East of France) show that structural features associated with the primary function of each level of communication remain constant, thus lending eight to thetheoretical description. Analysis of verbal exchanges with the decision-maker (Emergency Director) indicates that speaker distribution as well as nature of referent/topic were related to the phases of the dynamic environment and at the same time to the emergency director's decision strategies, reflecting"parallel processing" in interaction with actors on each of the supervisory levels.

INTRODUCTION

In recent years there has been a sharp rise in studies on cognitive activitiesin collective work situations in the field of Cognitive Ergonomics. Dynamic environment management -- including most complex tasks such as control processes-- as well as design situations (architecture, programming "in the large", ..) call for effective human cognitive interaction. Operators must cope with two interrelated tasks: problem solving of the process to be controlled (with or without support systems) and communication with other operators on various supervisory levels.

The Brehmer (1988) analysis of organizational requirements for decision making in complex systems points clearly to the role played by the hierarchical structure of decision making in situations such as open dynamic environment control. Forest fire fighting in the South East of France (with critical parameters such as aridity, wind and arson) lends itself particularly well as a paradigm for the study of what is termed distributed decision making that covers the entire processes of diagnosis, prognosis, planning, evaluation, mission assignment and execution control. Distributed decision making in forest fire fighting can be studied from the standpoint of the task, or from that of activity. For purposes of long-term development of training programs and operational support systems (including computer support) enabling fire officers to meet the challenge of handling major disasters, focus on task requirements at the highest supervisory level was selected here. More specifically, the main goals of the study were to analyze distributed decision making from the point of view of the emergency director (ED), who is in charge of operations each time a specific situation of this type arises.

---

[1]This research was supported by a grant PIRTTEM-CNRS and Ecole Nationale Supérieure des Officiers Sapeurs-Pompiers (Civil Safety Direction); it was developped in collaboration with Renan Samurçay (CNRS).

This particular focus creates a number of methodological problems as regards data collection and analysis. Inferring cognitive processes from verbal data has shown itself to be productive, and a number of methods have been developed to this end (see Bainbridge, 1985). These methods however deal primarily with the analysis of individual verbal protocols and transfer to collective situations calls for specific empirical work. Methods for analyzing expert/expert an expert/consultant dialogues in operative languages (Falzon, 1989) which are often oriented towards implementation of "explanations" in expert systems and ITS design can be used in cases where there is collaboration between two operators working on the same task.

The methodology adopted here to analyze collective decision making in forest fire fighting is based on a theoretical model (Rogalski, in press). It combines a collective task analysis on the "vertical" dimension which models task distribution for a hierarchical structure on several supervisory levels (Rogalski, 1989), and a "horizontal" dimension representing task distribution on each supervisory level, with a team (Command Post) as a support system assisting the ED (Samurçay and Rogalski, 1988).

A "cross analysis" method was developed and applied to verbal collective protocols to analyze exchanges (who is speaking to whom) and semantic content (WHAT -entities and topics) in order to relate cognitive strategies in decision making and task allocation, with cooperative work organization and control at a given supervisory level (Samurçay and Rogalski, 1988). Cross analysis can thus be seen as an extension of methodology developed recently for analysis of collective team work at a given supervisory level (Van Daele, 1988; Moray et al., 1989; Reinartz and Reinartz, 1989; Curtis and Walz, in press).

THEORETICAL ISSUES AND METHODOLOGY

Task Analysis

Cases of emergency management such as Mediterrananean forest fires place heavy cognitive demands on operators: neither the means for information gathering or fire fighting equipment are immediately available since place and time of disaster cannot be predicted beforehand and fires tend to spread. Diagnosis is a minor part of cognitive activity; the operators' task mainly involves hypothetical reasoning for prognosis on evolution, planning of appropriate actions in relationship to hypotheses on evolutions, and for anticipation of effects of these actions on subsequent ones. In Hoc's cognitive task terminology (1989) fire fighting can be described as a task with a limited number of parameters but high requirements in terms of process information, hence involving a considerable amount of qualitative reasoning and extrapolation. This necessitates a hierarchical model of operator knowledge representation.

The Emergency Director's final goal in decision making is to define a global device (composed of manoeuvres: a series of actions, carried out with appropriate means and the right time and place) and give the orders to implement it, to reach a stable target state, while limiting the negative consequences of an irreversible event on human lives, and to the physical and ecological environments. Decision making of this type can be formalized as a loop composed of information gathering, prognosis and planning, decision making and giving orders, execution and control. The entire process may involve as many as a hundred actors, grouped into several sub-systems having a number of functional roles

These sub-systems can be represented in an operational device with a hierarchical structure. The "operational center" (OC, level N-1) plays a strategic role in ways and means analysis, and operationalizes the tactical decisions made by the ED (at level N). The Command Post (CP, at level N) acts as a decision support system to the ED; "sector chiefs" (SC, level N+1) are in charge of top-down propagation of missions and bottom-up execution control. "Intervention groups" (IG) execute orders at the physical level (level n+i). Figure 1 shows how the operational device works in the case of a real large Mediterranean forest fire. In terms of decision making, these levels are parallel to those defined by Rasmussen for process control (Rasmussen, 1986).

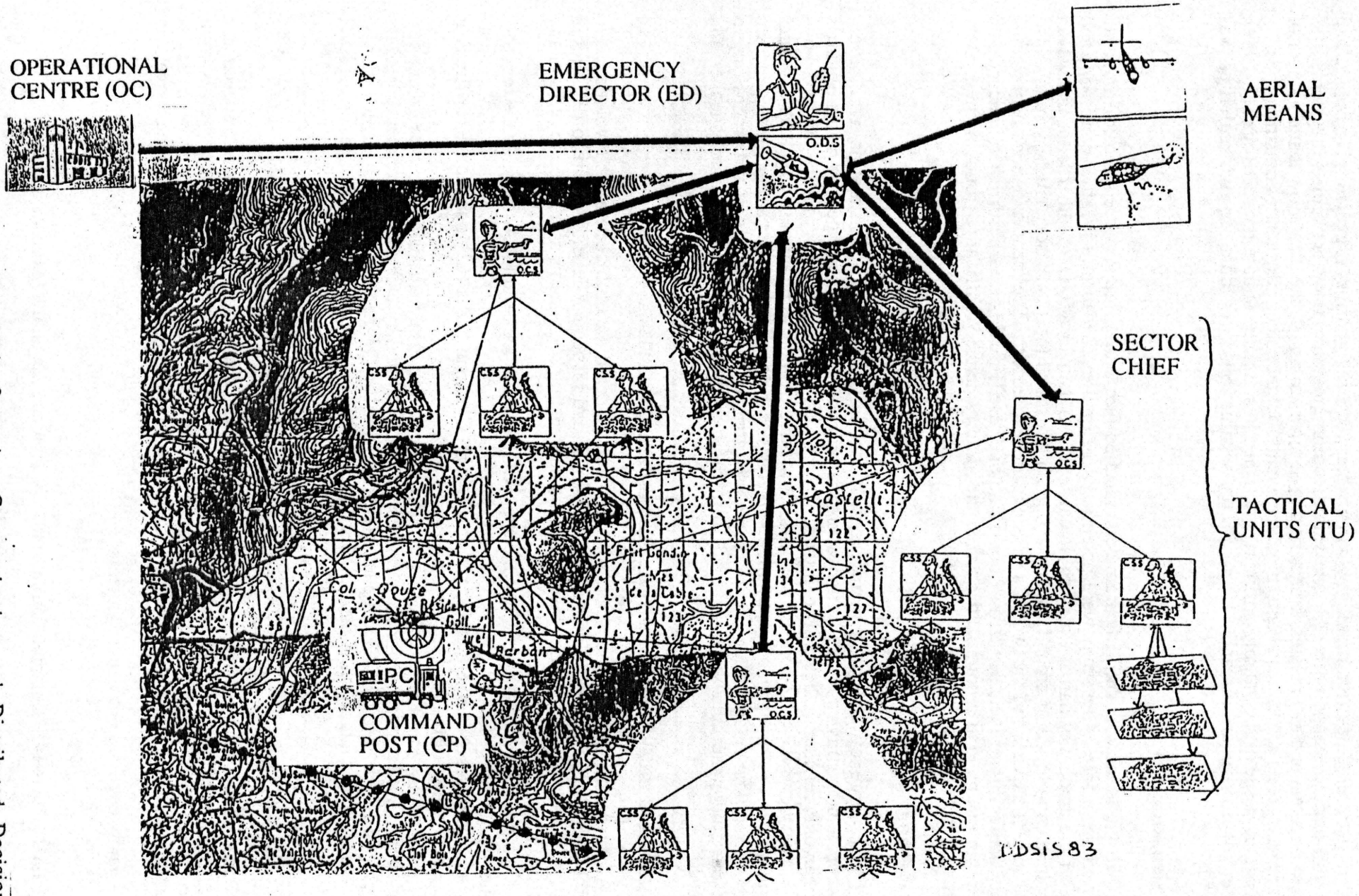


Fig.1 Schema of implementation of the operational device for a mediterranean forest fire. The boldened lines represent hierarchical communication between the three supervisory levels involving the ED. The doted lines on the map represent the theoretical limits of fire (for the given fire parameters). The 3 sectors (fire front and fire sides) ar hierarchically organized in the same way, the number of levels depending on the fire size and number of intervention groups. Tactical units denote the executive levels considered as a whole.

Study of the communication flow between sub-systems provides a mean of analyzing the process of distributed decision making which enables the ED to define, organize, give orders, and control appropriate manoeuvres. An analysis of communication flow between subsystems was conducted on a real and a simulated fire (during in-service training). The data show that a number of features remain stable across actual and simulated events, but that other effects are simulation-specific, thus raising the issue of the validity of findings obtained in simulated situations (see Rogalski, 1989). To better identify invariants and categorization of situations flow data for two real fires was compared.

Situation.

Two situations with defined parameters (labelled Fire G and Fire R) were selected from a set of mediterranean forest fires: they were potentially major fires (because of weather conditions and because of the amount of damage they could cause due to their location), no other large fire was in direct competetion for air or ground firefighting, in both cases all the supervisory levels were involved as of the start of the fire, and the two EDs were experienced experts. The main difference between the fires was availability of means in terms of delay: in one case (Fire R) an earlier fire had concentrated considerable means (and a command post) in the vicinity hence significantly reducing delays for firefighting means on the ground.

Methodology

The Emergency Director's main decision making activites are information acquisition, anticipation, and operative communication where the prime function of exchange is to mediate actions (rather than only regulating them). Not all these activities are observable; most must be inferred. Post-operation interviews are time-honored methods for making explicit some of these cognitive activities Data collected in real time are indispensable for analysis of the dynamics of decision making to ensure that only current knowledge of the event is being used for information acquisition and processing. In the present study, the data were obtained from radio exchanges, specifying time, speaker and addressee.

Collective verbal protocol analysis: sub-systems in communication flow

All exchanges dealing with operational frequency (heard by all the operators involved) were transcribed for each operation. Speaker and addressee and respective levels were recorded. Initial analysis (presented here) broke down speakers and hearers into subsystems as a function of level and specific functional role:

- strategical level (N-1): operational center (OC), and fire look-out (WA: surveillance for more rapid intervention)
- tactical level (N): emergency director (ED) and Command Post (CP: only active for fire R at the beginning of the fire)
- executive level (N+1) : sector chiefs and intervention groups, considered to be one entity: "tactical units" (TU : appliances for ground intervention), activated by OC, and "aerial means" (AM) : helicopters (HE : activated by OC) and planes (PL : activated at a higher supervisory level)
-before their integration in tactical units, means (ME) were considered to be the lowest level subsystems.

Analysis of decision making (from ED's point of view) through communication analysis

Analysis of exchanges involving ED employed a semantic-pragmatic approach to speech acts operative communication. (As we are analysing communication from the ED's point of view we will use the french word "interlocuteur" in order to denote the subsystem interacting with ED in communication, whatever it could be speaker or hearer). In EM communication intends to change :

-"knowledge state" of "interlocuteurs"

-of S, M, and A for elaborating and updating a common reference system
-of ED"s evaluation
-state of the dynamic situation
-disaster to be fought (danger, possible actions)
-means for fighting it (activation, guidance, release)

Secondly, speech acts were also analysed in terms of **function** : bring (**!**), ask for (**?**), propose (**p**) or receive (**r**) elements and **content** : information (**i**), about present of future state of affairs, evaluation (**e**), needs (**n**) for actions or means , and activation (**a**) of means or actions (including guidance and release). Some messages involved the communication itself (identification of "interlocuteur", definition of radio frequencies, communication efficiency,..) : these were denoted as "**C**".

Method of protocol coding

Several levels of analysis were defined :
-**exchange** : a continuous series of messages involving the same individual "interlocuteur" (without any other "interlocuteur" interrupting the communication);
-**message** : an input from a single speaker
-**unit** : a communicative item dealing with a single entity, function and content.

Figure 2 presents a coding involving the ED (in FireG) for the first 2 minutes. Time was indicated as frequently as possible at the beginnin g of an exchange, "entities", function and content are coded for each elementary unit.

Time	11 11														11 13	
Interlocutor	CC							AM			CC			TU		
entity ED	S	M	S				M	S			M			M		
entity Int.				C	C	M			C	S		C	M			
function	!	!	!			!	!	!		!	!			!		
content	i	b	e			i	b	i		e	b			a		

Fig. 2 Coding of the units involving the Ed for the first 2 minutes : "interlocuteur", "entities" (processed by ED and by his "interlocuteur") "functions" and "contents".

For these two minutes there are : 4 exchanges, 8 messages and 14 units.

RESULTS

1. Global communication flow

Figures 3 presents the communication diagram and the messages distribution (in percent) between the various subsystems, respectively for FireG and FireR.

Arrows indicate addressee ; percent appear on their left. Some addressees could not be identified for FireG (denoted X in the diagram). The results deal with the beginning of fire and last until the fire was under control (no further spread of flaming front).

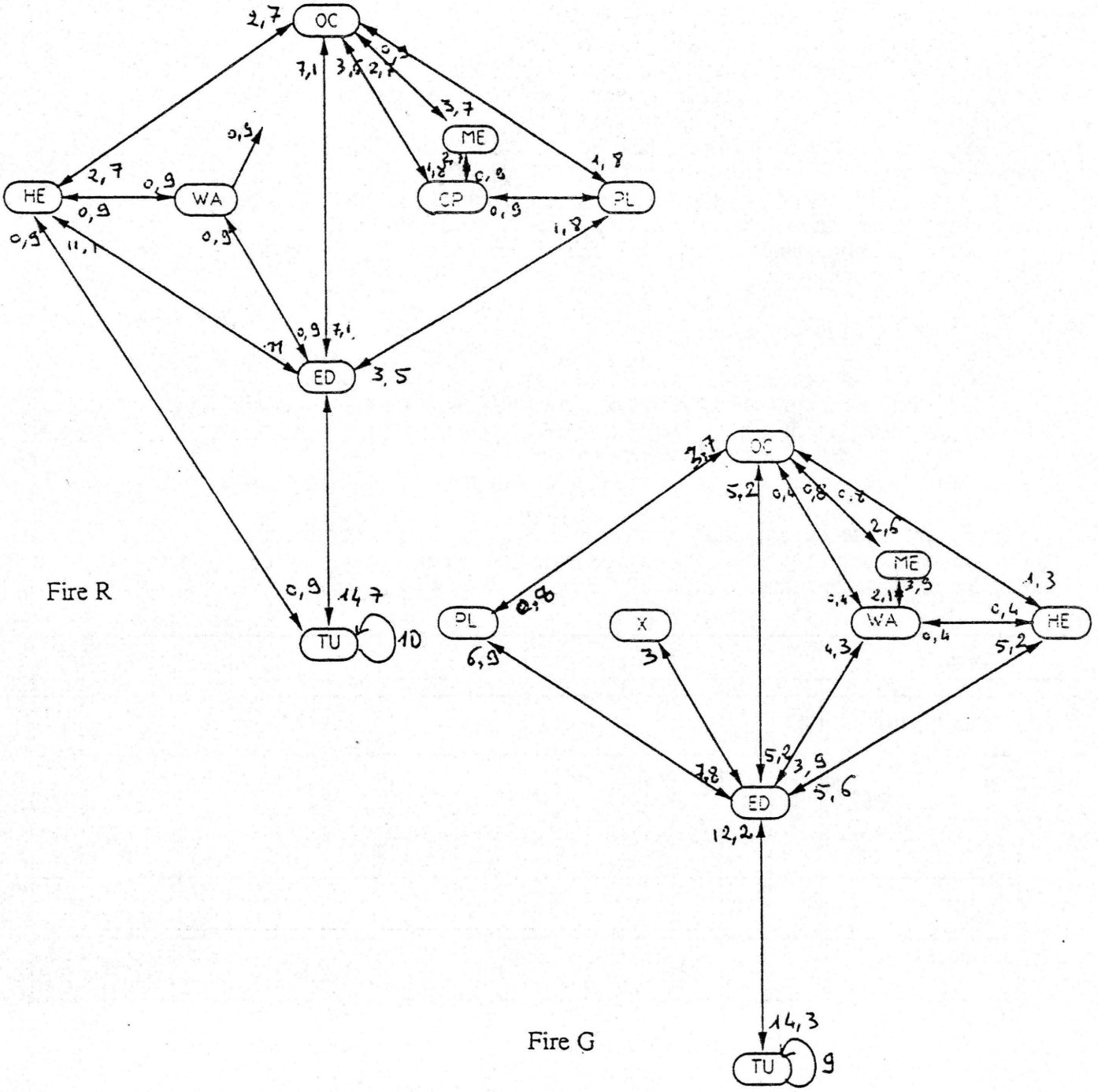


Fig.3 Communication diagram and messages distribution (in percent) for FireR and FireG

Structure of communication flow as well as global distribution of messages involving the subsystems ED, CO, WA, TU AM and ME were similar in the two situations. ED was involved in roughly 70 percent of the messages (68 in FireR and 73 in FireG), whereas TU mainly interacted

with ED (between 22 and 27 percent of the messages), and was partly involved in "internal communication". Local means (ME) were involved in about 5 percent of the messages. Two differences concerned : 1) location in the structure of the already organized CP for FireR and of the lookout (WA) for FireG, in the triangle CO/ME/CP and CO/ME/WA; 2) inside the airborne subsystem distribution of messages concerning respectively HE (present from the very beginnning in FireR) and PL (present earlier in FireG) differed between fires.

2. Temporal distribution of communication units as a function of "interlocuteur" (sub-system) : WHO is involved in communication with the ED?

Table 1 presents the distribution of the number of communicaiton units (involving ED) in time, as a function of "interlocuteur", for the first 30 minutes.

Table 1.a : temporal distribution of communication units in FireR

	Number of communication units for a five minute period						Total
Operational Centre	**08**	**12**	--	--	04	02	26
Tactical Units	01	**11**	**09**	**12**	06	03	42
Aerial Means	--	05	06	02	**09**	**09**	31
Lookout	**06**	--	--	--	--	--	06
Number of units	15	**28**	15	14	**19**	14	105

Table 1.b temporal distribution of communication units in FireG

	Number of communication units for five minutes						Total
Operational Centre	**10**	02	04	**16**	02	05	39
Tactical Units	11	04	**27**	12	11	**29**	94
Aerial Means	05	**31**	12	16	08	03	85
Lookout	**08**	02	03	03	05	06	27
Number of units	34	39	46	49	26	43	245

(Boldface figures indicates points in time when each sub-system elicitated the highest percent of units.)

Operational centre and lookout were active at the very beginning of fire inb the two situations. Except for this point, the two situation contrst in terms of number of comunication units as well as for temporal distribution of focus on the various sub-systems. In FireR there were less units, and they

involved OC, TU and AM in that temporal order. In FireG focus on one sub-system was shorter, beginning with OC (and WA), then AM, TU, returning to OC, ending to TU.

3. Distribution of entities (A, S, M) : with WHAT is communication dealing over time?

Figure 4 below represents temporal evolution of communication units about S, M and A (for the same time units as table 1) for FireR and FireG. The boldened line represents the total number of units ; the other lines represent distribution (in percent) of units dealing with the entities S (fire situation), M (means) and A (actions).

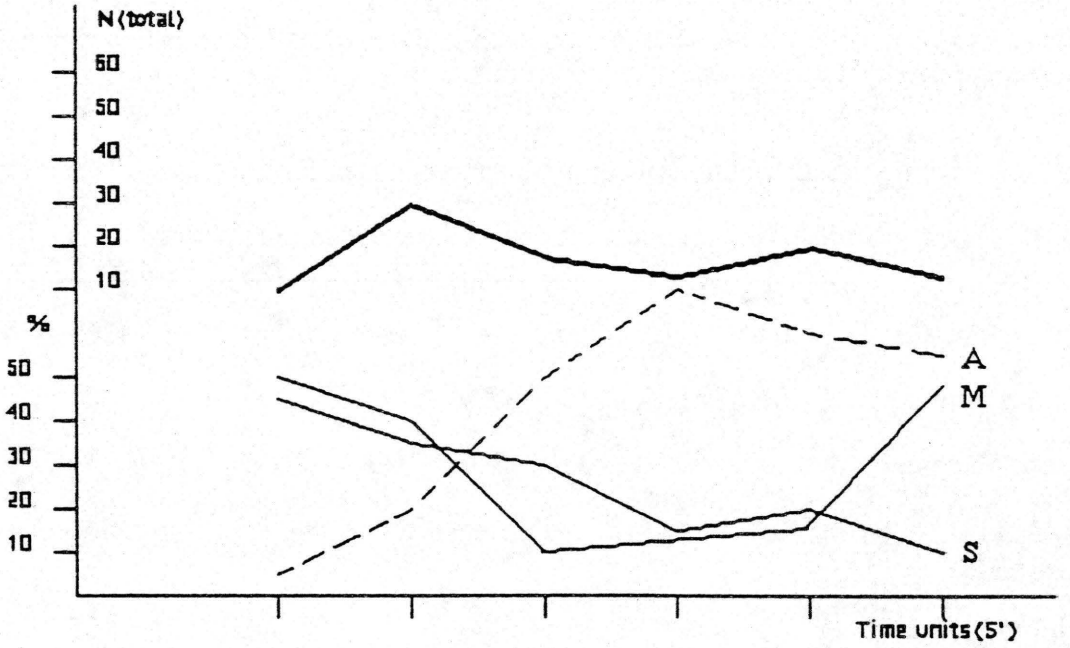


Figure4.a Temporal distribution of units for the entities S, M and A for Fire R

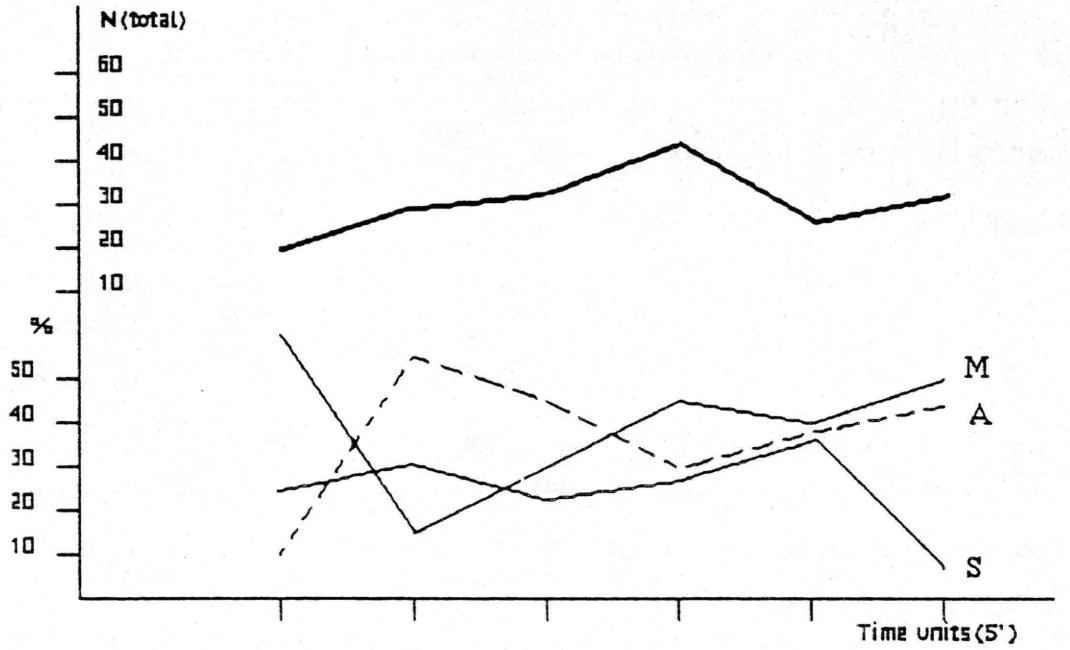


Fig.4.b Temporal distribution of units for the entities S, M and A for FireG

Evolution appears more complex to analyse, however some points can be stressedwhich can be interpreted in terms of relationship between semantical content of communication and fire episodes. At the very beginning communication was centered on the means, while at the end few units were dealing with the fire: this reflects the fact that resources have to be activated and organised in an operational device as soon as possible, while in the last minutes the fire was under control (no more progression of the flaming front), and means could be partly released, while specific actions have to be decided for the following phase (to attain the situation of "mastered fire" : no more flames). In the fourth time unit there was an acceleration in the rate of progression of the fire in the two situations (due to "fire jumps"), increasing the percent of communication units about actions for FireG and the total amount of communication for FireG (most of them being produced by ED).

4. "Crossing" communicative and semantic analyses: WHO is communicating about WHAT with ED?

The previous analyses which were respectively centered on the structure and on the semantical content of communication can be related together, through a deeper analysis of the series of communication units (produced by ED), involving a given "interlocuteur". In communication with **OC** series **giving information or evaluation on fire then expressing needs or evaluation of means** are relatively numerous, while series of **information or evaluation on fire were related to activation of actions** in communication with **AM and TU**. Moreover communication about the means were scarce when "interlocuteur" was Aerial Means, and more frequent with the sub-system of Tactical Units. In fact Tactical Units involved several supervisory levels, and were organized in several spatial sectors, hence functional communication was needed for allocating means to a given sector and eventually for guiding them to the appropriate place for intervention ; Aerial Means were directly operating on fire, with a large azimutal view, requiring neither allocation nor guidance.

DISCUSSION

The existence of these last series reflects some crucial points concerning the functions of the sub-systems defined as ED's "interlocuteur"s. The highest level (OC) was supporting the operation, by allocating means in function of diagnosis (giving information) and prognosis (giving evaluation) about fire (presented by the ED). The lower levels (AM and TU) were mainly engaged in the executive part of decision making; they were activated in explicit relationship with the threats (induced by fire) to be fighted.

Communication was evolving along time both with the phases of the dynamical event and with the functions to be fulfilled, as indicated respectively by Figure 4 (a and b) and by Table 1. Moreover, in communicating with his "interlocuteurs" EDs seemed to be engaged in some **parallel processing** : continuity of the functionality of the communication was maintained through the variation of the effective individual "interlocuteurs". In fact from the point of view of functional sub-systems (represented in Table 1) one can observe the succession of activation of the support function of OC (for means) and of activation of executive functions of TU and AM (for actions).

In FireG activation of support function was explicitly followed by control of execution, while execution of actions could be directly observed by ED so that he was mainly activating new actions (with few explicit evaluation, and few questions about them). In FireR control of actions seemed to be less "parallel", so that Aerial Means were "frequently" asking for missions after the critical phase of the fire (this fact explaining the high percentage of communication units about action during the ten last minutes -see Table Ib).

CONCLUSION

"Crossing" communicative and semantic analyses of verbal exchanges between operators involved in "distributed decision making" appears as a general method, enabling comparative analysis

between situations when there is no preexisting reference solution to relate to(Samurçay & Rogalski, 1988; Rogalski, 1989).

Considering sub-systems as "interlocuteurs" of the decision maker was done from a structural model of "operational device" (fig.1) ; it allows to compare the structure of communication flow between to "similar" fires. The structure of communication flow is revealing in terms of global distribution of messages and role allocation for sub-systems.

Analysis of the temporal structure of exchanges with the ED indicates that parallel processing in communication management and in decision making took place. It also shows a relationship between fire "episodes" and communication distribution on entities and/or between "interlocuteurs".

A more fine grained analysis of messages served to refine hypotheses on relationship between some cognitive activities of the decision maker and communication as a mediator for actions. Fire diagnosis and prognosis intervene both when ED is activating the support systems for means allocation, and when he is activating operative sub-systems for actions execution.

However, in order to precise hypotheses on the components of operational expertise it appears necessary to analyse exchanges at a more detailed level than those used in the present analyses. From a more detailed analysis (on FireG) of the units dealing with information and evaluation of the situation (!iS and !eS) in relationship with units dealing with activation of actions and needs of means (!nM and !aA) it was possible to contrast ED's capacity for long-term anticipation, implying drawing conclusions on "ways and means" defined at a tactical level from the very beginning of fire, as opposed to short term anticipations involving outcomes at the "physical" level, expressed by some actors in the other sub-systems.

Acknowledgement

We want to express our thanks to the Director and the officers of the Direction Departementale des Services d'Incendie et de Secours of the Var (DDSIS83) for their welcome and for their active collaboration.

REFERENCES.

Bainbridge L. (1985) Infering from verbal reports to cognitive processes, The research interview, Academic Press, 201-215.

Brehmer B. (1988) Organization for decision making in complex systems, in L.P.Goldstein, H.B.Andersen & S.E.Olsen (Eds), Tasks, errors and mental models, Taylor and Francis, London, 116-127.

Curtis B., Walz D. (in press) The psychology of programming in the large : team and organizational behaviour, in J.-M.Hoc, T.Green, R.Samurçay & D.Gilmore (Eds), Psychology of programming, Academic Press, London.

Falzon P. (1989) Ergonomie cognitive du dialogue, PUG, Grenoble.

Hoc J.-M. (1989) Cognitive approaches to process control. In : G.Tiberghien (Ed.) Advances in cognitive science, pp 178-203, Chichester, Wiley and Sons.

Johnson P.E., Zualkernan I., Garber S. (1987) Specification of expertise, International Journal of Man-Machine Studies, 26, 161-181.

Moray, N., Sanderson, P.M., Vicente, K.J. (1989) Cognitive task analysis of a team in a complex work domain : a case study, Second European Meeting on Cognitive Science Approaches to Process Control, CEC-JCR Ispra -University of Siena, Siena, 333-348.

Reinartz, S.J., Reinartz, G. (1989) Verbal communication in collective control of simulated nuclear power plant incidents, Second European Meeting on Cognitive Science Approaches to Process Control, CEC-JCR Ispra -University of Siena, Siena, 195-204.

Rogalski J. (1989) Cooperative work in emergency mangement : analysis of verbal communications in distributed decision making, Second European Meeting on Cognitive Science Approaches to Process Control, CEC-JCR Ispra -University of Siena, Siena, 183-193.

Rogalski J. (in press) Distributed decision making in emergency management : using a method as a framework for analysing cooperative work and as a decision aid. In : J.Rasmussen, B.Brehmer & J.Leplat (Eds) Modeling Distributed Decison Making, Wiley and Sons.

Samurçay R., Rogalski J. (1988) Control of dynamic environment with distributed decision making. Methodology for analysing collaborative work. ECCE4, Cambridge.

Rasmussen J. (1986) Information processing and human-machine interaction, North-Holland, Amsterdam.

van Daele A. (1988) Collective activity in continuous processes : comparative analysis in four work situations, Report, University of Liège.

A METHODOLOGICAL APPROACH TO COMPUTER SUPPORTED COOPERATIVE WORK

Thomas Schäl (University of Technology (RWTH) Aachen, Federal Republic of Germany and RSO, Milan, Italy)

Buni Zeller (RSO, Milan, Italy)

Summary

This paper presents the basic concepts of a methodology to analyze and evaluate cooperative networks within a work-group setting. A conceptual framework is introduced where cooperation emerges as a fundamental phenomenon in human work. The network can be analyzed as a non permanent organizational form by its main commitment, its nodes, their relation and the articulation of work. Synthetic models help to identify the type of cooperation in groups. Coordination, collaboration and co-decision are identified as the main types of cooperation. The network's performance is defined by others than the traditional model of message transfer. The considered categories 'discipline' the communication and the resulting networks. The final result is the evaluation of the space for improvement by organizational and technological means, for example by using hypertexts software for the common information space and communication technologies for the conversation space.

Introduction

Cooperative networks will be an emerging pattern for enterprises in the nineties. Innovative companies are observing with great attention the psychological, organizational and sociological theories and the (computer-based) tools emerging in the field of Computer Supported Cooperative Work (Greif, 1988), or Groupware (Johansen, 1988), in order to find better ways for orientation and strategies for their organizational change. Nevertheless, still missing are:

- a theoretical framework to analyze and understand cooperative networks that are active within the organizations (Schmidt, 1988; Bannon & Schmidt, 1989);

- a method to design cooperative networks, choosing the tools supporting them, defining the skills of the human beings and performance of the network so that the effectiveness can be maintained in time.

The authors propose an approach facing the above two points (De Michelis et alii, 1989; Schäl & Zeller, 1989, 1990). Its guide-lines to design cooperative networks are based on the consonance of the organizational and technological environment.

Basic distinctions for cooperative work

The CSCW-field is still in a phase of defining its scopes and its differences to other research communities (Office Automation, Human-Computer-Interface, Artificial Intelligence, etc.). People are coming from these areas to CSCW with different backgrounds

and they have to develop an interdisciplinary theoretical framework for a new scientific discipline within CSCW. At the moment most of the work in CSCW is concentrated on the development of computer systems which should support cooperative work. The understanding of cooperative work and its implications for the design of computer systems before the development of specific applications is still missing. If we want to understand more deeply the nature of cooperation occurring within a group, it is useful to investigate the forms assumed by the interactions among its members, critical situations in cooperation, the different nature of cooperative networks in relation to the permanent organization, etc. Supporting cooperation by computer means requires at least understanding cooperation and recognizing the different spaces for design.

Three main types of cooperation are identified in order to distinguish different cooperative networks (De Michelis et alii, 1989; De Michelis, 1990; Schäl & Zeller, 1989, 1990). These fundamental types of cooperation help to describe aspects which are substantially different regarding the nature and the scope of the action to be done, the relationship established among persons, and the type of communication and information occurring within the group.

Coordination is the first type of cooperation. Coordination as a cooperative process is present where the actions of individual persons contribute to the action of others and the others' actions contribute to an individual action. The main problem in coordination is the synchronization of persons, actions and the consistency of the individual actions with respect to the whole process.

The second type of cooperation is *collaboration* as the working-together in the execution of a certain action. The success of collaboration depends on the contribution made by individuals to generate a common understanding and a shared knowledge. At the end of the collaborative process the contribution of individuals cannot be identified because the final product is an entity as a result of all contributions of the group.

The third type of cooperation is *co-decision*, where the individuals contribute to a group-decision. The decision can be taken by equals (all participants have the same qualification and position in the decision process) or by persons of different status (each person participates to the decision on the basis of his specific role). Credibility assessment, creating a common understanding, and sharing a common knowledge, are important aspects in co-decision processes which, therefore, in general are similar to collaboration. The co-decision process likely cannot be split into the individuals' contribution.

A methodological approach to cooperative networks

The methodological approach to analyze and evaluate cooperative networks is based on two socio-technical guide-lines: the first one considers the design of a system within the never ending innovative processes in an enterprise. It is composed of planning, design and field test (Butera, 1990). The second guide-line supports choices for socio-technical systems on the basis of two elements: firstly the consonance among the enterprise's model, its organization and the skills of the involved persons, and secondly the appropriatness of characteristics of performance perceived by the system environment to the socio-technical system (Bolognani et alii, 1988; Bolognani, 1988, 1990).

The methodology has the aim to support, firstly, the description of characteristics and critical aspects of cooperative networks. Secondly, to identify key-components sup-

porting cooperative work in the system. Thirdly, to improve cooperation and network performances. The application of the methodology foresees data collection, distinction of several categories in cooperative work and identification of critical aspects in co-operation. The methodology helps to develop a framework to analyze cooperative work and a common language among the persons involved in its application. Therefore it becomes a guide-line for the social process during its application.

Types of cooperation can be identified in order to distinguish different cooperative networks. In addition to the three fundamental types of cooperation (coordination, collaboration and co-decision), two mixed approaches of 'diagnoses' are introduced in order to describe cooperative networks. These occur when collaboration or co-decision are mixed with problems of coordination in time.

The coherence of the network with the model is checked in the diagnosing phase for the four components (commitment, relation, articulation, nodes). The presence of unharmonic indicators represent a possible space for action (re-design the network, training, organizational change or technical support).

The underlying paradigms

The methodology considers several approaches in the field of Computer Supported Cooperative Work. The first approach concerns the workflow analysis which is mainly based on the 'language/action perspective', developed by Winograd and Flores (1986). This approach which is a further development of 'speech act theory' (Austin, 1962; Searle, 1969, 1975a, 1975b), has been incorporated in several research projects and developments, e.g., Contract (Marca, 1989), Chaos (De Cindio et alii, 1988) and the Coordinator (Action Technologies, 1988).

A second approach comes from economics and organizational theory. It regards the 'transaction costs perspective' (Wiliamson, 1975; Ouchi, 1980). This was further developed in CSCW, especially in the research projects 'Information and Object Lens' at the MIT (Malone et alii, 1986a; Malone & Lai, 1988), and by Ciborra who considers the coherence between organization and type of cooperative network as an important factor to reduce transaction costs (Ciborra, 1987; Ciborra & Olson, 1988). Besides the coherence between organizational context and type of cooperation the transaction cost approach helps to develop a model for the relationship among nodes and their transaction in the cooperative network concerning rules for communication and information management (Malone et alii, 1987).

Other concepts used for the methodology are derived from 'coordination mechanics' (Holt, 1986) concerning the 'centres' in a cooperative network with individual or commonly shared resources. It integrates the role people play in the centres. Different roles are furthermore defined by the communicative behaviour of people managing their daily work (Machay, 1988), and choosing communication channels (Reder & Schwab, 1988).

The performance of cooperative networks is defined by 'communication disciplines' (Petri, 1977, 1979). Petri's 'communication disciplines' offer a theoretical framework for pragmatics in human communication, moving the observer from outside to inside the system. This pragmatic approach overcomes the perception that criteria for a good performance are only quick and reliable networks at low cost. The network is instead characterized by the following 'disciplines': synchronization, identification, ad-

dressing, naming, delegation, authorization, reorganization, valuation, copying, cancellation and composition. The communication disciplines are, for example, embodied in the CHAOS research project (De Cindio et alii, 1987).

Analyzing cooperative networks

The final scope is to identify characteristics of cooperative networks in an organization. This can be accomplished by using indicators which evidence the most significant components (commitment, relation, articulation and nodes) of a cooperative network.

A commitment characterizes a task, or a set of related tasks, with the final scope of a future action. This can be a 'realization' (product, service, etc.) or a 'decision' for a new possibility 'to be' (e.g., new organizations, new structures, new business, new responsibilities). The relation component is characterized either as synchronization (e.g. of actions, time, space, etc.) or as sharing (e.g. of information, culture, knowledge, etc.). The relation can have a high or a low variance within the process. The articulation of work depends on the degree of structuredness and on the number of identified parts in the process. Nodes can be virtual (organizational units) or real persons, and they play different roles of archivers, managers and performers in the process (Malone et alii, 1987).

At the beginning of the methodology's application it is necessary to identify the relevant commitments which are important for the organization. This is done by workshops and first interviews concerning the identified commitments. The data collection and workshops serve also to bring organizational aspects of group-work to the attention of the participating persons and to develop a common framework concerning the underlying concepts and culture (action learning programme).

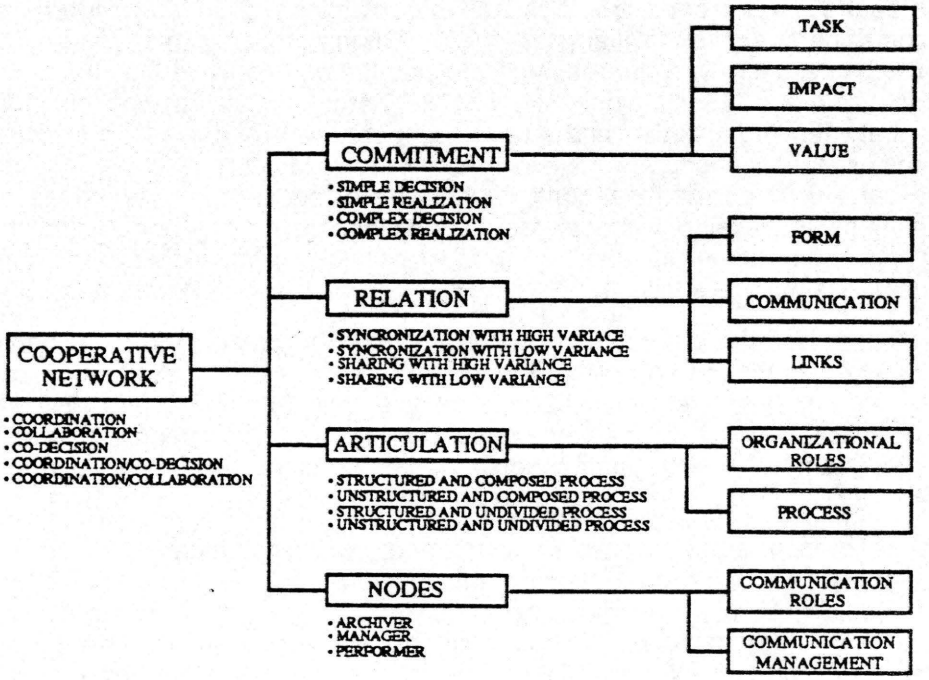


Figure 1: Types and components of cooperative networks

The model to characterize cooperative networks

During the analyzing phase two different check-lists are used for data collection. The first check-list characterizes the cooperative network. General information is collected about the cooperative network within a commitment, e.g. a project, and the time-scale (beginning, phases, deadlines) in a free form. The second part has closed questions with a scale from 1 to 5 (from low to high) concerning the commitment (decision or realization, its impact on the organization and its value for the company), the composition of the final product, the distinction of professional and organizational roles (Butera, 1987), the persons' competence, the relation between persons, the types of conversation and the communication channels. In the third part, the nodes of the network are identified. Material and immaterial resources are listed, which represent used or available centres (Holt, 1986). The check-list is used during interviews with persons who have a general view about the specific cooperative network, e.g. project leaders.

The model to characterize nodes in the cooperative network

The second check-list to characterizes the nodes in the cooperative network has also a structured section (closed questions with a scale from 1 to 5) and an unstructured section. The structured part concerns the activities and the management of commitments, information and new possibilities, the node's audience, the intensity of the relation with other persons, and the characteristics of communication (channel, frequency and type of information). In the unstructured part, the person indicates the resources personally used or available, identifying the role s/he plays concerning the immaterial resources (administrator/author, co-administrator/co-author, user/reader, revisor/ proof-reader), the inputs and the outputs. The check-list is used during an interview with the single person.

Data elaboration

Indicators for cooperative networks and its components are elaborated according to tree-structures which allow to calculate synthetic indexes at different levels. The information concerning the commitment is represented in a figure divided in four areas (simple or complex decisions and realizations). The structure of the commitment is given in a similar diagram which indicates the degree of structure and composition of the 'process'. Another figure provides an index for the form of relation among the nodes which can be of synchronization or sharing (information, competence, culture) with high or low variance.

Data concerning the nodes are represented in several maps. The first map shows the communication roles of those nodes which are part of a network of a single commitment. There are three types: informative, operative and coordinative roles. The second map gives an overview for the management of the nodes' communication concerning the commitment which can be done by archiving, executive or delegating criteria (archiver, performer, manager) (Machay, 1988). A coherence is assumed for role and communication management of a person in a specific network.

The centres are defined on the bases of data concerning the nodes' resources (Holt, 1986). Usually every node has an individual centre for a certain commitment which is individually controlled (administered) concerning material resources (e.g. computer or terminal) and other immaterial resources (e.g. software applications). The resources are physically distributed in individual and shared locations. Further resources

are allocated in spaces shared with other nodes (e.g. manuals, applications in local area networks). Resources are identified in order to create a virtual centre which groups together all the resources which are needed by an individual for the commitment. Nodes have different competences in their centres. They are the administrator for the individual resources, co-administrator, revisor or user for the commonly shared and other's individual resources.

Relevant spaces to design cooperative networks

The common information space

Problems concerning information management are relevant in collaborative processes. This is true in general for the production, revision, filing and distribution of documents. This is also true when collaboration is not directly related with creating or modifying documents but people collaborate on a common task and need to share all the information characterizing this task. In other words, knowledge workers collaborate via a common information space which is a public domain accessible for the collaborators.

A software product, for example, during the different stages of its development, can be described by a series of deliverables which are documents like, product requirements, data analysis, functional specifications, design specifications, development plans, source codes, unit and integration test plans and libraries, certification plans and libraries, user manuals, beta test plans and libraries, etc. Deliverables associated to tasks are information which is related to both, the product and its development. Therefore the information space is defined by the relationship of the information relevant to the product or to the production process. Every change in the product or its production process requires its diffusion in the information space concerning information which is logically related to the change.

The information space is also relevant for co-decision. When several persons participate in a decision process information sharing has a strong role, as in collaboration. But in this case it is possible to distinguish three different types of information: the information regarding the item of the decision, the information regarding the criteria on the basis of which the choice can be made and, finally, the decisions already taken, both if they concern related topics, or if they are steps towards a final decision. Co-decision supports must reduce the effort for people participating in the decision process to share information, opinions and already taken decisions, in order to avoid both, the continuously return to already made steps and the leaving out of persons, as if they are not necessary to take the decision.

A tentative idea of an information space was already given by Busch (Bush, 1945). Meanwhile models for information spaces have been developed (Engelbart et alii, 1973; Engelbart, 1982, 1984; Anderson, 1988). New technologies, like hypertext (Conklin, 1987) and hypermedia systems (Gale, 1988), allow to put this kind of model for knowledge representation into practice.

The conversation space

The productivity of work is strongly influenced by the correct coordination of the interaction among the members of the process and by the cost for quality. Quality cost is highly dependent on the correct exchange of information, e.g. deliverables, among all

people involved in the development of a product. Therefore, coordination supports must reduce the 'transaction costs' taking care of the management of communication of all the nodes in the network. Furthermore, each process has to be seen as a coordinated workflow based on a recurrent organization of work. Some workflows are formally defined. Others are inherited from the experience, without any explicit design. Workflows are organizational and communication patterns based on standard and unexpected actions (break-down) in order to fulfil a particular condition of satisfaction requested by someone. Network members perform three different types of activities associated with particular tasks: Firstly, they negotiate with other persons the characteristics and conditions for the specific task. This leads to the definition of a commitment. Secondly, they produce a report or other products for a given task. Thirdly, every outcome is verified and approved when it is delivered. Following thsi approach in the language/action perspective (Flores, 1981; Flores & Ludlow, 1981; Winograd & Flores, 1986), the development of a product can be represented as a network of commitments between requestors and actors. Everyone who contributes in conversations is part of the network. The relations among the persons are developed by diffent conversation types (conversation for action, for possibility, for clarification, for orientation) (Winograd, 1986) which define the conversation space.

Evaluation of cooperative networks

The performance of cooperative networks is characterized by models developed on the communication disciplines (Petri, 1977, 1979). The observer (user) of the system is inside the network. This overcomes the classical communication theory in cybernetics which is based on the information theory developed by Shannon and Weaver (1957). This mechanistic system reflects an outstanding user's view describing communication by information source, message, transmitter, signal, noise, receiver and destination. Performance is defined by speed, robustness and efficiency. Modern approaches want to realize flexible communication systems which are able to adjust themselves to the organization of the social system where they are implemented. Communication factors allow therefore a network to maintain itself adequately in the presence of complex relations among nodes. The communication disciplines synchronization, identification, addressing, naming, delegation, authorization, reorganization, valuation, copying, cancellation and composition can be seen as a set of network performance evaluation criteria. These criteria represent different and complementary critical factors for cooperation.

Synchronization: This discipline is concerned with getting proper timing restraints and it defines how different speech acts constituting a conversation and any other activities are synchronized among themselves. This could be referred to a measure of time (clock, agenda) or simply to the indication of temporal relations which link activities. In general, synchronization should be based on a partial ordering in terms of causality as opposed to an ordering in time. Criteria for performance evaluation are the synchronization of communication channels, process, time (e.g. completion dates for commitments) and space.

Identification of nodes and resources: This discipline defines the conditions which allow to recognize the source of messages which are exchanged on the network, and to establish the appropiatness of the message's receiver and destination. Criteria for a good network performance are the identification of nodes' competence and type, the identification of resources, its names, and its organizational and physical collocation.

Addressing: This discipline defines routes and system paths which allow to address a message on a cooperative network to an identified destination through a net of channels and agencies. Criteria for a good network performance are up-to-date addresses and routings.

Naming: This discipline is concerned with the modality how names are determined for various entities (persons and resources) on a network. This defines the bases for a common knowledge and to what extend its ambiguity could be avoided or resolved. Important criteria for naming are the significance of names, the standardization of naming, the possibility to maintain names in time and to transfer them into other contexts.

Delegation: This discipline defines how and when a delegation can be made. This concerns both, the delegation of tasks and the delegation of responsibilities from one person to the other. Delegation has also to cope with some effects on synchronization, addressing and other disciplines. Criteria for performance are the delegation's visibility on the cooperative network, its possible predetermination, its cancellation and the management of the delegation's consequences.

Authorization: This discipline assigns and represents access rights to resources, priorities going along with these access rights, the rules and obligations for accessing resources and, if necessary, the rules for the supervision of all these activities. Criteria for performance are the authorization's visibility on the cooperative network, its possible predetermination, its cancellation and the management of the authorization's consequences. Both, delegation and authorization, define the roles performed by persons on the cooperative network.

Riorganization: This discipline defines the rules for up-dating the cooperative network with regard to changes and adoptions of the network caused by the impact of the other communication disciplines. Criteria for a good performance are the re-organization on time, its visibility, its degree of automation and its capacity of a conformable adoption.

Valuation: This discipline has to resolve the problem of limited resources on the cooperative network. All the time algorithms and rules are defined to resolve conflicts among nodes who want to use e.g. a communication channel. Nodes might get special rights on the use of resources or a node might be associated with a relative value on the cooperative network. The performance is examined by economical and organizational (hierarchical) criteria for valuation and rules for priority or casualness.

Copying: This discipline treats aspects of information handling which are related with copying documents. Furthermore it defines the necessary integrity and coherence of copied information. Copying is related to other disciplines, e.g. delegation and authorization which define who has sufficient rights to copy certain information. The up-dating of existing copies and the integrity of contents after a partial or complete copying of information are criteria for a good network performance.

Cancellation: This discipline defines the possibilities to cancel information so that the main contents of documents and unique information cannot get lost. Protection and easiness are performance criteria.

Composition: This discipline defines the possibilities to compose a new document which can derive from existing documents or information. This is highly related to copying, but also to other disciplines like delegation and authorization which define who has the

right to compose what kind of information. Generation, modification and links of information are the performance criteria.

The model for evaluation

Aspects of communication are evaluated from the user's point of view in order to evaluate the network's performance and its characteristics with a third tree-structured check-list. It is used during workshops or in individual interviews. The communication disciplines are grouped together concerning *execution* (synchronization, identification, addressing, naming), *organization* (delegation, authorization, re-organization) and *administration* (valuation, copying, cancellation, composition). The evaluation starts from the users' judgements for the basic level and the relative weights for the importance of every level. Thus the performance of an existing network can be evaluated and the demand for the network's performance expressed. The result is a guide-line for design of cooperative networks (Bolognani et alii, 1988; Bolognani 1988, 1990). The basic assumption of this approach is that information technologies have to be justified by their contribution to improve the organizational performance. Under that perspective the authors suggest to analyze the characteristics of cooperation technologies in terms of their capability to cope with a specific set of communication disciplines.

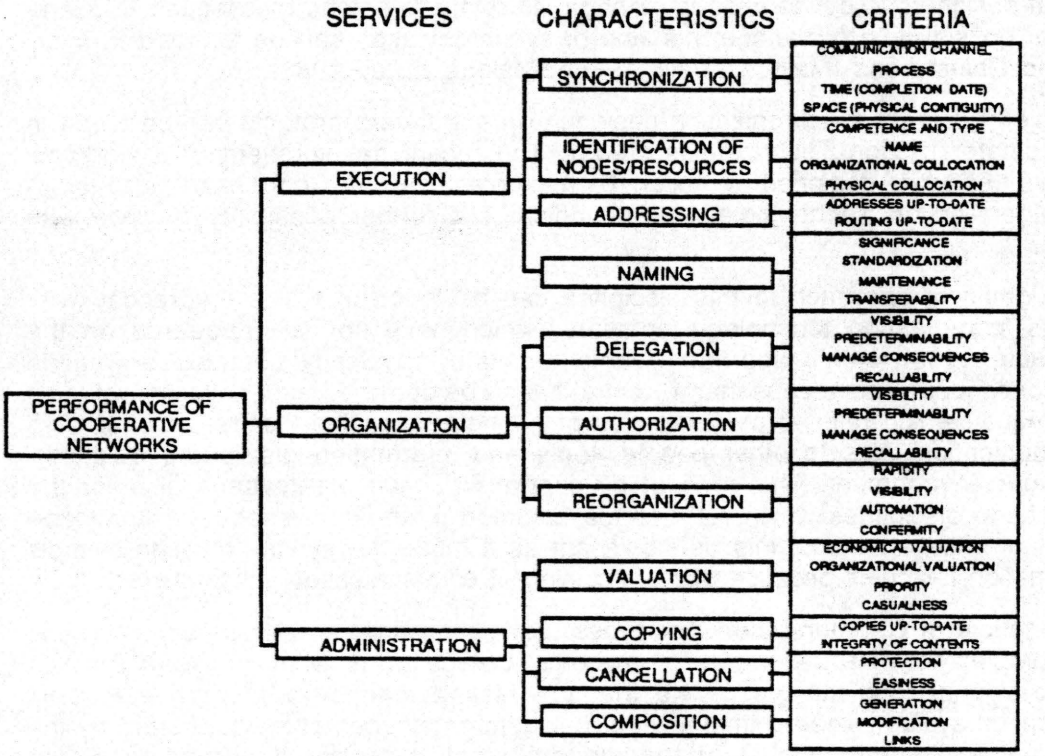


Figure 2: Characteristics and criteria for the performance of cooperative networks

Enabling Technologies for Groupware applications

The enabling technologies considered in this paper for evaluation by the communication disciplines are electronic mail systems and hypertext software. These are main sub-

jects of research and development in the CSCW community and they address the two previously illustrated spaces to design computer support for cooperative work.

Electronic mail

Electronic mail applications can have a relevant impact on the communication disciplines *synchronization*, *identification*, *addressing*, *delegation* and *reorganization*. The first three disciplines have been partially addressed by conventional email applications. Time and space synchronization is supported by asynchronous communication systems which allow people to exchange messages from different time and places. Addressing and identification are partially supported by modern routing systems and this is the subject of X.500 standardization activities by ISO and CCITT (Prinz & Pennelli, 1989).

Nevertheless, we observe that different aspects of the communication disciplines are not supported by conventional email systems. Content synchronization requires the capability of the system to interpretate the semantics of communication or, at least, to manage the relation among different communicative behaviours. Practical examples of real semantic synchronization are still difficult to be found in natural language recognition techniques. More practical possibilities exist by considering a content synchronization based on semi-structured messages (Malone et alii, 1986b) which allows the user or the designers to define rules to manage the relations among messages. An example of user possibilities to manage this kind of synchronization can be founded in Information and Object Lens (Malone et alii, 1986a; Malone & Lai, 1988).

An example of content synchronization designed on a standard protocol can be found in The Coordinator (Action Technologies, 1988). The recent development of a workflow management software designed by Action Technologies over the Coordinator's conversation manager (Action Technologies, 1990) indicates a further possibility to cope with this issue.

Another significant approach to this discipline can be found in Holt's theoretical work and in his 'coordination technology' software, which have not still appeared on the market (Holt, 1986). Holt's approach wants to 'make physically contiguous what is logically contiguous'. This means that a 'centre' has to be designed as a collection of tools and information which refer to the execution of a single task. The 'centre' includes also the communication links to other people, tools and information distributed in a networked work environment. The concept of 'centre' will have a relevance also for the communication disciplines which refer to the common information space (authorization and valuation). Nevertheless, this is significant as a model for synchronization of tools and information channels because of the inclusion of communication links.

The final couple of communication disciplines, delegation and reorganization, introduce a more complex set of aspects in email systems. Delegation is found in systems which allow explicitly or implicitly people to manage task delegation within an electronic communication system. The minimal level of delegation management is supported by the 'forward' utility present in most of the modern email systems. It allows people to 'transfer' their communication to other people in a more or less transparent way. A stronger and more explicit approach to delegation exists in The Coordinator which includes a specific 'delegate-function' to manage it.

The discipline reorganization requires a system which is capable to model and support the changes on the network (its structure, communication rules, etc.) in an efficient

and transparent way. This necessitates an intelligent networking system basically able to redirect any requested connection to the destination of a physical node where the person or information is actually present.

Hypertext software

Hypertext software can have a relevant impact on the communication disciplines *naming*, *copying*, *cancellation*, *composition*, *valuation* and *authorization*.

Naming can be supported by a system which allows to maintain a network of references to the same object without using the same name. The differentiation of names can be preserved as a value in a complex environment by an hypertext model of links which allows different people to reach the same object through a different network of reference links.

Copying, cancellation, and composition are typical problems in a context where people have different rights over a common information space. Conventional data base management systems have been designed within a model which couples the data structure and the browsing system. Within that model any structural change in the information space requires the modification of its structure for access. For example, the introduction of a modification in a document created by a different person and archived on a network server leaves the following two possibilities for the modifier: to modify the access right and modify the original document or to make a copy and modify it. In the first case the integrity of the information is likely lost, in the second case the consistency of the information cannot be kept.

Within the hypertext model the data structure is separated from the browsing system. This allows people to link a new information to the existing one; the two different informations will exist separately but could be accessed in the same domains of browsing. This model can cope with the above mentioned three communication disciplines creating a new way to manage a complex common information space.

The hypertext model has still to be further developed to cope with the disciplines valuation and authorization. In both case the hypertext must include the notion of intelligent links which allow to associate a set of attributes and production rules (Malone & Lai, 1988). Intelligent links must allow people to create different links from the same information node and to trigger production rules to solve the conflict between different paths. This structure would open the possibility to build a more complex authorization model and an appropriate valuation system to manage the resource environment.

An example of this authorization system could be the creation of different browsing paths for different users on the same technical documentation. An example of the valuation system could be the creation of a rule which defines the priority in the access to a specific application analysing the organizational position of two concurrent users or the importance of the tasks which require to access applications or information. A model to cope with valuation and authorization can be also found in Holt's work (1986). In fact the notion of 'centre' includes the functions related to these disciplines.

Acknowledgements

We wish to thank our colleagues in RSO, especially Mario Bolognani, Federico Butera and Giorgio De Michelis, who developed with us the methodological approach to cooperative networks. We gratefully acknowledge the revision of a previous paper made by Dietrich Brandt and other colleagues.

Address for correspondence: RSO, Via Leopardi 1, 20123 Milano, Italy
Tel.: x39-2-72000583; Fax: x39-2-806800

References

Action Technologies Inc. (1988) *The Coordinator Version II: User's Guide*, ATI, Emeryville, California

Action Technologies Inc. (1990) *Essentials of Workflow Management Technology*, ATI, Emeryville, California

Anderson R. (1989) *Video and CSCW*, position paper for the workshop on Computer Supported Cooperative Work in Research Laboratories, Joint Research Centre (JRC) of the EEC, Ispra, 4th-5th December 1989

Austin J.L. (1962) *How to do things with words*, Oxford University Press, London

Bannon L.J., Schmidt K. (1989) *CSCW: Four Characters in Search of a Context*, in Proceedings of the First European Conference on Computer Supported Cooperative Work, Gatwick, London, 13th-15th September 1989, pp. 358-372

Bolognani M. (1988) *Una metodologia provata per la progettazione congiunta di tecnologia informatica, organizzazione e sistema sociale*, in Proceedings of the Conference Joint Design of Technology, Organization and People Growth, Venice, 12-14 October 1988

Bolognani M. (1990) *La Socio-Informatica*, Franco Angeli, Milano

Bolognani M., Granata G., Novara F., Zeller B. (1988) *A Computer Based Assistant for the Contingent Design of Office Systems*, in Proceedings of the conference EURINFO '88, Athens, 16-20 May 1988, pagg. 74-80

Bush V. (1945) *As we may think*, in The Atlantic monthly, 176(1), June 1945, pp 101-108

Butera F. (1987) *Dalle occupazioni industriali alle nuove professioni: tendenze, paradigmi e metodi per l'analisi e la progettazione di aree professionali emergenti*, Franco Angeli, Milano

Butera F. (1990) *Options for the Future of Work*, in Butera F., Di Martino V., Köhler E. (eds.) (1990) *Technological Development and the Improvement of Living and Working Conditions*, Kogan Page, London

Ciborra C.U. (1987) *Reframing the role of Computers in Organizations - the Transaction Costs Approach*, in Office, Technology and People, 3, 1987

Ciborra C.U., Olson M.H. (1988) *Encountering Electronic Work Groups: A Transaction Costs Perspective*, in Proceedings of the Second Conference on Computer-Supported Cooperative Work, Portland, Oregon, September 26-28, 1988

Conklin, J. (1987) *Hypertext: an introduction and survey*, in IEEE Computer, September 1987, pp.17–41

De Cindio F., De Michelis G., Simone C. (1987) *The Communication Disciplines of CHAOS*, in Rosenberg G., Voss K. (1987) *Concurrency and Nets*, Springer, Berlin

De Cindio F., De Michelis G., Simone C. (1988) *Computer-Based Tools in the Language-Action Perspective*, in Speth R. (ed.) (1988) *EUTECO 88: Research in Networks and Distributed Applications*, North Holland, Amsterdam

De Michelis G. (1990) *Burotica*, in Enciclopedia delle Scienze Sociali, Istituto della Enciclopedia Italiana, Roma

De Michelis G., Schael T., Zeller B. (1989) *Le tecnologie della cooperazione*, Working Paper, RSO Futura, Milano

Engelbart D.C. (1982) *Toward High-Performance Knowledge Workers*, in Office Automation Conference'82 Digest, San Francisco, pp 279-290

Engelbart D.C. (1984) *Collaboration Support Provisions in AUGMENT*, in Office Automation Conference'84 Digest, Los Angeles

Engelbart D.C., Watson R.W., Norton J.C. (1973) *The augmented knowledge workshop*, in Proceedings of the AFIPS Conference, volume 42, New York, June 1973

Flores C.F. (1981) *Management and communication in the office of the future*, PhD Thesis University of California at Berkeley (Hermenet, San Francisco 1982)

Flores C.F., Ludlow J. (1981) *Doing and Speaking in the Office*, in Flick G., Sprague R. (eds) (1981) *DSS: Issues and Challenges*, Pergamon Press, London

Gale S. (1989) *Adding Audio and Video to an Office Environment*, in Proceedings of the First European Conference on Computer Supported Cooperative Work, Gatwick, London, 13th-15th September 1989, pp. 121-130

Greif I. (ed.) (1988) *Computer-Supported Cooperative Work: A Book of Readings*, Morgan Kaufmann Publishers, San Mateo, California

Holt A. (1986) *Coordination Technology and Petri Nets*, in LNCS 222 (ed. Rosenberg) (1986) *Advances in Petri Nets*

Johansen R. (1988) *Groupware: Computer Support for Business Teams*, The Free Press, New York

Machay W.E. (1988) *More than just a Communication System: Diversity in the Use of Electronic Mail*, in Proceedings of the Second Conference on Computer-Supported Cooperative Work, Portland, Oregon, September 26-28, 1988

Malone T.W., Grant K.R., Turbak F.A. (1986) *The Information Lens: An intelligent system for information sharing in organisations*, in Proceedings of the CHI '86 (Conference on Human Factors in Computing Systems), Boston, Mass., April 13-17, ACM, New York, pp 1-8

Malone T.W., Grant K.R., Lai K.Y., Rao R., Rosenblitt D. (1986) *Semi-Structured Messages are Surprisingly Useful for Computer-Supported Coordination*, in Proceedings of the First Conference on Computer-Supported Cooperative Work, Austin, Texas, December 3-5

Malone T.W., Benjamin R.I., Yates J. (1987) *Electronic Marcets and Electronic Hierarchies*, in Communications of the ACM, Vol. 30, No. 6

Malone T.W., Lai K.Y. (1988) *Object Lens: A "Spreadsheet" for Cooperative Work*, in Proceedings of the Second Conference on Computer-Supported Cooperative Work, Portland, Oregon, September 26-28

Marca D. (1989) *Requirements, Paradigms and Design Guidance For Developing Coordinator Programs*, in Proceedings 11th International Conference on Software Engineering, IEEE, Pittsburg, Pennsylvannia, May 15-18

Ouchi W.G. (1980) *Markets, Bureaucracies and Clans*, in Administrative Science Quarterly, 25,3

Petri C.A. (1977) *Modelling as a Communication Discipline*, in Beilner H., Gelenbe E. (eds) (1977) *Measuring, modelling and evaluating computer systems*, North Holland, New York

Petri C.A. (1979) *Kommunicationsdisziplinen*, in Petri C.A. (ed.) (1979) *Ansätze zur Organisationstheorie Rechnergestützter Informationssysteme*, Oldenburg

Prinz W., Pennelli P. (1989) *Relevance of the X.500 directory to CSCW applications*, in Proceedings of the First European Conference on Computer Supported Cooperative Work, Gatwick, London, 13th-15th September 1989, pp. 289-302

Reder S., Schwab R.G. (1988) *The Communicative Economy of the Workgroup: Multi-Channel Genres of Communication*, in Proc. of the Second Conference on Computer-Supported Cooperative Work, Portland, Oregon, September 26-28

Schael T., Zeller B. (1989) *A Methodology to Design Cooperative Networks: An Organizational, Technological and Cultural Approach*, position paper for the workshop on Computer Supported Cooperative Work in Research Laboratories, Joint Research Centre (JRC) of the EEC, Ispra, 4th-5th December

Schael T., Zeller B. (1990) *Organizational, technological and methodological issues to design cooperative networks - a case study of an Italian company*, working paper, RSO, Milano, March 1990

Schmidt K. (1988) *Cooperative Work: A Conceptual Framework*, position paper for the workshop on 'New Technology, Distributed Decision Making and Responsibility', Bad Homburg, May 1988

Searle J.R. (1969) *Speech Acts: an Essay in the Philosophy of Language*, Cambridge Un. Press, Cambridge

Searle J.R. (1975a) *A Taxonomy of Illocutionary Acts*, in Gunderson K. (ed.) (1975) *Language, Mind and Knowledge*, University of Minnesota, Minneapolis

Searle J.R. (1975b) *Indirect Speech Acts*, in Cole, Morgan (eds) (1975) *Syntax and Semantics*, Vol. 3: Speech Acts, Acadamic Press, New York

Shannon C.E., Weaver W. (1957) *The Mathematical Theory of Communication*, University of Illinois Press, Urbana

Williamson O.E. (1975) *Market and Hierarchies: Analysis and Antitrust Implications*, The Free Press, New York

Winograd T. (1986) *A Language Perspective on the Design of Cooperative Work*, in Proceedings of the First Conference on Computer Supported Cooperative Work, Austin, Texas, December 3-5, 1986

Winograd T., Flores C.F. (1986) *Understanding computers and cognition - a new foundation for design*, Ablex Publishing Corporation, Norwood, New Jersey

HCI Design and Evaluation Methods

A KNOWLEDGE BASED APPROACH TO SUPPORT THE PRODUCT OFFERING PROCESS

Nerio ALLAMANNO, Pier Paolo PEROTTO

Olivetti Systems and Networks, via Jervis 77, 10015 IVREA, Italy

For a succesful result in offering an Information Technology product to the market it is fundamental: 1) to analyze the requirements of the potential customer 2) to design the architecture of the solution and the type of products which can match the customer requirements 3) to configure each product correctly taking into consideration all the internal constraints. This procedure requires a lot of knowledge and experience which is very difficult to transfer to all the people involved in the product life cycle. The Product Offering Process Methodology, based on a Human Factors approach, is the answer to these needs. The final aim is to improve the ability of Olivetti Systems and Networks worldwide organization by means of a knowledge based system namely Leonardo International, designed for representing the proper updated knowledge in an integrated office environment and information system solution.

1. INTRODUCTION

1.1. Aims

On the part of an Information Technology (IT) supplier, the *Product Offering Process* (POP) involves all the issues to market a product quickly, homogeneously and widely. This process coexists with the company Product Life Cycle (PLC) exchanging knowledge and information. Because of the fact that the *interpersonal communication capabilty* seems, in many cases, to be the bottle neck in the knowledge transfer, a knowledge based system using Artificial Intelligence (AI) techniques may improve the effectiveness of the knowledge transfer among different actors inside the supplier's organization.

1.2. Approach

The key concept adopted is that for a valid commercial offer an a-priory understanding of the user requirements is fundamental. However, predicting all the new requirements derived by the introduction of a specific product in the particular work situation is quite difficult. In general, *a product gives rise to new needs* because it affects the complex interrelations between user, task and environment. The realities of a work situation evolution could be described as follows in two phases.
Acting phase: a set of new requirements induces a modification in the user's tasks and in the organizational environment. Such a modification can be supported by the current product offer.
Listening phase: a set task performed by a class of users in the organizational

environment gives rise to new requirements.
In order to support the POP in the first step, a frame has been defined in which the current product offer is properly linked to the new tasks and environments that meets the customer's requirements. In the second step, the *future requirements* can be hypothesized by the customer as soon as he perceives the reality of the new tasks in the new environment. The future requirements are precious hints for the specification of the product offer evolution. Applying AI concepts in order to support the POP by a Knowledge Based System (KBS) the efficiency and effectiveness of the Acting phase and Listening phase can be dramatically improved.

2. THE PRODUCT OFFERING CONTEXT

Selling IT means: 1) managing the relationship with the customer and 2) activating the necessary Company's skills for a) designing the offer strategy, b) planning all the internal activities, c) taking the commitments. Figure 1 shows a grouping in three main cooperating areas of the complex procedures that act in practice.

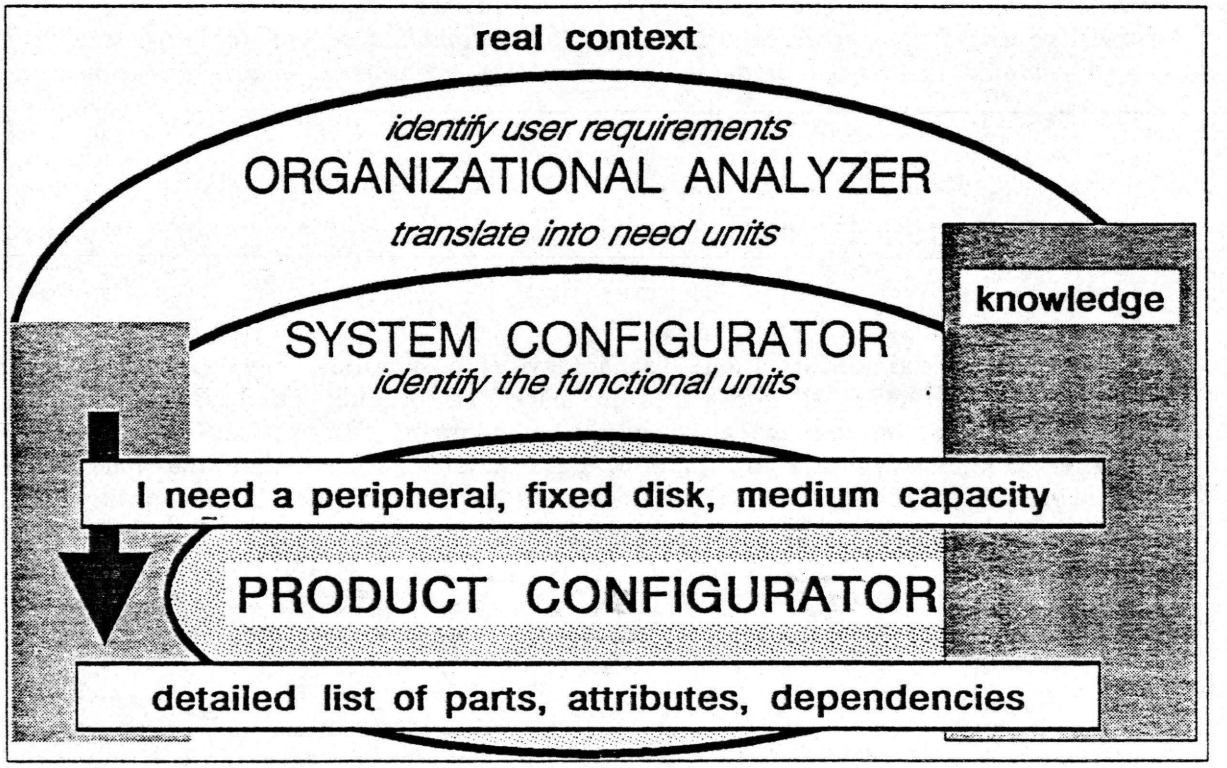


Figure 1: Areas of the Product Offering Process

The *organizational area* is where the real user's context has to be investigated, taking into consideration the peculiar characteristics of the tasks to which the users are committed, the personal skills and motivation of the users. The purpose is the identification of IT requirements in a wide range of very different environments, the changing role of the users, even more problem driven, is to be located and described by translating the specific findings into a more generalized and standard set of needs.

This *language of abstract representation* of the requirements can be further processed by linking this picture with the IT functionalities given from the composition of the product offer, this to obtain the *functional coverage* of the set of needs.

At the bottom a more technical procedure is to be taken: one of determining the hardware and software detailed list of the components of the products that are able to better satisfy the functional requirements.
In these phases all the aspects of dependencies, prerequisites, compatibility, performances availability and cost are to be seen, this at the level of the specific random codes that can be ordered.

3. THE ROLE OF THE ARTIFICIAL INTELLIGENCE

3.1. Demystifying Artificial Intelligence

The added value of AI is both cultural and technological. Cultural because AI has introduced the idea of putting knowledge, instead of data, as the focus of an information system. Technological because the *expert systems* not only incorporate data processing alghorithms, but also *methods of reasoning*. In other words, the basic pillars are not only "Data Bases" but also "Knowledge Bases". To talk about knowledge with the purpose of developing a profitable product in mind, requires a *pragmatic model* of what we intend for knowledge.

Many philosphers have tried to find a significant definition of knowledge and here we do not want to compare our with other ontological ones. So, cutting through the smoke and mirrors around AI, we can describe it in terms of a system which has an input and produces an output. For this purpose we have found the following definition correct: *the knowledge is such that it discovers, from the data, the proper actions which solve a problem.* Remaining in the definition world, the knowledge can be seen as a black box where data inputted returns effective actions.

3.2. Type of knowledge in the Product Offering Process

The knowledge involved in POP is not all of the same type. Some knowledge must be elicited from human experts, but other must be derived from the process which is built by different designers. A distinction must be made between the knowledge which is *created in the mind of one human expert* and another kind of knowledge where *the expert become just a medium* through which knowledge is transferred. This difference is very important to capture because it generates different project criteria. In the first case it is reasonable to think that such a knowledge will evolve quite slowly or, in general, not quicker than the experience of the expert. In the

second case this assumption is not true any more because this knowledge could change very quickly even more than the experience of single human experts. In fact this is always generated by a project team made up of several people working together, the expert, in this case, being just a person which synthesize the rules of the process defined by the team.

To explain this concept we have defined the first type of knowledge *Excellence Knowledge* (EK) and *Process Knowledge* (PK) the second. Designing a system based on EK means creating a model for knowledge representation which must be as homogeneous as possible with the reason style of the expert: the more the model representation is close to the expert's mind, the more the system will be effective. Meanwhile, designing a system based on PK by creating a model close to the mind of the expert could be misleading and not effective, because, as seen, the process is created and defined by a project team, only when the expert learns PK *become the effective expert* and only then a knowledge engineer can elicit knowledge from him using the model which has been defined on the base of his style reasoning. This mean that only at the end of this long chain we are able to acquire PK, but this also means that in the meantime PK *could have became obsolete*. The result is that it is not convenient to get PK using a model close to the way of reasoning of the expert, but it could be much more effective to have a model derived from the process itself. Coming back to the definitions, we can compare PK and EK applying the concept of *Deep Knowledge* (DK) and *Shallow Knowledge* (SK), where PK is a particular case of DK and EK is a particular case of SK.

4. THE KNOWLEDGE FLOW

The POP involves many areas in the company like Research & Development (R&D), Manufacturing, Marketing, Commercial Divisions, Information Systems, Customer Assistance, Logistics and so on. Each area gives its contribution in terms of creation and processing of knowledge and data. In general, it seems correct to assign to R&D the role of the main generator of new knowledge in the company, whereas Logistics generates and formalizes most of the data. This knowledge and data are also spread among different areas. For example, there is a knowledge flow that goes from R&D to Logistics, Marketing, Manufacturing. Furthermore, some areas, like Marketing and Customer Assistance, are expecially involved in knowledge transfering from R&D to the Commercial Divisions. In most organizations the state of the art is such that data is only partially transferred through IT, being knowledge still transferred through documents or verbally: the tools are drafted or partial and not integrated. The efficency is limited by the proliferation of paper documentation, meetings, telephone calls and people training. In this scenario the effectiveness depends too much upon personal efforts which, in general, are not homogeneously distributed in the organization.

5. THE STRUCTURE OF THE PROCESS

The product offering activity, according to the context shown in chapter 2, has been subdivided into three areas: 1) investigation of customer requirements (Customer Analysis), 2) design of the solution (Architectural Design) and 3) implementation of the solution (Product Configurator). The EK knowledge is involved in the Customer Analysis, both EK and PK in the Architectural Design and mainly PK in the Detailed Configuration. Because of the three different types of knowledge and the different types of activities, each area is covered by a dedicated Knowledge Based System (KBS) as shown in figure 2. A specific KBS for the administration of all the knowledge is also been defined and a personalized Customized Offer module has been added to gather the results and to produce the proper representation of the outputs.

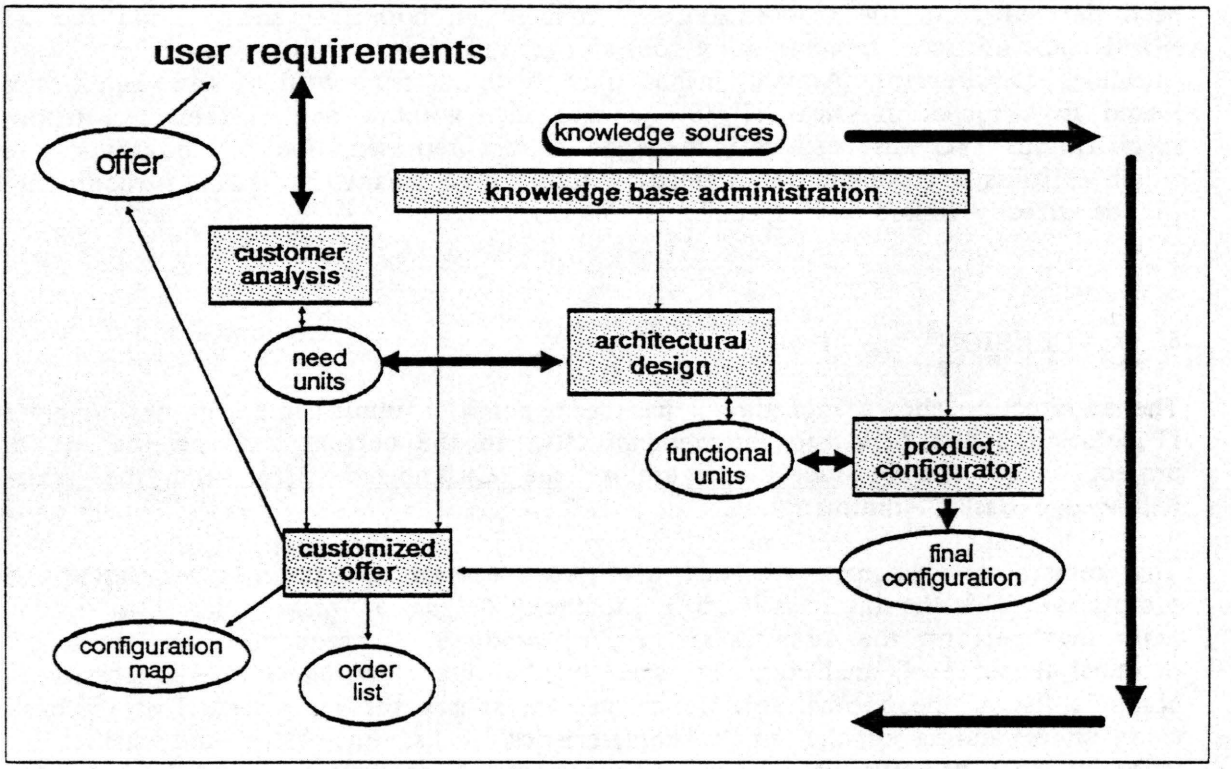


Figure 2: The Knowledge Based Systems

The effective action of the KBS is to find the product which both meets the Customer requirements and satisfies the costraints of the product itself. Considering that whilst many solutions could be available in theory, the optimal solution does not exist in practice because a well defined criterion of optimum which can be applied in all the situations is not yet available. Also if such a criterion exists for certain particular cases, it could be applied to select the optimal solution between all the possibilities, in the more general cases the worst condition is assumed: this will represent the sub-optimal solution which satisifies the constraints. For this reason, in order to converge one specific solution through all the possibilities, the architecture of the KBS has been designed for considering a *knowledge driven generate and test* approach.

In order to assure the cooperation between the areas of the system it has been necessary to define a format of representation of some fundamental item: we have called the expression of the customer requirements *Need Unit* (NU) and the description of the related functional architecture *Functional Units* (FU). This has been derived from the theoretical basis concept of both NU and FU. In fact the FU is an abstract representation of an atomic functionality which satisfies a Customer requirement. Atomic means that such a functionality can be directly linked to a set of physical HW/SW components without any further architectural specifications. NU has been defined as an astract representation of an atomic need which satisfies a Customer requirement. Atomic here means that such a requirement can be directly linked to a specific set of FU.

6. CONCLUSION

The architecture shown in figure 2 has been partially implemented in an integrated IT solution called *Leonardo International* (LI). In the current state of the art the project covers the Product Configurator, the Customized Offer and the related Knowledge Based Administration.

The remaining Customer Analysis and Architectural Design modules have been prototyped. This having classified a satisfactory *type of tasks* related to *specific users* that perform the tasks using *specific products*. Consequently, by applying a practical tecnique of analyzing the task and the user environment it is possible to obtain a list of the optimal core functions that satisfy the real users. This implies a clear understanding of the user's characteristics, skills, knowledge and tasks. In a word, in this prototyping activity the concept of *orgware* has been operationally applied, in considering: a) the characteristics, abilities and limitations of the users and b) the organizational settings and its critical characteristics, including the tasks that the users are committed to do. Orgware has been intended for all the activities involved in establishing requirements specification, training of the users, changing work procedures, etc.: in general all the activities which are closely linked to the organizational implementation of an IT system.

ACKNOWLEDGEMENTS

We wish to thank Giorgio Berutti, Giorgio Chiantore, Flora Resta and Piero Scaruffi who helped us in this effort and who gave contributions to this activity.

REFERENCES

[Berutti et al 89]
Berutti G., Mauriello G., Emiliani F. et al, Sistemi Informativi OS&N: situazione e prospettive, Olivetti Systems and Networks, Nov. 89

[Mittal et al 89]
Mittal S., Frayman F., Towards a Generic Model of Configuration Task, IJCAI proceedings, Detroit, Aug. 1989

[Butera et al 89]
Butera G., Chiantore G., Perotto P., Resta F., Leonardo, Olivetti Systems & Networks, July 89

[Allamanno et al 88]
Allamanno N., Saffi F., Progetto Configuratore Esperto, Olivetti Systems & Networks, Dic. 1988

[Baker 88]
Beker E., Expert System for Configuration at Digital: XCON and beyond, Communication ACM (vol. 31-32), Dec. 1988

[D'Appollonio 88]
D'Appollonio V., Olivetti Open System Architecture: sistematizzazione e razionalita' nella offerta informatica dipartimentale, in: Sistemi di Telecomunicazione, Dic. 1988

[Gaglio et al 88]
Gaglio, Puliafito, Perotto, Paolucci, Some Problems on Uncertain Knowledge Acquisition for Rule-based System, Decision Support System, International Journal, North Holland (vol. 4 n. 3), 1988

[Schafer 88]
Schafer G., Functional Analysis of Office Requirements: a Multiperspective Approach, Wiley, 1988

[Blacker et al 87]
Blackler F., Oborne D., Information Technology, The British Psychological Society, 1987

[Butera et al 87]
Butera F., Bolognani M., Zeller B., Tecnologie e problematiche organizzative, Cantiere di Informatica per le Professioni Aziendali, in: Zerouno, Feb. 1987

[Novara et al 87]
Novara F., Bertaggia N., Allamanno N., Usability evaluation and feedback to designers, in: Bullinger H.J. & Shackel B. (eds.), Human-computer Interaction (North Holland), 1987

[Wiseman 87]
Wiseman C., Strategia e Computer, CEDAM - Padova, 1987

[Bjorn-Andersen et al 86]
Bjorn-Andersen N. et al., Implementation of Office Systems, in: Verrjin-Stuart A.A. & Hirscheim R.A. (eds.), Office Systems (North Holland), 1986

[Mumford 86]
Mumford E., Using Computers for business success, Manchester Business School, 1986

[Wu et al 86]
Wu H., A Tool Kit for Costructing Knowledge Based System Configurator, AAAI proceedings, Philadelphia, 1986

[Bolognani et al 84]
Bolognani M., Corti E., La fabbrica del Software, Franco Angeli Ed., 1984

[Shortliffe et al 84]
Buchanan B.G., Shortliffe, Rule based Expert System, Eddison Wesley, 1984
[Mumford 83]
Mumford E., Designing Human Systems, Manchester Business School, 1983
[Damodaran et al 80]
Damodaran L., Simpson A., Wilson P., Designing Systems for People, NCC Ltd, 1980
[Rogers et al 76]
Agarwala R., Rogers E.M., Communication in Organizations, Free Press, 1976

Combining Natural Language and Hypermedia as New Means for Information Access

Giuseppe Carenini, Marco Ponzi, Oliviero Stock
Istituto per la Ricerca Scientifica e Tecnologica
38050 Povo, Trento, Italy
e-mail: stock@irst.it

1. Introduction

Artificial intelligence approaches to information access through natural language (NL) have yielded a lot of insight to the issue and a number of specific techniques; yet, so far they have not been sufficiently creative in seeing in the computer an agent with a large bandwidth of communication with the human user. In our view, natural language can become a fundamental paradigm of communication between human and machine, provided it is integrated with other channels, some of which might also be unavailable in human-to-human communication. Just to make the claim more concrete with an example: if we want to refer in a dialog to something mentioned before we would use a pronoun or a definite noun phrase, causing a difficult and nondeterministic process in the hearer (and probably in the speaker too); if we have the references available on the screen, we could explicitly refer to them through pointing. We believe that in the long run a new modality of natural language will emerge, after the spoken and the written one: the perspectives we are exploring for information access are just a window into that.

In this paper we shall begin with discussing multimodality and hypermediality and how they fit into an artificial intelligence view of information access. After that, we are going to analyze some issues in the integration of images with natural language and in the production of language, combined with hypermedial capacity as a means to provide information to the user in a flexible, individual and rich way.

All these ideas have emerged while developing a prototype of a dialog system connected to a videodisc that gives information about Italian frescoes of the Fourteenth Century. In the paper we shall refer to this work. The system, beside using language, shows images of frescoes and combines film sequences. Images are active so that the user may refer to items by pointing, combined with the use of linguistic demonstratives. The dialog may cause zooming into details or changing the focus of attention into other frescoes. The aim is not only to provide information. The system also has the goal of promoting other masterpieces: in the art field the potential experiencer is likely to build an interest when exposed to images in a coherent way.

2. Multimediality and hypermediality

Recent advances in digital video and audio real-time processing suggest that we will soon face a new revolution in the design of interactive systems (Fox, 1989).

The first effect of these advances is the feasability of the implementation of *multimodal* systems, i.e. of systems that use traditionally independent communication media (such as pictures, animation, sound, speech, printed text etc.) in an integrated way.

The generalization of the idea of hypertext to multimediality gives rise to the concept of *hypermedia*: "a style of building systems for information representation and management around a network of multimedia nodes connected together by typed links" (Halasz, 1988). Multimodality and hypermediality, more than other paradigms, put an emphasis on the possibility of realizing an intelligent interface that amplifies capabilities we have in nature (as opposed to trying to reproduce them). In fact, while it is true that people, in face-to-face communication, not only use spoken language but also take into account gestures, the visual environment and more, it is not so easy for people to reproduce or create images, and especially to make an integrated active medium out of the combination of images and language.

Intelligent multimodal systems are often used in interfacing a user with a dynamic process such as the simulation of the operations inside a factory (Cohen et al., 1989), a simulated battlefield (McLaughlin and Shaked, 1989) or the activities of U.S. Navy ships (Arens,1989). In such applications, the advantage of integrating multiple media in output is obvious, for instance, to explain sequences of operations or to display the status of a complex process. Similarly for input, pointing to images on a screen may individuate the objects involved in some desired action.

Hypermediality has recently opened interesting perspectives on the problem of accessing loosely structured information. Such systems promote a navigational, explorative access to multimodal information: the user, browsing around the network, is at the same time both exploring the network and searching for useful information. An hypermedia system can be used in different ways for different goals, just like a book (e.g. a manual) that can be read in a completely sequential way, quickly leafed through in order to get an idea of its content, or scanned in search of particular information.

3. How hypermedia and NLP can help each other

Many of the problems that prevent NL systems from providing useful interfaces stem from the current inability to handle the difficulties and misunderstandings that arise during dialogs. The study of actual human interactions suggests that the various forms of human dialogic communication can be seen as modifications to a single conversation's basic structure. The *organization of conversation* depends in essential

ways upon the material and social circumstances of human interaction, i.e. upon those features that characterize dialog as an example of *situated action*. Suchman (1987) underlines the importance of local control for the success of face-to-face human interaction: we can remedy the uncountable troubles that occur during communication not because we can predict all the possible problems, but because the organization of conversation provides a very powerful mechanism for exploiting the contextual information shared by the people taking part to a dialog. Multimediality offers a way of enlarging the share of the interactional situation that is available both to the user and the system, allowing for a better understanding between them. If both the participants can refer to a rich range of contextual knowledge, having the opportunity of shifting initiative in order to signal and remedy interactional problems, it should be possible to achieve a satisfactory simulation of the recover mechanisms of human interaction. For instance, if the user fails in understanding a natural language description identifying a character represented in a picture, the system might generate a kind of deictic action such as a graphical highlightening of the area of the picture representing the character.

An example of synergetic use of language and other media is deictic pointing to graphical representations of objects by means of a mouse or a touch screen. Pointing to objects in the world is one of the many interactional devices that we use in face-to-face dialog and which are completely unavailable when communicating with a machine by a single communication channel.

Hypermediality provides a powerful mean for the organization of highly heterogeneous and unstructured information. This kind of knowledge is not easily handled by natural language systems (and AI systems in general). In situations in which an AI application has a formal representation of only a subpart of the relevant domain knowledge, it is not possible to generate natural language text about information not explicitly represented in the system. A possible solution could be the dynamic generation of hypertextual (maybe hypermedial) nodes pointing to a canned hypermedia network about those areas of the domain which have not been formally represented.

We now turn to discuss how NLP can solve a major hypermedia problem. One of the problems of hypermediality underlined by Conklin (1987) is *disorientation*: hypermedia offer more degrees of freedom, more dimensions in which one can move, and hence a greater potential for the user to become lost or disoriented; the user has the problem of having to know where s/he is and how to get to some other place that s/he knows (or thinks) exists in the network.

One of the proposed solutions to the second aspect of the disorientation problem is the integration of a query facility within the system. Such a facility should provide a way of jumping inside the network without having to follow the predefined paths through it.

Halasz (1988) underlines that it is possible to identify two different types

of queries about an hypermedia network:

1) *content queries* allow to retrieve all objects (nodes or links) that satisfy some requirements. All objects in the network "are considered as independent entities and are examined individually for a match to a given query": content search ignores the structure of the network.
2) *structure queries* allow to retrieve a subnetwork matching a given pattern

NL can be a useful tool for handling such queries, if we assume that the system has some information about the knowledge presented by the different nodes and about the semantics of the structure of the network.

3.1. Towards a deeper complementarity

Both hypermedia technology and NLP are interesting paradigms for the development of intelligent interfaces of the future. Beyond the constructive local interactions just discussed, the two fields are starting to share common techniques and methodological foundations.

The necessity of an experimental and psychologically grounded approach is asserted in a large part of the literature about hypermedial systems. For instance, Nielsen (1990) describes the techniques adopted in the design of a HyperCard application and the research results extracted from the usability tests performed during the development phase. It would certainly be very useful if similar approaches were more common in the NL community. There are few exceptions to the general tendency of ignoring the experimental issues in the design of AI systems. In the field of NLP, one interesting possibility is the study of the practical consequences of Suchman's remarks about the relevance of conversational analysis for human-computer interaction. Frohlich and Luff (1989) have build an experimental system based on these ideas. Oviatt and Cohen (1989) have addressed the problem of evaluating the effectiveness of different forms of spoken and written linguistic communication and discussed the implications for the design of NL systems.

Many of the problems addressed in the research on the development of intelligent interfaces are connected to the need of tailoring the behaviour of the system to the characteristics of the particular user. User modeling is a fundamental area of research in dialog processing. The relevance of the field for NL understanding is connected to the need of representing the communicative context in which a sentence is uttered (Kass and Finin, 1988). In the field of NL generation, from content selection to lexical selection, from the choice of particular rhetorical structures to the choice of particular syntactic structure, a user model is a powerful knowledge source that drives the generation process. The generated text, tailored on the intended reader, is a more effective communicative act (Hovy 1987).

The hypertext research community begins to acknowledge the importance of user modeling. Conklin (1987) points out that a common difficulty arising in interacting with a hypermedia system is the *cognitive*

overhead caused by the number of links that it is possible to follow from each node. He then suggests a number of possible solutions based on the clever use of various multimodal techniques (e.g. a one to three line pop-up explanation of the side reference, or a graphical browser which shows the subnetwork into which a link leads). It would be very useful to be able to tell which links the specific user is less likely to follow, in order not to display them at all (or to display them with a lower degree of relevance). The development of powerful user modeling techniques could probably cause an impressing increase in the applicative usefulness of hypermedia systems. For instance, Fox (1989) claims that the coupling of multimediality and an accurate user model might dramatically improve the bandwidth and effectiveness of intelligent tutoring systems.

We believe that there is an important aspect that must not be underestimated in NLP sytems: the problem of providing to the user a model of the data available and possibly of the kind of queries that the user can eventually propose to the system. Most NL systems were proposing a paradigm for which NL was a front end to a DB. If the user was made familiar with the contents of the DB, the problem became unnaturaly limited.This is the case for instance of administrative DB's. The type of contents is generally well known to people that eventually had the possibility of using a NL front end through which they could propose the questions they would have had to express in SQL or similar query languages. If, on the other hand, one has a menu of choices that provide in every point the possible values of parameters in a request , the model is too restrict to accomodate a more personal view of the domain, and, obviously, the expressive power of requests is necessarily very limited.

The combination of the relative freedom provided by a natural language interface and a graphical presentation of some organized subdomains may be of potential impact. The suggestive power of a graph, or also of explicit promises of more information in the form of buttons may provide a bias towards guiding the view of the system and of the information it can provide, that the user is to adopt. And of course the user, before making a precise request, can explore "the surroundings".

This particular aspect needs new studies on the cognitive side. See also Slack and Conati in this volume.

4. First steps towards a new form of interface

The natural language processing community, naturally interested in applications in the information access area, so far has given too little attention to innovative situations in more complex computer-human interaction. Access to information within the NLP community has often been considered from two points of view:

- the data base tradition brought into an intelligent question answering system. In applications of this kind , there can be quite sophisticated input, but there is no specific user model and discourse management is accordingly limited;

- the dialog oriented approach, usually employed for expert systems, takes into account intention recognition, and more generally a user model in the context of some limited task.

We are building a hypermedial system with the goal of integrating these two AI approaches to NL interaction by dynamically switching from information seeking subdialogs to the role of an expert adviser. Kass and Finin (1988) point out that these two situations embody two symmetric aspects of natural communication: in the first case the user always has the initiative and s/he is responsible for the successful progress of the interaction; in the second the initiative will be handled by the system which will have to build a sophisticated user model in order to avoid possible misunderstandings. In our case, the user will have the responsibility for communication when exploring the domain, i.e. when asking for frescoes and other works of art with certain characteristics. The system will observe the behaviour of the user in order to build a model of his/her interest and of his/her degree of expertise in Italian art. It should take the initiative when it detects a user's goal which requires some kind of expert reasoning or when it detects a user's misconception.

Research on the design of interactive systems capable of extensively handling initiative shifts is of basic importance for the improvement of human-computer interaction. A system that assumes sole responsibility for communication will not give to the user the opportunity of interrupting it and try to recover from any trouble that could have arisen (Kass and Finin 1988). On the other hand, if the system just answers the queries of the user, without sharing the responsibility for communication, it will not even need a sophisticated user model and it will never attempt to solve any problem unless explicitly requested.

The idea of exploration dialogs fits well in this framework: if a user is facing a system that s/he has never used before it is very unlikely that s/he might have a clear idea of which formulation of his/her problem can be understood by the system. It is also possible that the user does not have enough information about the domain to be able to produce any clear formulation of his/her problem. The situation of the system is more or less symmetric: it does not know what the user is interested in and, if it knew, it does not know how to present relevant information in an optimal way, since it cannot yet consult any user model. So the interaction will initially be handled on a local basis; during this phase each participant will build a model of the other and will start setting up the goals that it/he/she will pursue during the progress of the dialog.

4.1 The AL Fresco System

In order to address the complexity of the interaction just outlined, our system has grown into a workbench where to integrate and test different communicative paradigms and media.

The AL Fresco system is an interface to information about Italian art of the 14th century. Such knowledge consists of:

- a KB expressed in a KLONE-like language called KRAPFEN (Franconi, 1990). It represents knowledge about features of the frescoes, painters, monuments etc.
- a NoteCards hypermedia network (see below) containing unformalized knowledge such as critical opinions on the paintings and their authors;
- a set of images and films stored in a videodisc.

We have considered the integration of NL and hypermedia both in input and output. In input our efforts have been focussed on combining the interpretation of NL deictic references with pointing to images displayed on a touch screen (Samek and Strapparava, 1990).

For instance the user can ask a question such as: *"Who is this person?"* while touching the image of a person appearing in the tcurrently displayed fresco. Both the linguistic and the graphical deixes may be ambiguous, but usually the correct integration of the information provided by the two channels allows the identification of the intended reference.

NL and hypermediality are coupled in the output phase as well. The system's answer is an hypermedia node containing a (possibly generated) text with links to both the hypermedia network and images from the videodisc. In particular, the output of the NL generation component is passed to the hypermedia module which transforms the plain text into an hypertextual entry point to the underlying hypermedia network. Rethorical schemata promote the introduction of individual contents for which, in that situation, it is sensible to associate buttons pointing to hypermedial information.

The hypermedia module is based on the NoteCards system. NoteCards is a general hypermedia environment developed at Xerox PARC. It was originally designed as "an authoring system providing its users with facilities for creating and modifying hypermedia structures", but its powerful programmer's interface makes it easy to integrate Lisp programs into the NoteCards environment and/or to integrate NoteCards into another Lisp-based environment (Halasz, 1988). The primitive constructs of the system are *cards* and *links*:

- Cards are usually displayed on the screen as standard Xerox Lisp windows containing a piece of text. NoteCards allows for the creation of new card types.
- Links are used to interconnect individual cards into networks; they are displayed as icons inside the substance of a card. Clicking the mouse in such an icon retrieves the destination card and executes the associated actions (usually the card is displayed on the screen).

Our system presents generated text as a card, providing the user a dynamic entry-point to the static base of hypertextual information. Presenting the user a mix of generated and canned text can be misleading. In fact the user could: (a) make references to canned text (the system is unable to understand); (b) overestimate the system's capabilities. The use of graphics overcomes this problem, making clear the difference between

the two kinds of text. Canned text can be presented in a radically different graphical format (e.g. a simulated open book). Also the integration between the NL generation component and the videodisc is achieved through hypertext technology. We have defined a new card type such that opening a card of that type causes the execution of a videodisc action: in order to access images or film sequences relative to an instance mentioned by the system, the user only needs to click on the link icon associated to its name.

5. Conclusions and future work

We have discussed multimodality and hypermediality within an artificial intelligence view of information access. Some issues in the integration of images with natural language and in the production of language, combined with hypermedial capacity, have then been analyzed. We have presented the experimental dialog system AL Fresco: an NL system connected to a videodisc that gives information about Italian frescoes of the Fourteenth Century, under development at IRST.

Up to now AL Fresco has been used as a workbench for the implementation of a bunch of ideas concerning the integration between NL and other communication modalities; in the near future we shall test the usefulness of such ideas for the design of systems that should operate in applied domains. In this perspective, psychologically founded analysis of explorative behaviour of users of an information retrieval system will become essential. In this volume, Slack and Conati present some preliminary results. We think that the interaction of hypermediality with AI technology opens a wide range of new perspectives for the development of intelligent interfaces and that it will ultimately lead to an innovative paradigm for man-machine interaction.

References

Arens, Y., Miller, L., Shapiro, S.C., Sondheimer, N.K., "Automatic Construction of User Interface Displays", *Proc. of the 7th AAAI Conference*, St. Paul, Minnesota, August 1988

Cohen, P.R. et al., "Synergetic Use of Direct Manipulation and Natural Language", *Proc. CHI 89*, Austin, Texas, May 1989

Slack, J., Conati, C., "A Framework for Exploring Exploration", *Proc. ECCE5*, Urbino, 1990

Conklin, J., "Hypertext: An Introduction and Survey", *IEEE Computer*, 20, 9, 1987

Fox, E.A., "The Coming Revolution in Interactive Digital Video", *Communications of the ACM*, Volume 32, n. 7, July 1989

Franconi, E., "The YAK Manual: Yet Another KRAPFEN", *IRST Manual n. 9003-01*, Trento, Italy, 1990

Frohlich, D.M., Luff, P., "Conversational Resources for Situated Action", *Proc. CHI 89*, Austin, Texas, May 1989

Halasz, F.G., "Reflections on NoteCards, Seven Issues for the Next Generation of Hypermedia Systems", *Communications of the ACM*, Volume 31, n. 7, July 1988

Hovy, E., "Generating Natural Language under Pragmatic Constraints", *PhD Thesis*, Yale University, March 1987

Kass, R., and Finin, T. ,"Modeling the User in Natural Language Systems", *Computational Linguistics*, Volume 14, n. 3, September 1988

MacLaughlin, D.M., Shaked, V., "Natural Language Text Generation in Semi-Automated Forces", *BBN Report n. 7092*, July 1989

Nielsen, J., "The Art of Navigating through Hypertext", *Communications of the ACM*, Volume 33, n. 3, July 1990

Samek-Lodovici, V., Strapparava, C., "Identifying Noun Phrase References, the Topic Module of the AL Fresco System", *Proc. ECAI 90*, Stockholm, Sweden, 1990

Suchman, L.A., "Plans and Situated Action", Cambridge University Press, 1987

AN ASSESSMENT OF THE CONSTRAINT SATISFACTION APPROACH FOR DESIGN : A PSYCHOLOGICAL INVESTIGATION

Françoise DARSES

C.N.R.S, Paris 8, U.R.A 1297, Equipe de Psychologie Cognitive Ergonomique, 2 rue de la Liberté, 93526 Saint-Denis, France
INRIA, Domaine de Voluceau, Rocquencourt BP 105, 78153 Le Chesnay Cedex, France

This paper presents a first assessment of the constraint satisfaction approach for design, from a psychological point of view. More and more CAD systems are constraint-based. The question is to determine if this view is compatible with the cognitive processes involved in design. An empirical study conducted in the domain of computer network design outlines how the designers deal with constraints. It is stressed that some of the features of the CAD systems are relevant to the characteristics of the problem solving process : the formalization of the constraint concept is consistent with its cognitive representation, the least committment strategy is made possible, distinction between the nature of constraints is provided. On another hand, some additional cognitive concepts to be taken into consideration in constraint-based systems are highlighted : the concept of constraint must be specified more acutely; the representation of plans of action and schemata should become available.

INTRODUCTION

When the designers report their design activity, they readily describe the problems they have to cope with in terms of constraints : the technological properties of the equipment, the priority to be given to particular characteristics of the problem, the order in which the actions have to be achieved, the specifications expressed by the customer --all these are viewed as constraints to be satisfied. From that point of view, an increasing number of CAD systems have been developed using a constraint satisfaction approach. The assumption of this approach is that the process of solving is viewed as the progressive reduction of an initially very broad search space. The problem is finding a means to achieve this reduction. A clever way of coping with this is to think that, as soon as a constraint linked to the problem is formulated, the object to be designed is partially described. Therefore, the question is how to manage the various constraints of the problem.

In the first part of this paper, a brief review of the constraint-based CAD systems is presented, as well as the principles of the constraint satisfaction approach. Then, the compatibility between the constraint-directed reasoning implemented in these systems and some of the cognitive characteristics of the design process are exposed. In the third part, some additional concepts to be taken into consideration for a constraint satisfaction approach of design are suggested.

1 CAD SYSTEMS BASED ON A CONSTRAINT SATISFACTION APPROACH

Domains of application

There are some professional situations where the problem is to find the best way of managing resources. It might be for instance a scheduling task or a planning task in the industrial domain. Here, the experts proceed by trying to limit the set of constraints linked to the problem. Fox et al (1982), report that experts use approximately 80 to 90% of their time determining the constraints in the environment that will affect their scheduling decision (use and resources to be needed, technical specifications, economical and human considerations, ...), and only 10 to 20% of their time actually constructing and modifying the schedules. Recent studies have been conducted in the domains of scheduling factory job-shops (Fox et al, 1982; Chandra and Marks, 1986), machining plans of mechanical parts (Descotte et Latombe, 1985), designing experiment plans (Oplobedu et al, 1989) or planning peptide synthesis (Janssen et al, 1989). They have attempted to deal with these problems by representing and managing the constraints involved in these activities.

Spatial arrangement problems are another example of a situation where some specific resources have to be properly allocated. In these problems, the number of objects and /or the number of the possible locations per object is too large to permit direct solution. From this point of view, in the domain of architecture, Alexander (1972, in Quintrand, 1985) proposed considering the design process in terms of requirements and elementary needs to be structured by the analysis of their mutual relationship. In other terms, the main work of the architect is seen as the formulation and interpretation of the specifications, which amounts to a description of the constraints linked to the problem. Many applications have been made of this theory which handle the architectural problem in terms of constraints. For instance, the program PROMET (Maroy, 1971, in Quintrand, 1985) provides a break down of the constraints by way of a tree representation. The sub-sets looked for are determined by the lowest branches of the tree, while the highest allow for the definition of three or four principal systems of constraints. A more recent approach of the architectural design using a constraint approach has been developed by Manago (1985) in a system named LEGO. The task of this system is to complete and modify the original sketch of a building so that all the constraints are respected. Various space planning problems have been examined and implemented such as the determination of protein structures (Brinkley et al, 1987), furniture arrangement (Berlandier, 1988) or bathroom design (Willey, 1981, in Brinkley et al, 1987).

The formalization of the concept of constraint

From the constraint satisfaction paradigm, the concept of constraint can be understood as a relation between variables. The idea is that there exists a set of variables, each to be instanciated in an associated domain, and there exists a set of constraints, limiting the set of allowed values for specified sub-sets of the variables (Nadel, 1986; Mackworth, 1987). Consequently, a constraint is viewed as the set of possible combinations between the values of its associated variables (Stefik, 1981). For instance, in the domain of computer network design (Darses, 1990a), the constraint "length-compatible" bounds the variable "type of cables, regarding their maximum length {thin -> up to 185 m; thick -> up to 500 m; optic fiber -> up to 3000 m}" and the variable "segment to be cabled { 0 to n meters}". Then, according to this relationship between the two variables, there are six possible

values of the constraint (see table 1) which represent six different partial descriptions of the network to be cabled.

---

insert table 1 around here

The constraint satisfaction approach distinguishes three operations acting upon the constraints (Stefik, 1981, Meseguer, 1989) : constraint formulation, constraint propagation and constraint satisfaction.

Constraint formulation is the addition of a new constraint in the design process. *Constraint propagation* is the creation of new constraints from old constraints. When constraints are propagated, they bring together new requirements from separate parts of the problem. These modifications can be the addition or the suppression of some values because of the formulation of the new constraint. For instance, the propagation of the constraint "length_compatible" expressed previously, creates a new constraint which could be named "environment_compatible". For this constraint, the type of cable to be chosen (thin, thick or optic fiber) depends on the cable location (inside or outside). Therefore, the propagation of the constraint "length_compatible" has triggered the formulation of the new constraint "environment_compatible"; by satisfying this new constraint, a restriction of the initial set of possible values is obtained. *Constraint satisfaction* is the operation of finding values for variables so that a set of constraints on the variables is satisfied. When a constraint is satisfied, a part of the solution is described more accurately and thus, the set of possible solutions is delineated more acutely.

2 COMPATIBILITY BETWEEN THE CONSTRAINT-DIRECTED REASONING IN CAD SYSTEMS AND THE COGNITIVE CHARACTERISTICS OF DESIGN

The CAD systems based on a constraint satisfaction approach have not been psychologically and ergonomically evaluated yet. There are some attempts to take into consideration the characteristics of the human problem solving processes, but most of the research efforts are aimed at improving the techniques and the methods (backtracking algorithms, consistency algorithms, cooperative algorithms, constraint propagation, etc).

The question is to determine if designers actually view the problem which they have to solve, in terms of a set of constraints to be satisfied. In order to make a first assessment of the compatibility between the constraint-directed reasoning formalized in CAD systems and the cognitive characteristics of the design process, an empirical study has been conducted in the domain of computer network design.

The field of investigation

In network design, the experts have at their disposal initial data, such as the type of the machines to be connected, their location, the plan of the building and some specific requirements listed by the customer. From these data, choices must be made (Darses and Falzon, 1989) : topological (location of the cables in the building), technological (specific properties of the network equipment), functional (architecture of the network).

An experiment has been conducted which consisted of asking six designers (3 experts and 3 novices) to deal with the same problem of network configuration. Actually, valuable information about the management of constraints is carried out through the comparison of the problem solving process from various levels of expertise. The data collected are verbal protocols obtained through simultaneous verbalization. For the purpose of assessment, these data have been described in terms of constraints. A methodology (Darses, 1990a) based upon the formalisms developed in the constraint-based systems (the formalization of the concept of constraint is depicted in the description of the rationale above) was applied. This study highlights some interesting findings (Darses, 1990a,b) and allows a first ergonomic assessment of the constraint-based CAD systems. The main outcomes are exposed in the following sections.

Constraint as a relation between variables

From a psychological point of view, the first question is to determine if it is relevant to formalize the concept of constraint as a relationship between variables. The analysis of the problem solving processes shows that half of the utterances in the verbal protocols can be expressed in term of a relation between variables.

Moreover, such a formalization of the constraint concept illustrates a remarkable characteristic of design situations : there is no unique solution to a design problem. Different final solutions can be drawn, each as good as the other, depending on the designer. Defining the concept of constraint as a set of possible combinations between the values of variables linked together accounts for the existence of numerous solutions to the same initial problem : as the instanciations of the constraints can differ according to the individual choices of the designers, many different solutions can be drawn.

Constraints and least committment strategy

The least committment strategy (Sacerdoti, 1977; Lebahar, 1983) covers the fact that the designers try to keep open as late as possible the choices to be made in the design process. This characteristic of the psychological behaviour of the designers is closely tied to the notion of constraint. By deferring the decisions as long as possible, designers avoid prematurate and inappropriate decisions which would narrow the possibilities and create unresolvable conflicts between sub-problems. Therefore, one can consider that the interactions between the sub-problems, the management of which appears crucial in design, can be seen as the expression of constraints being bound to the problem (Stefik, 1981). As a matter of fact, the conflicts which emerge in the course of design are often expressed in terms of constraints by the designers themselves. For instance, in network design, a conflict can arise between the hierarchical network architecture which is achieved and the budgetary limits which are overpassed. The definition of a constraint as a set of possible values to be progressively restricted in the course of the process, makes possible the least-commitment strategy. It also accounts for the implicit propagation of the constraints occurring in the problem solving process.

Formulation and management of constraints

The traditional constraint satisfaction approach works with a fixed number of constraints that are all known at the beginning. But the designers' behaviour is much different : when starting to solve the problem, the set of constraints is not entirely known. The verbal protocols collected in network design show that information about the specificities of the site to be cabled are asked for all along the process. Moreover, the constraints are usually not fully specified at the beginning but are progressively expressed during the problem solving process. For this reason, some authors suggest a progressive refinement of the constraints (Stefik, 1981) or a progressive introduction of the constraints into the design process (Janssen, 1989).

As a matter of fact, beyond the difficulty of expressing the constraints, a greater complexity arises from the management of constraints during the planning process. Since formulation, propagation and satisfaction of constraints can be broadly characterized as inferences in problem solving, the different roles these operations play in the solving process have been interpreted in the empirical investigation. The analysis of the protocols has stressed three occurrences : formulated constraint, constraint formulated and satisfied simultaneously, constraint formulated and satisfied later on in the process (the satisfaction is delayed). Moreover, it has been highlighted that these methods of managing the constraints differ greatly between the level of expertise of the designers, and according to the type of constraints. These differences in the constraint management should be taken into consideration in the development of CAD systems.

Validity constraints and preference constraints

Eastman (1970), analyzing the activity of architectural designers, notices that the competence of the designers depends on the type of constraint they deal with and on the way they have organized and structured these constraints in memory. This suggests a need to distinguish more accurately the type of constraints handled in the course of the design process. A way of characterizing the constraints is to take into account their importance and their priority in the design process. Actually, the designers know that it is not possible to achieve a solution where all the constraints would be equally satisfied. Thus, priorities have to be made, according to the features of the problem and according to the individual choices.

The necessity to account for the fact that the designers rate the constraints involved in the problem according to their importance or their priority has been introduced in the computational systems in several ways. Although it is not a constraint-directed system, Sacerdoti's program (1974) formalizes this evaluation knowledge with the introduction of criticalities . These are a set of procedures the role of which is to control the effects of any action which could jeopardize the plan. Criticalities are very similar to the notion of constraint, since they are the formulation of the specifications of the problem domain, their weight being the expression of their importance in the problem solving process. Descotte et Latombe (1985) attach weights to the constraints. The constraints are expressed as production rules : each right-hand side of the rule is weighted according to the importance of its satisfaction in the human expert's view.

Janssen et al (1989) have distinguished *validity constraint* from *preference constraint.* A validity constraint must be verified in the solution for it to be admitted. Most of the validity concerns the technological prescriptions, topological prescriptions or customer

specifications. For instance, the compatibility between the type of cable and the length of the segment to be cabled is a validity constraint. But the designers choose one solution rather than another in the set of all the allowable solutions, based upon preference constraints. Whether a preference constraint is applied or not, the object to be designed remains an allowable object. In networking design, a preference constraint may provide for instance the possibility of a further extension of the configuration.

The distinction between preference and validity constraints has been pointed out in the empirical study. What differs greatly between these constraints is the way they are posted. Preference constraints are expressed at the beginning of the solving process and are satisfied late in the problem solving process, while validity constraints are formulated and satisfied simultaneously in the process. Moreover, a remarkable difference between experts and novices is outlined in their competence of managing preference constraints : the novices are unable to handle these constraints by postponing their satisfaction, while this method of working is readily used by the experts.

A problem arises because the limits between preference and validity constraints are sometimes fuzzy. As a matter of fact, some constraints can be considered either upon validity or preference criteria. For instance, the final cost of the network is considered a validity constraint when the customer initially specifies a maximum cost figure. But cost can be viewed as a preference constraint if the designers impose upon themselves a "reasonable price" criterion, although it was not specified by the customer. This characterization of the constraints may also change during the course of the design process. We assume that the constraint-based systems must provide this ability.

3 ADDITIONAL CONCEPTS NEEDED

As shown in the previous section, some features of the constraint-directed reasoning implemented in CAD systems are compatible with the characteristics of the designers' reasoning. But the empirical study has also pointed out some aspects of the cognitive processes involved in design, which must be taken into consideration in these systems. Two facets must be enlarged so as to give an account of the psychological activity of the designers : a more accurate definition of the concept of constraint should be obtained and the concepts of action plans and schemata should be introduced.

3.1 Constraint

Deduced and Prescribed constraints

From a psychological point of view, it seems necessary to make a distinction between two kinds of constraints. As a matter of fact, some constraints have their value previously set, as for instance : "the building is six floor high" or "we want optic fiber to be used". These constraints differ from those the instanciation of which is submitted to the decision of the designers. For example, it is up to the designer to choose the type of cable according to the length of the segment to be cabled.

Formally, there is no reason to distinguish the "pre-instanciated" constraints from the constraints "to be instanciated". But some findings (Darses, 1990a) suggest that the psychological concept underlying is not the same. One have highlighted that the level of

expertise is correlated to the ability of handling the one or the other type of constraint : the experts make use of prescribed constraints ("pre-instanciated") for about 30% of the total of items, when novices use them for 20% only and rely their process more on deduced constraints ("to be instanciated").

Constraints and rules

Constraints are formalized (Sriram and Maher, 1986) in the CAD systems either as a relation among variables, or as rules (productions), or objects (schemas or frames). For instance, the system GARI (Descotte and Latombe, 1985; Descotte and Delesalle, 1986) implemented for generating the machining plans of mechanical parts applies a rule-based approach. For instance, a constraint is expressed as following : "If a hole H1 opens into another hole H2, then one is recommended machining H2 before H1 in order to avoid the risk of damaging the drill".

However, a problem arises from this formalization of the concept of constraint. Let us see the following utterances. Formally, they can be expressed as production rules :

(1) "If the length of the segment to be cabled does not overpass *n*, then the type of cable is *x*"

(2) "If the cable is a thin cable, and if the machines are connected through a BNC connector, then the length of the cable is the length of the segment to be cabled plus the number of connections to be done multiplied by ten meters".

The difficulty which appears here is that these two utterances do not have the same operative meaning. As a matter of fact, the knowledge described in (1) can be used for network design by triggering either the "type of cable" variable or the "segment to be cabled" one. The production rule is not oriented and it makes sense to say :

(1') "If the type of cable is *x*, then the length of the segment to be cabled does not overpass *n*".

On the other hand, the utterance (2) is oriented. It does not make sense to try to reverse the sentence. It would loose its operative property. Actually, the knowledge contained in the reversed sentence is unlikely to be useful for the designers :

(2') "If the length of the cable is the length of the segment to be cabled plus the number of connections to be done multiplied by ten meters, then the cable is a thin cable and the machines are connected through a BNC connector".

This suggests that the designers might handle differently these two kinds of cognitive entities. Thus, for a constraint satisfaction approach of design, it seems necessary to distinguish on one hand the *constraints* (non-oriented relationship between variables) and the *rules* (oriented relationship). A CAD system based on a constraint satisfaction approach should give to the designer the possibility of dealing in different ways with the expression of constraints and the expression of rules.

3.2 Other units

Since the prominent part of constraint has been pointed out in the reasoning of the designers, it is tempting to see the process of design as the development of formulation and satisfaction of constraints, the succession of which shapping progressively the object to be designed until the final solution is reached. But some steps of the solving process are unlikely to be interpreted in terms of constraint. The analysis of the protocols show that the cognitive activity occurring during design cannot be reported only by way of constraints. There are at least two additional cognitive entities which are triggered simultaneously with the constraints : the action plans and the schemata.

Action plans

The analysis of the protocols has stressed that the designers build their reasoning process from a set of action plans (Falzon and Darses, 1990). The assumption is that the designers have in mind a set of action plans, which are triggered in the course of the design process. For network design, three action plans have been defined. The top-down representation of an action plan (see figure 1), does not prevent bottom-up steps to be developed. How the constraints trigger or act on the development of action plans has not been investigated yet. Here is an example of an action plan applied by the network designers.

---

insert fig 1 about here

Schemata

Another cognitive unit to be called up in the course of the design process is formalized as a schema. We assume that the designers represent to themselves a set of "design objects". The description of the current object to be designed is build on one of these schemata. In other terms, these schemata are a kind of blue-print which provide a representation of the successive states of solution. Each of these objects have peculiar features, which can be formalized as the variables of the associated schema. For instance, in computer network design, the designers handle with the object "backbone", which can be formalized as follows :

---

insert tab 2 about here

The constraints act on these schemata so as to give appropriate values to the variables. The constraints are used to manage the relationship between variables belonging either to a same object or between two different objects. For the object "backbone" described above, it exists a constraint which bounds the variable 1 and the variable 2 (see the constraint environment_compatible), but it exists also a constraint which bounds the variable 5 of this object with another one (not depicted here) named "sub-network".

CONCLUSION

This paper highlights that many features of the constraint-directed reasoning, as it is implemented in the CAD systems, are compatible with the characteristics of the cognitive processes involved in design. On another hand, this first investigation stresses that the constraint satisfaction approach should take into consideration additional concepts so as to improve the psychological validity of the systems. These enlargements would likely provide some ergonomic advances for the implementation of CAD systems.

REFERENCES

Berlandier, P. (1988) Intégration d'outils pour l'expression et la satisfaction de contraines dans un générateur de systèmes experts (Report #924). Sophia-Antipolis, France: INRIA.

Brinkley, J., Buchanan, B., Altman, R., Duncan, B., Cornelius, C.(1987) A heuristic refinement method for constraint satisfaction problems. Knowledge systems laboratory (Report n° 87-05). Computer science department, Standford University, 1987, January.

Chandra, N., Marks, D. (1986) Intelligent use of constraints for activity scheduling. Applications of IA in engineering problems.

Darses, F., Falzon, P. (1989) The design activity in networking. General remarks and first observations (Technical report). Rocquencourt, INRIA, March.

Darses, F. (1990a) Gestion de contraintes au cours de la résolution d'un problème de conception de réseaux informatiques (Report #1164). Rocquencourt, France: INRIA

Darses, F. (1990b) Constraints in design : towards a methodology of psychological analysis based on AI formalisms. Communication accepted for presentation at Interact'90. Cambridge, August 27-31.

Descotte, Y., Delesalle, H. (1986) Une architecture de système expert pour la planification d'activité. Proceedings of Sixth International Workshop Expert Systems and their Applications, Avignon.

Descotte, Y., Latombe, J.C. (1985) Making compromises among antagonist constraints in a planner. Artificial Intelligence 27, 183-217.

Eastman, C. (1970) On the analysis of intuitive design processes. Emerging methods in environmental design and planning. Gary Moore (Ed.) Proceedings of the design methods group. First International Conference, Cambridge, Mass.: MIT Press.

Falzon, P, Darses, F. (1990) Conception de réseaux informatiques: une approche psychologique. Communication accepted for presentation at 01 Design, Première table ronde nationale sur les nouvelles méthodes et paradigmes de modélisation de processus de conception créative. Cabourg, October 21-23.

Fox, M.S., Allen, B., Strohm, G. (1982) Job-shop scheduling: an investigation in constraint directed reasoning. Proceedings of AAAI-82, Pittsburgh.

Janssen, P., Jégou, P., Nouguier, B., Vilarem, M.C. (1989) Problèmes de conception: une approche basée sur la satisfaction de contraintes. Proceedings of Ninth International Workshop Expert Systems and their Applications, May 29-June 2, 1989, 1, 71-84.

Lebahar, J.C. (1983) Le dessin d'architecte. Roquevaire: Parenthèses.

Mackworth, A.K. (1987) Constraint satisfaction. In S.Shapiro (Ed), Encyclopedia of Artificial Intelligence. New-York: Wiley, 205-211.

Manago, C. (1985) Lego: un système qui traite des contraintes. Proceedings of Cognitiva 85, Application à la CAO dans le bâtiment. Paris, June, 4-7.
Meseguer, P. (1989) Constraint satisfaction problems: an overview. AICOM, 2 (1), March 1989.
Nadel,B.A.(1986) The general consistent labeling (or constraint satisfaction) problem (Technical Report). Department of Computer Science, University of Michigan.
Oplobedu, A., Marcovitch, J.,Tourbier, Y. (1989) CHARME: un langage industriel de programmation par contraintes, illustré par une application chez Renault. Proceedings of Ninth International Workshop Expert Systems and their Applications, May 29-June 2 1989, 1, 55 -70.
Quintrand, P (1985) La conception assistée par ordinateur en architecture. Paris: Hermès.
Sacerdoti, E.D. (1974) Planning in a hierarchy of abstraction spaces. Artificial Intelligence, 5, 115-135.
Sacerdoti, E.D. (1977) A structure for plans and behavior. New-York: Elsevier.
Stefik, M. (1981) Planning with constraints (MOLGEN: Part 1). Artificial Intelligence, 16, 111-140.

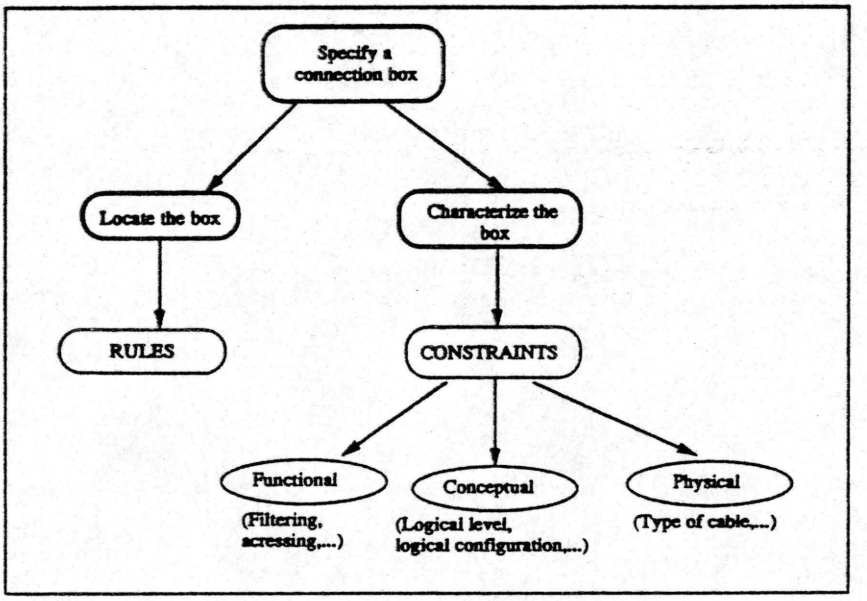


Figure 1. An action plan for specifying a connection box

type of cable		thin	thick	optic fiber
segment to be cabled	-> 185 m	x	x	x
	-> 500 m		x	x
	-> 3000 m			x

Table 1 The description of the constraint "length_compatible"

backbone
variable 1: location of the cable variable 2: type of cable variable 3: junction points variable 4: logical level of the junction points variable 5: connection boxes variable 6: location of the connection boxes variable 7: physical configuration of the backbone

Table 2 The representation of the object "backbone"

QUALITY EVALUATION EXPERIMENTS OF ITALIAN SPEECH SYNTHESIZERS

C. DELOGU, A. PAOLONI, P. POCCI

Fondazione Ugo Bordoni

Besides the more reliable interfaces such as keyboard, mouse, touch screen, and so on, an important role in man-machine communication can be played by the vocal interface, that is, the interface generally used in man-to-man interaction. In the paper we consider only an aspect of the vocal interface: speech synthesis. A large diffusion of speech synthesis applications (e.g. for information services, like weather forecast, traffic information, and so on) is foreseeable. To allow and facilitate this diffusion it is necessary to develop reliable and valid performance measures to compare different systems for different applications. Although much work on this argument has been done, precise methodologies for evaluation of text-to-speech synthesis systems are yet to be defined.
The purpose of the paper is to contribute to this field by proposing a Reaction Time Measure as a good measure to evaluate voice output devices. The results of two experiments on quality evaluation of Italian synthesizers using the time taken by subjects to recognize a word generated by a speech synthesizer are reported.

Introduction

All the studies on the perception of synthetic voice show that in comparison with the natural voice, synthetic speech is a poor signal, which provides less acoustical information and requires from the listeners a greater effort to be understand.

There are many different aspects of synthetic speech that can be evaluated: intelligibility (how much identifiable the linguistic message is), naturalness (how much the synthesized speech is like a natural voice), acceptability (the overall user's satisfaction in a

specific task), and quality (the overall system's performance evaluation). Among these aspects, intelligibility and quality are those generally more taken into consideration by evaluation methodologies (Pols 1989).

Intelligibility is evaluated at the segmental and supra-segmental level of speech. With regard to the segmental level the intelligibility of single phonemes or speech sounds, using simple syllables or word forms like CV, CVC, or VCV, is evaluated with subjective tests like the Diagnostic Rhyme Test (DRT) (Voiers 1983), or the Modified Rhyme Test (MRT), where words that differ only with respect to their initial consonant are used with a forced choice response of 2 or 6 alternatives respectively. Many objections to these tests for speech synthesis evaluation have been raised. The main objection is that with these tests consonants are only tested in initial position, which is a rather unusual context in running speech. New kinds of diagnostic measures to evaluate segmental aspects of speech have been proposed, as for example the Nonsense VCV Test developed at the Department of Speech Communication of the Royal Institute of Technology in Stockholm (Carlson et al. 1989), or the Cluster-Based-Rhyme-Test constructed at the Ruhr-University in Bochum (Jekosch 1989).

With regard to the supra-segmental level the intelligibility evaluation tests concern word stress, word duration, syllable structure, sentence accent, intonation, and so on. In these tests the subjects hear a series of phrases or sentences pronounced by the synthesizer, and after each phrase or sentence, they have to repeat back or transcribe what they have heard, or answer some questions. The problem with this kind of test is that the task is facilitated by contextual information (mostly semantic information) that allows the subjects, on the basis of their hearing part of a sentence, to guess as to the remainder of the sentence. To reduce this contextual cue, some work on supra-segmental aspects evaluation has been carried out using syntactically correct but semantically unpredictable sentences (Pisoni 1981, Grice 1989). In

these sentences, the predictability of each word is extremely low. The only cue the listener has is syntactic.

Different kinds of test to evaluate the overall quality of synthetic speech have been proposed. The common characteristic of these tests is that they are subjective (i.e. they imply listener judgements) and global (i.e. they make overall comparisons between one specific system and one or more other systems, with respect to their global performance). These evaluation methods are often derived from psychophysics like for example pair comparison, categorical scaling, and magnitude estimations. In pair comparison the listener's task is to make comparative judgement for all possible pairs of stimuli (spoken sentences). Category scales require the subject to indicate a judgement about the speech by using an 'interval' scale made of numbers or adjectives. The magnitude estimation procedure consists of presenting two sentences to the listeners and asking them to assign a number (decimals, fractions, or whole numbers) that is equal to the magnitude of their impression of the judged attribute. These methodologies used to evaluate the overall quality of speech can be also used to evaluate the naturalness and the acceptability of synthetic speech (Pavlovic et al. 1990).

General methodological considerations

We have run two experiments to study the Reaction Time measure (RT) as a methodology to evaluate the overall quality of synthetic speech. In the experiments the time taken by subjects to recognize a word generated by a speech synthesizer is used as a measure of the quality of different systems. With respect to the classification of evaluation methodologies given before, the RT can be considered as a global measure to make overall comparisons among different synthesizers, with respect to their global performance.

In the two experiments we used both natural and synthetic speech. The natural speech was produced by female and male speakers. The synthetic speech was produced by two different synthesizers: the Infovox system with both female and male voice, and a text-to-speech synthesizer developed at our laboratory (the FUB system) with a male voice only.

36 syntactically correct but semantically unpredictable sentences were prepared, with 5 different syntactic structures. A target word that had to be recognized by the subjects was decided for every sentence. There were 12 target words, each repeated in three different positions (initial, central, and final position). A first group of 18 sentences (the first 6 target words in the three positions) was pronounced by both the male speaker and the Infovox system with male voice. The other 18 sentences were pronounced by both the female speaker and the Infovox system with the female voice. The FUB system pronounced all 36 sentences. In this way 108 recorded sentences were pronounced.

20 subjects (10 females and 10 males, 10 per test) participated in the experiments. Subjects were told that before hearing a sentence they would see a word on the screen, and they would be asked to push a key when they heard that word. Each subject heard all 108 recorded sentences, that were randomly ordered. The reaction time employed by every subject to recognize the target word was recorded.

Experiment 1

In the Experiment 1, 10 subjects had to recognize the target words which were distributed in the pronounced sentences in initial, central, and final position.

The results showed that subjects employed less time to react to the target word when it was pronounced with natural speech, and with respect to the two synthesized voices they employed less time to recognize the target word pronounced by the FUB system than that pronounced by the Infovox system. Furthermore, a shorter reaction time was noticed when the

target word was at the end of the sentence, because in this case the attention of the listener is very high and syntactic constraints on recognition had accumulated by that time (see table 1).

Table 1

Reaction time (in sec.) employed by subjects to recognize a target word (in initial, central, or final position) in sentences pronounced by the Infovox System, the FUB System, and by a natural voice.

SYSTEMS / WORD POSITION	SYNTHESIZERS		HUMAN VOICE
	INFOVOX	FUB	
INITIAL	0.53	0.52	0.45
CENTRAL	0.47	0.39	0.35
FINAL	0.48	0.36	0.29
AVERAGE	0.49	0.42	0.36

Experiment 2

Even if the results of the experiment 1 are statistically meaninguful, the difference among the reaction times obtained for the three different kinds of speech is quite restricted.

We have run a second experiment, in which 10 subjects were asked to recognize a target word that sometimes was not in the sentence. We expected that in this way subjects would employ more time to recognize the target word in the sentence and that therefore the results in the various conditions would be more differentiated.

Table 2 shows that the hypothesis on an increase in reaction time has been confirmed, but the reaction times are not very different in the three conditions.

Table 2

Reaction time (in sec.) employed by subjects to recognize a target word that sometimes was not in the sentence.

SYSTEMS / WORD POSITION	SYNTHESIZERS		HUMAN VOICE
	INFOVOX	FUB	
INITIAL	0.61	0.54	0.55
CENTRAL	0.50	0.47	0.41
FINAL	0.61	0.50	0.41
AVERAGE	0.57	0.51	0.46

Discussion

After these two experiments, we have repeated experiment 1 using not the RTM, but the Mean Opinion Square Test (MOS). The MOS test consists in asking to subjects to give a quality score according to a judgement scale (EXCELLENT/ GOOD/ FAIR/ POOR/ BAD) that appears on the screen after each spoken sentence.

The aim of this test was to make a comparison between the RT and the MOS as quality evaluation measures. As we expected, the results indicated that the most favourite voice was the natural

voice. Between the two synthesizers the less favourite one was the voice generated by the Infovox system (see table 3).

Table 3
Average quality judgements given by subjects for sentences pronounced by the Infovox System, the FUB System, and by a natural voice. Number 1 = BAD; 2 = POOR; 3 = FAIR; 4 = GOOD; 5 = EXCELLENT.

SYSTEM	SYNTHESIZERS		HUMAN VOICE
	INFOVOX	FUB	
QUALITY JUDGEMENT	1.7	2.3	4.7

A comparison between the data obtained with the two different evaluation methods is shown in fig. 1.

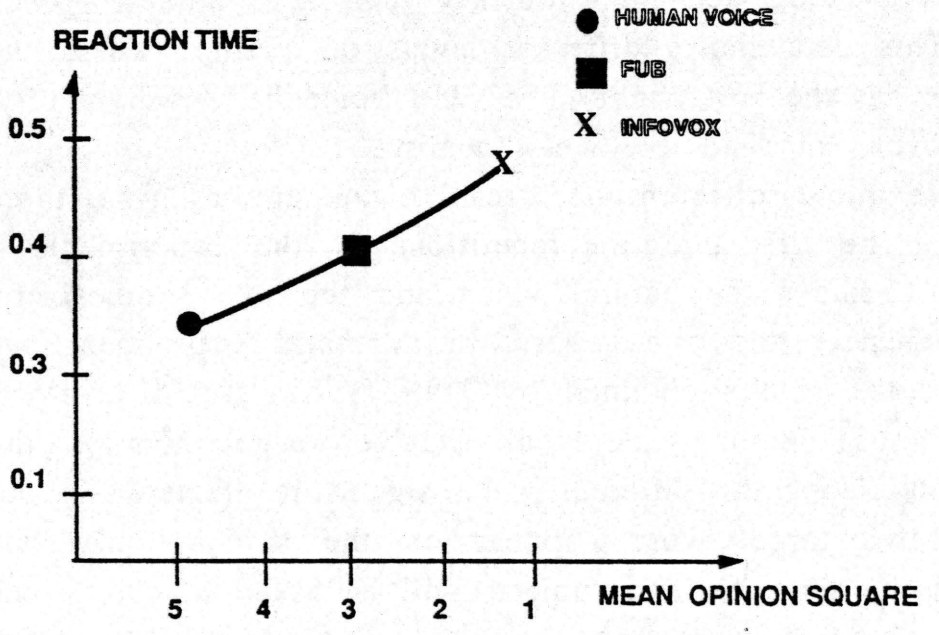


fig. 1: comparison between the RT and the MOS measures

We think that the high correlation between the RT and the MOS (see table 4) attests the validity of the RT as a global quality measure of speech synthesizers, and furthermore, since to give meaningful results the MOS measure needs a large number of subjects, the RT could replace the MOS.

Table 4
Correlation coefficients between the RT and the MOS measures

CORRELATION COEFFICIENT	τ
τ INITIAL	0.99
τ CENTRAL	0.87
τ FINAL	0.88
τ MEAN	0.93

In the two RT tests the target words in the natural voice sentences were recognized more quickly than in synthetic speech sentences. This data shows different levels of quality: where the natural voice has the best quality, the FUB voice has lower quality; and the Infovox voice has the worst quality.

To have more differentiated results we are running three experiments. The first two are repetitions of the experiments 1 and 2 using besides the natural voice and the two synthesizers, another synthesizer and two coders. In the third experiment, we want to test the results obtained by Pratt (1981) that if subjects have to choose among more alternative target words, the differentiation among the different vocal signals is greater. In our experiment two target words appear on the screen, only one contained in the sentence, and subjects will be asked to choice one of them.

We are also planning some experiments to compare the RT measure and the Magnitude Estimation measure, using 4 synthesizers and four coders.

References

Carlson, R. and B. Granstrom, 1989. 'Evaluation and development of the KTH test-to-speech system on the segmental level'. Proceedings of the ESCA Workshop. Noordwijkerhout, The Netherlands, 1.3.1-1.3.4

Grice, M., 1989. 'The assessment of synthetic speech intelligibility using semantically unpredictable sentences'. Proceedings of the ESCA Workshop. Noordwijkerhout, The Netherlands, 1.5.1- 1.5.4.

Jekosch, U., 1989. 'The cluster-based rhyme test: A segmental synthesis test for open vocabulary'. Proceedings of the ESCA Workshop. Noordwijkerhout, The Netherlands, 1.4.1-1.4.4.

Pavlovic, C.V., M. Rossi, and R. Espesser,1990. Use of the magnitude estimation technique for assessing the performance of text-to-speech synthesis systems. Journal of Acoustical Society of America Vol 87 N 1, 373-382.

Pisoni, D.B., 1981. Perceptual evaluation of voice response systems: intelligibility, recognition, and understanding. Research on Speech perception. Progress Report 7, Indiana University.

Pols, L.C.W., 1989. 'Assessment of text-to-speech synthesis systems'. In: A. Fourcin, G. Harland, W.J. Barry, V. Hazan (eds), Speech Input and Output Assessment: multilingual methods and standards. Chichester: Ellis Horwood, 53-81.

Pratt, R. L., 1981. On the use of Reaction Time as an Intelligibility Measure. British Journal of Audiology 15, 253-255.

Voiers, W.D., 1983. Evaluating Processed Speech Using the Diagnostic Rhyme Test. Speech Technology, Vol 1, No 4, 30-39.

The Activity Theory as an Alternative Framework for HCI Research and Design

Kari Kuutti
University of Oulu, Dept. of Information System Science
Linnanmaa, SF-90570 Oulu, FINLAND
e-mail: tolkkéfinou (EARN), kuuttiétolsun.oulu.fi (INTERNET)

Abstract

During the end of 1980s several attempts have been made to find a new theoretical framework for to guide the HCI research and design. The cause of this search has been the serious problems in applying the mainstream framework, the information-processing model -based cognitive science.
The paper presents a novel approach, based on Activity Theory, and discusses its possible usability in HCI research and design.

1. Introduction

Human-Computer Interaction (HCI for short) has existed quite a long time as a research domain and gained a reputation as one of the central elements in designing computer applications. There exists several international journals covering the domain, several international conferences are been held every year and a huge number of books around the topic has been published. Many, if not most, curicula for software design professionals contain a course in HCI. Given this record, one would assume that there exists a well-established body of harmonious scientific knowledge, covering the basic foundation of the discipline. At first sight this belief seems to be valid, because apparently HCI seems to be based on the application of the information processing branch of cognitive psychology and its multidisciplinary cousin, cognitive science. For example, John M. Carroll wrote only some years ago in his preface of that time state-of-the-art survey: "One of the most active new research areas in cognitive science is that of human-computer interaction" and "The chapters in this volume provide an interim report on the project of establishing an applied science of human-computer interaction grounded in the framework of cognitive science" (Carroll 1987, p. ix-x).

This harmony is fallacious, however. During the late 1980s a debate against the use of cognitive science as the foundation of HCI has surfaced,

criticizing even the very basic assumptions. Examples from existing problems and criticism can be found e. g. Bannon (draft), Bannon & Bødker 1989, Bødker 1987, Ehn 1988, Suchman 1985, Thomas & Kellogg 1989, Winograd & Flores 1987 and Witheside & Wixon 1987. Both the object and methods of the "mainstream" research has been questioned. Critics would like to expand the research object towards users and his work tasks in real life. Methodologically the "Cartesian" ideal of cognitive science - borrowed from natural sciences - has been seen unable to penetrate the human side of the interface. Now, in the beginning of 1990s we thus have both the established, cognitive science -based "orthodoxy" and the emerging, although diverse "opposition"[1]. The opposition has manifested its existence in s.c. "Kittle House workshop" in 1989, the results of which will be published in Carroll (forthcoming). Although it is certainly too early to speak of a shift in paradigm, we are anyhow witnessing some kind of crisis in HCI research area: "Currently, a reanalysis of HCI as a practical and scientific endeavour is underway" (Carroll et al. 1990, p. 1). What has been the main driving force behind this development?

According to the literature there are several problems and at least the following ones can be named:

- the problem of contextuality
- the problem of lacking focus
- the problem of needed multidisciplinarity
- the problem of a gap between research and practice.

These are discussed in more detail below.

The problem of contextuality.

The critics are quite unanimous in that the basic model of human-computer interaction - a well-defined communication between two information processors - is unduly narrow and cannot grasp the essence of a situation. Some kind of context should be included, but it seems to be difficult to do that within the conceptual framework of cognitive science. In any case, the domain of design and research is expanding (Grudin 1989b).

The problem of lacking focus.

The on-going debate has caused some confusion, and many different viewpoints - partially overlapping, partially conflicting - have been presented as guides to HCI development (and thus also research). Which one should be selected as the dominant one to which all the others should be related? Or is this possible at all - and if not, how shall we then proceed?

The problem of needed multidisciplinarity.

Studying the "context" of an interaction brings forth topics which definitely belong to more than one discipline, and it is generally accepted that HCI research should be able to take these into account and use them as parts

---

[1] It is interesting to note, that there is a similar ongoing discussion within information systems design in general, see e. g. Mumford et al. 1985, Budde et al. (in press) and Klein & Hirschheim 1987, and even within social sciences as a whole, see Reason & Rowan 1981. These connections cannot be elaborated here, however.

of the whole picture."To make an adequate analysis, we need to take multiple disciplinary perspectives - psychology, linguistics, anthropology, sociology and computer science - *simultaneously*" (Carroll 1987, p. xi) or "An interdisciplinary approach is needed to increase our HCI knowledge base" (Fischer 1989). But that is not so easy: if research results from several different disciplines should be used, how they can be fitted together? Every one is using his own frameworks and concepts, which rarely match each other or are even comparable to any marked extent.

A problem of the gap between theory and practice.

Indeed, although HCI as an academical discipline is flourishing, it is quite hard to find signs from its impact in design, as it was found in one recent survey: "None of the designers interviewed reported ever having used any HCI design and evaluation technique, some stated that they were unaware of the existence of such techniques" (Bellotti 1988, p. 23.). This inability of cognitive science to give practical advice has been certainly one of the most influential factors triggering attempts for to search other basis for research and design. (See Bannon (draft), Bannon & Bødker 1989, Witheside & Wixon 1987).

Because of the problems caused by the defects of cognitive science there has been a search of an alternative theoretical framework to find a new foundation for HCI research and development. It has resulted a variety of suggestions, like Heidegger's philosophy of "Being" and self-organization theory by Maturana (Winograd & Flores 1987), or Wittgenstein's language philosophy (e. g. Ehn 1988), and the search goes on.

The purpose of this paper is to introduce shortly one more alternative theoretical framework, namely that of the Activity Theory. Thus far its applications in HCI research have been quite modest, but it is worth of a closer look. Due to the space limitations the introduction will be brief indeed and it is not possible to compare the Activity Theory with other existing alternatives[2].

2. An Introduction to the Activity Theory

"Activity Theory" is in fact not a monolithic "theory" but more like a shared framework, an emerging multidisciplinary school of scientific thought connected by the central category of activity[3]. The original foun-

---

[2] The interested reader is adviced to consult e. g. Kuutti (forthcoming), Raeithel (in press), Wertsch (1981), (1985), the proceedings of the 1st Intl. Congress of Activity Theory (Hildebrand-Nilshon & Rückriem 1988-89) or some classics, like Leontjev (1974), (1978).

[3] The concept of activity has its roots in 18th and 19th century classical German philosophy, from which it was adopted into materialistic philosophy and finally elaborated by Marx. Because of this heritage a certain amount of knowledge of Marxist terminology is needed in order to avoid misunderstandings. The philosophical concept of activity, and also other ideas of Marx were taken as a

dation of Activity Theory was laid by L. S. Vygotsky during the 1920s and early 1930s as a "cultural-historical" school of psychology. His work was continued by A. N. Leontjev and A. R. Lurija who developed his ideas further and started to use the term "activity". It was believed that individual existence and actions were insufficient as a basis for a theory of human psychic functions and that a contextual grounding was needed. Thus the concept of activity was formulated as a basic category for mediating between the human psyche and the environment.

Although the concept of activity as a scientific tool was first formulated within Soviet psychology, the area of its applications is now far larger - both scientifically and geographically[4].

Perhaps the basic idea is that there exists a "fundamental type" of context, which is called an activity. It is meaningless to study essentially human qualities using a smaller object of research, because without that basic context one cannot grasp the essence of the phenomenon.

The concept of activity

The main goal of activity theory is to enable the study of genuine individual and social developmental processes while maintaining their contextuality in as rich a form as possible. The fundamental idea is that activities in which participate are the basic units of development and human life and thus form the base of the study of all contextuality.

Activities - an individual can participate in several at the same time - have following properties:

- an activity has *a motive* and activities can be separated according to their motives. A motive tells us why an activity exists.
- an activity is a *collective phenomenon.*
- an activity has a *subject*, who understands the motive of the activity. This subject can be individual or collective. Not all participants necessarily understand the motive of the activity in which they are participating (alienation).
- an activity exists in *a material environment* and *transforms it.*
- an activity is a *historically developing* phenomenon.
- *contradictions* are the force behind the development of an activity.
- an activity is realized through conscious and purposeful *actions* by participants.

Y. Engeström has made in (Engeström 1987) an attempt to establish a simple model of the concept of activity as a culturally mediated phenomenon . This means that the relationships between the main compo-

---

starting point for the development of a Marxist theory of psychology after the Russian revolution.

[4] Although it has never dominated any dicipline (it is still being debated even in Soviet psychology), it has gained users around the world. It has also been found to be useful and flexible for the analysis of other disciplines, such as education, the social sciences, cultural research, anthropology, work science etc. The activity theory "school" is just become organized, the First International Congress on Activity Theory having been held in Berlin 1986, and the Second Congress in Lahti, Finland in 1990. From 1988 on there has also existed a journal - the Multidisciplinary Newsletter for Activity Theory.

nents of the concept cannot be defined arbitrarily but are situation-bound and have developed historically in the course of the cultural process.

Cultural mediation is dealt with in Engeström's model by introducing a third, mediating member into every main binary relationship which carries with it the cultural heritage of the situation. Thus the central relationship - that between the subject and object of activity - is mediated by a tool into which the historical development of the relationship between subject and object thus far is condensed. This simple structure is not adequate to fulfill the needs of a consideration of the systemic relations between an individual and his environment, however, and thus Engeström proceeds to the structure in the Fig. 1.

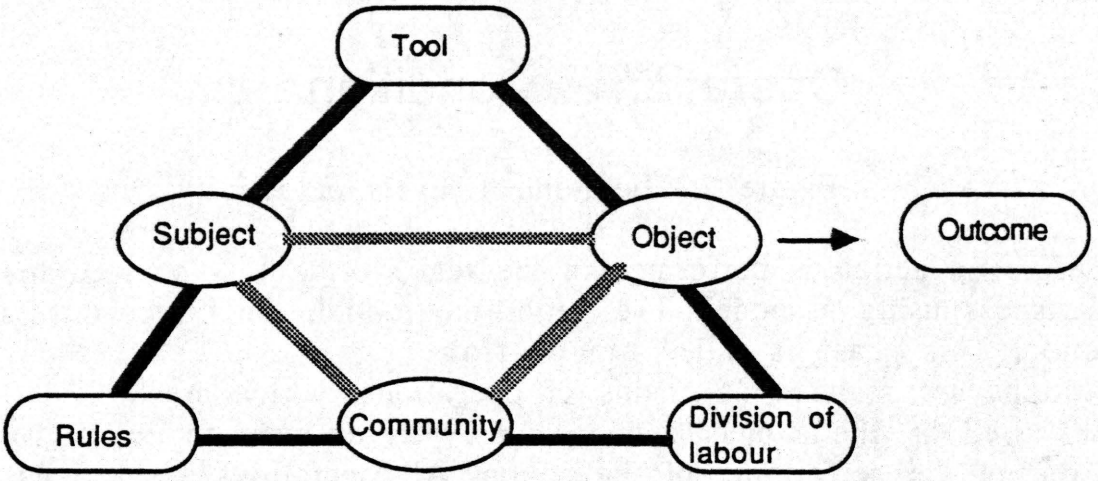


Figure 1: Structure of an activity

This systemic model - which according to Engeström is the simplest possible - contains three mutual relationships between subject, object and community. The relationship between subject and object is mediated by tools, that between subject and community is mediated by rules and that between object and community is mediated by the division of labour. We can interpret rules as being a form of institutionalized communication. On the other hand, organization is an institutionalized form of some division of labour. Each of the mediating terms is historically formed and open to further development.

Activities can be distinguished by their **motives**, which tell us why they exist. Human life consists in practice of innumerable parallel or intertwined activities, in which the individual participates.

Activity-action-operation hierarchy

Besides this overall structure activities also have a hierarchical inner structure. Activities consist of **actions** or chains of actions, which in turn consist of **operations**, resulting the structure in Fig. 2.

When we participate in an activity, what we are actually doing is just performing conscious actions which have a material object and an immediate, defined **goal**. (The term material must be understood as in

Marxist philosophy.) The actions cannot really be understood, however, without a frame of reference created by the corresponding activity. An activity may be realized using different actions, depending on the situation, while one and the same action can belong to different activities, in which case the different motives of activities will cause the action to have a different **personal sense** for the subject in the context of each activity.

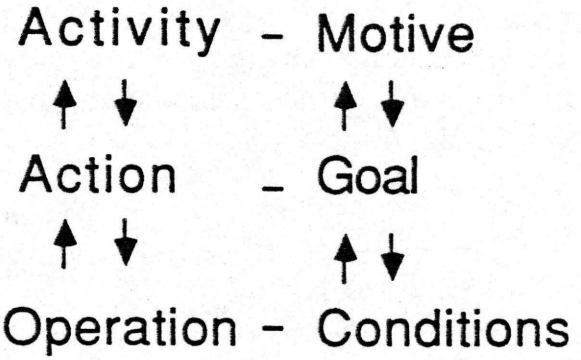


Figure 2: Hierarchical levels of activity.

Before an action is performed in the real world, it is planned in the consciousness using a model. The better the model, the more successful the action. This phase is called **orientation.**

Actions are realized as chains of operations, which are well-defined routines used by the subject subconsciously as answers to **conditions** faced during the performing of the action. All operations are actions at the beginning, but when the corresponding model is good enough and the action has been practiced long enough, the orientation phase will fade and the action will be collapsed into an operation, which is much more fluent. At the same time a new action is created which will have broader scope and will contain the recently formed new operation as a subpart of itself. On the other hand, when conditions change, an operation can again "unfold" and return to the level of conscious action (so that it is not a conditioned reflex).

The border between action and activity is also blurred, and movements are possible into both directions. This flexibility of the basic concepts is one of the reasons why they are gaining popularity within different disciplines. On the other hand, it also means that it is in fact impossible to make a general classification of what is an activity, what is an action etc. because the definition is totally dependent on what are the subject, object etc. in a particular real situation.

External and internal activity

One very important feature is that activities have a double nature: there is both an external and an internal "side" in every activity. The subject and the object of an activity are in a reciprocal relationship with each other: on the one hand, subject is transforming the object - on the other hand, the properties of the object penetrate into subject and trans-

form him or her. The external activity is according to Activity Theory the primary one, and the internal activity, created from the external via **internalization**, reflects it. In internalization "...processes are subjected to a specific transformation: they are generalized, verbalized, abbreviated, and, most importantly, become susceptible of further development which exceeds the possibility of external activity" (Leontjev 1974, p. 18).

A person's internal activity assimilates the experience of humanity in the form in which it manifests itself in the corresponding external activity. "It means that a person's mental processes acquire a structure necessarily linked to sociohistorically formed means and modes, which are transmitted to him by other people through teamwork and social intercourse." (Leontjev 1974, p. 19).

External and internal activity are thus a similar in structure, and all the elements of external activity will have their counterparts in internal activity and cultural-historical meditation will penetrate into the relations between them. The similarity means also that the internal activity consists of conscious mental actions, which again consist of subconscious mental operations.

3. Discussion: the relationship between Activity Theory and HCI

In order to discuss about interface design we must define the larger framework, the relationship between activities and computer applications. Firstly, it can be stated that computer applications are condensed, objectified external and internal operations (the term operation being used here in the Activity Theory sense). Thus it can also be said that they replace some parts of a work activity and become a part of it. "In a man-machine system human activity is distributed both among people and among technical facilities." (Zinchenko & Munipov 1989, p. 198). The view of computer applications as replacements for operations is too unidimensional, however, and is not sufficient to analyze the whole situation. Information systems in fact penetrate the whole activity, both the external and the internal side of it, and every mediating member of the activity structure. The classification in the figure 3 helps to explain the tight intertwining relationships involved.

This multiple penetration means, in fact, that research and development for information systems cannot be separated from research and development for whole work activities. When a new system is implemented, all the aspects of the corresponding work activity will change - either intentionally or accidentally.

Role of user \ Area of support	External activity	Internal activity	Relations within community
Passive	Automation of well-defined operations	Control: support for pre-determined actions through pre-determined internal models	Coordination: by establishing a division of labour
Active	Tool: support transformation of material objects	Learning: support for the building of a new model for the situation	Communication: enabling the formation of social intercourse

Figure 3: Types of work support in information technology

Then, what Activity Theory (AT for short) can give to HCI research and development? There is not yet any definite answer to that question. Thus far AT is applied within the HCI field only in a handful of publications[5], and much more work has still to be done. In any case, the view of AT differs radically from the standard cognitive science view of communication-between-two-information-processors. In the light of AT there is no "user" just "interacting" with the computer application, but *an active subject* using the *application as a tool* in *manipulating some object* so that the result of the manipulation is meaningful in the *context of some activity.*

When studying interfaces, the main emphasis lies obviously in the domain of actions and operations: "The idea is that ideally, in the use situation, the user should direct operations, but not actions towards the computer artifact, in order to proceed smoothly, and not perform actions which require constant attention. In particular we should look for conditions for operations towards the artifact and for operations towards subjects or objects through the artifact. (...) What we want is software and hardware supporting the human operation of the computer application in a specific type of use activity, constituting some of the material conditions for triggering specific operations in a specific use situation" (Bannon & Bødker 1989, pp. 21-22).

Thus the interface design according to AT is, in fact, the design of the work flow in some particular work process. The crucial importance does not lie in the analysis of isolated tasks (which in HCI studies seem to be very narrow and technologically determined, see Grudin 1989a), but the *synthesis* of the task "threads"; the connections between isolated tasks for to form a meaningful whole and the role of action-operation dynamics in

[5] Bødker (1987), Bødker (1989) and Bannon & Bødker (1989) may be the best examples.

it, the connections between different task threads within the whole individual work context, and, finally, the connections of the task threads of one individual work with those others co-working in the same work activity.

How able is AT in answering the problems defined earlier in the paper? Because of the lack of applications, the evaluation has to based only on the conceptual potential of the theory.

1) The problem of contextuality: AT has evidently a strong potential. The concept of activity forms a natural context for actions to be supported by applications. The concept of orientation enables the studying of situational mental models, and the concept of interiorization enables the studying of the formation of mental models in a shared practice.

2) The problem of lacking focus: AT resolves the focusing problem by subsuming the interface design under work activity and process design. On the other hand, it demands that the relative weight of interface design within the whole information system design must increase - because it is crucial for the success in supporting the work, it must be one of the foremost activities in the design process. It cannot be neglected as in many recent information system design methods (Iivari 1989) or stuffed into some restricted corner, but be present during the whole development process, like in "integrated design" by Gould (1988).

3) The problem of multidisciplinarity: again, AT apparently has a strong potential. Activities as wholes cannot be exhaustively studied by any individual discipline. In fact, one arrives at the conclusion that several disciplines should actually have the same context of the research object, namely the context formed by activity. Although they are focused on different aspects of activity, all the other context-forming parts must be also taken into account in order to preserve the validity of the research. This common context of the object of study could dramatically enhance the possibilities for different disciplines to discuss with and benefit each other. Of course, this is a research programme more than an actual situation, but steps have already been taken in that direction.

4) The gap between theory and practice: AT gives conceptual tools for studying also practical situations and not only artificial laboratory settings. It is another question how easy those tools are to use in practice, of course. Anyhow, there exists an encouraging example: it has been possible to develope a successful methodology, based on AT, for developing whole work activities (Toikka et al. 1985, Engeström & Engeström 1986).

4. Conclusion

Activity Theory seems to be possessing some such features, which have been on the wish list of the recent debate concerning the background theory for HCI. The dominating paradigm - cognitive science - has been criticized because of the lack of many of those features. Especially, Activity Theory seems to give some hope in mastering the complexity of real work and use situations. Although the firm practical evidence is still

mostly lacking, it is at least not worse than other alternatives - certainly worth of a closer study.

References

Bannon, L. J. (draft) *A Pilgrims Progress: From Cognitive Science to Cooperative Design*, Dept. of Computer Science, Aarhus University.

Bannon, L. J. & Bødker, S. (1989) *Beyond the Interface: Encountering Artifacts in Use*. Dept. of Computer Science, Aarhus University, Report DAIMI PB-288. (Will also be published in Carroll (forthcoming)).

Bellotti, V. (1988) Implications of Current Design Practice for the Use of HCI Techniques. In D.M. Jones and R. Winder (Eds.) *People and Computers IV*, Cambridge Univ. Press, Cambridge, pp. 13-34.

Budde, R. & Floyd, C. & Keil-Slawik, R. & Züllighoven, H. (Eds.) (in press) *Software Development and Reality Construction*. Springer Verlag, Berlin.

Bødker, S. (1987) *Through the Interface - a Human Activity Approach to User Interface Design*. Univ. Aarhus DAIMI PB-224. (To be also published in book form by Lawrence Erlbaum Associates).

Bødker, S. (1989) A Human Activity Approach to User Interfaces. *Human Computer Interaction*, vol. 4 no. 3 pp. 171-195.

Carroll, J. M. (Ed.) (1987) *Interfacing Thought, Cognitive Aspects of Human-Computer Interaction*, MIT Press, Cambridge, MA.

Carroll, J. M. (Ed.) (forthcoming) *Designing Interaction: Psychology at the human-computer interface*. Cambridge Univ. Press, New York (1991).

Carroll, J. M., Kellogg, W.A. and Rosson, M. B. (1990) *The Task-Artifact Cycle*. Research report RC 15731 (#69538) 4/6/90, IBM Research Division, New York. (Will also be published in Carroll (forthcoming)).

Ehn, P. (1988) *Work-oriented design of computer artifacts*. Arbetslivscentrum/Almqist & Wiksell International, Falköping, Sweden.

Engeström, Y., (1987). *Learning by expanding*. Orienta-konsultit, Helsinki.

Engeström, Y. & Engeström, R.(1986): Developmental work research: the approach and application in cleaning work. *Nordisk Pedagogik*, vol. 6, n:o 1 ss. 2-15.

Fischer, G. (1989) Human-Computer Interaction Software: Lessons Learned, Challenges Ahead. *IEEE Software*, January 1989, pp. 44-52.

Gould, J. D. (1988) How to Design Usable Systems. In M. Helander (Ed.) *Handbook of Human-Computer Interaction*. Elsevier Science Publ. / North-Holland

Grudin, J. (1989a) The Case Against User Interface Consistency. *Communications of the ACM*, vol. 32 no. 10 (October 1989), pp. 1164-1173.

Grudin, J. (1989b) *The Computer Reaches Out*. Århus Univ. DAIMI PB-299.

Hildebrand-Nilshon, M. & Rückriem, G. (eds.) (1988-89) *Proceedings of the 1st International Congress on Activity theory, vol. 1-4,2*. System Druck, Berlin.

Iivari, J. (1989) Levels of Abstraction as a Conceptual Framework for an Information System. In E. D. Falkenberg and P. Lindgreen (Eds.)

Information System Concepts: An In-depth Analysis. North-Holland, Amsterdam, pp. 323-352.

Klein, H. & Hirschheim, R. (1987) Social Change and the Future of Information Systems Development. In Boland, R. & Hirschheim, R. (eds.) *Critical Issues in Information Systems Research.* John Wiley & Sons, Chichester, pp. 275-305.

Kuutti, K. (forthcoming) The Activity Theory and Its Applications to Information Systems Research and Development. To appear in the proceedings of the IFIP 8.2 workshop "The Information Systems Arena of the 1990s - ISRA-90" (December 1990, Copenhagen)

Leont'ev, A. N. (1974) The Problem of Activity in Psychology. In *Soviet Psychology* vol. 13 no. 2 pp. 4-33.

Leontjev, A. N. (1978) *Activity, Consciousness, Personality.* Prentice-Hall, Englewood Cliffs, New Jersey.

Mumford, E. & Hirschheim, R. & Fitzgerald, G & Wood-Harper, A.T. (Eds.) (1985) *Research Methods in Information Systems.* North-Holland, Amsterdam.

Raeithel, A. (in press) An activity-theoretical foundation for design. In R. Budde, C. Floyd, R. Keil-Slawik & H. Züllighoven (Eds.) *Software Development and Reality Construction.* Springer Verlag, Berlin.

Reason, P. & Rowan, J. (eds.) (1981) *Human Inquiry. A Sourcebook of New Paradigm Research.* John Wiley & Sons, Chichester.

Suchman, L. (1985) *Plans and situated actions.* Cambridge Univ. Press. Cambridge.

Thomas, J. C. & Kellogg, W. A. (1989) Minimizing Ecological Gaps in Interface Design. *IEEE Software* January 1989, pp. 78-86.

Toikka, K. & Engeström, Y & Norros, L.(1985) Entwickelnde Arbeitsforschung. Theoretische und methodologisce Elemente. *Forum Kritische Psychologie* 15, ss. 5-41.

Wertsch, J. V. (Ed.) (1981) *The concept of activity in Soviet psychology.* Sharpe, Armonk, New York.

Wertsch, J.V. (1985) *Vygotsky and The Social Formation of Mind.* Harward Univ. Press, Cambridge, Mass.

Winograd, T. & Flores, F. (1986) *Understanding computers and cognition: A new foundation for design.* Ablex Publ., New Jersey.

Witheside, J. & Wixon, D. (1987) Discussion: Improving Human-Computer Interaction - a Quest for Cognitive Science. In J. Carroll (Ed.), *Interfacing Thought, Cognitive Aspects of Human-Computer Interaction*, MIT Press, Cambridge, MA, pp. 353-365.

Zinchenko, V. & Munipov, V. (1989) *Fundamentals of Ergonomics.* Progress Publ. Moscow.

Motivation, Research Management and a Conception for Structured Integration of Human Factors with System Development Methods: An Illustration Using the Jackson System Development Method.

Lim, K. Y., Long, J. B. and Silcock, N.

Ergonomics Unit,
University College London,
26 Bedford Way, London
WC1H 0AP, U. K.

Abstract

This paper presents the motivation underlying the contention that methodological integration between Human Factors (HF) and Software Engineering (SE) design, would ensure appropriate consideration of HF design concerns throughout the system development life-cycle. It is argued that research on integration could lead to a generalised conception of HF and SE relationships within an overall design life-cycle. This conception would facilitate appropriate recruitment of HF inputs such as guidelines and tools. It is also suggested that initial research towards the achievement of this long term objective would be facilitated by a class of SE methods, commonly referred to as Structured Analysis and Design Methods. Research into methodological integration has been initiated for one of these methods, namely the Jackson System Development (JSD) method. This paper reports a description of what is entailed by the research undertaking, its management strategies and the output derived to date, i.e. the current conception of HF integration with the JSD method.

1. Motivation for Integrating Structured SE and HF Methods

In recent years, various approaches have emerged for ensuring appropriate incorporation of Human Factors (HF) contributions during system design. These take the form of: methods, tools, and guidelines (and to a lesser extent, standards[1]). Although each of these approaches contributes towards highlighting HF concerns during design, inadequacies are widely recognised. For example, methods by themselves cannot ensure good design (Newman, 1989); the recruitment of guidelines lacks systematicity and may require tailoring to specific application contexts (Smith, 1986); tools[2] may offer only narrow functional coverage (Hartson and Hix, 1989), etc. Current remedial support predominantly takes the form of expert advice. Although the latter support may have had some success, questions remain as to when and how experts should be recruited relative to the system design life-cycle. Exhortations are frequent that advice should be sought early rather than late in the design cycle. This advancement of HF inputs may alleviate the 'too-little-too-late' problem. However, the form and scope of HF concerns with respect to the design cycle remains firmly in the realm of craft (see Long and Dowell,

[1] There are doubts about the effectiveness and validity of existing standards for ensuring adequate HF input into software, e.g. Smith (1986); Potosnak (1988); Rogers and Pegden (1977); Hirsch (1984), etc.
[2] See later paragraphs for an expansion of this assertion.

1989; Smith, 1986). Furthermore, judgements on what constitutes appropriate early HF input remain subjective. To ensure relevant and effective HF contributions, HF design concerns must be appropriately contextualised to the system design life-cycle. Advancing this argument, it follows that appropriate HF involvement throughout the design life-cycle should be ensured. Only then might HF contribution to system design achieve its full potential. To attain this goal, the following basic requirements must be met:

(a) adequate identification of HF concerns and support required throughout system design. This support may be classified into three broad categories, namely the provision of procedural support ('principled' conduct of design, e.g. iterative and structured design, clear documentation of design rationale, modular design, demarcated roles for analytic and empirical evaluation, etc.); declarative support (e.g. expressions of HF knowledge); and communicative support (between designers and users, and between designers). Means for expressing these types of design support may then be configured. For instance, methods and tools predominantly provide procedural support for design (e.g. methods have generally been regarded to be useful in enforcing order and systematic management of design complexities; and tools should be contextualised to methods (Hartson and Hix, 1989)), while guidelines tend to be declarative in nature. Communicative support relates to the notation used for design documentation, and/or tools for modelling/ exemplifying design prototypes for further analysis or feedback elicitation.
(b) appropriate organisation and timing of HF concerns and support with respect to the system design life-cycle. To achieve these aims, a high level conception of inter-disciplinary design relationships needs to be defined/identified between SE and HF. This conception would capture inter-relationships such as information inter-dependencies corresponding to overlapping design concerns, relative progression of design scope and process for developing the design artefact, and representation of intermediate design deliverables. Identification of these inter-dependencies is necessary to characterise appropriately the nature of design collaboration that is required between HF and SE.

The final objective is to derive an overall system development life-cycle, that explicitly describes inter-weaving HF and SE design concerns/scope, process and notation; and hence to define HF and SE inter-dependencies throughout design. To derive such a conception, SE may serve as a model[3], as system development has traditionally been viewed and organised as a life-cycle. Furthermore, this view has also been expressed in the form of methods, a class of which direct system design as a series of well-defined stages, that are structured in terms of their scope, process, notation. This particular class of methods[4], commonly referred to as Structured Analysis and Design Methods (SADMs), has recently received wide attention in SE (e.g. Bell et al, 1988; Holman; 1988). Although the methodological structures of SADMs are well defined and developed, it cannot be assumed that they would be adequate by themselves for organising HF concerns as well as those of SE. As Mumford (1988) aptly points out, methods are not neutral — they embody particular visions of the world and incorporate the values of their creators. As HF design needs are not explicitly represented and allowed for within the system design agenda of SADMs, prevailing constraints (e.g. time frame of deliverables) would influence the effective application of HF methods. These problems have been reported by Meister (1984). Consequently, it is insufficient to develop structured schemes for HF input (e.g. procurement schedules for HF advice from experts) which fail to consider HF and SE inter-dependencies throughout the system design life-cycle. Thus, to ensure efficient management of HF involvement *throughout* system development, HF and SE design concerns should not only be structured but integrated as well. In this regard, methodological structures of SADMs could usefully serve as frameworks for integration. Research could then be undertaken to structure HF design methods (in the current form and extensions thereof) over the system design life-cycle, and subsequently to be integrated with those of SE. Methodological integration would thus be defined over the dimensions of

[3] HF methods cover the system design life-cycle less widely than SE methods, and tend to focus on later design stages. This emphasis derives historically from the late recruitment of HF.
[4] Other methods generally fail to satisfy the conditions of well defined scope, process and notation, that are explicitly structured over a (more or less) complete coverage of the system design life-cycle.

design scope, process and notation.

Such an integrated conception or framework of SE and HF within an overall design life-cycle, would allow the following inferences to be made:

(a) that HF tools should not be developed in a 'vacuum' but integrated with respect to the system design life-cycle. This notion is analogous to SE concepts of "total systems solution" or "life-cycle support" (Hewett and Durham, 1987). Requirements of the design task must be identified if appropriate functional coverage is to be realised in the tool. Functional support may be classified into three broad categories, namely support for: specification (e.g. documentation and editing, consistency checks with respect to notational constructs, management support for design deliverables such as planning and progress tracking aids, etc.); evaluation (e.g. simulators, protoypers, animators, etc.); and implementation (e.g. compilers, linkers, implementation environments, application generators, etc.). Such Computer-Aided Software Engineering (CASE) tools are beginning to emerge on the basis of SADMs (Hewett and Durham, 1987). Integrated Project Support Environments (IPSE), with the potential for supporting collaborative work, are also being pursued via SADMs (Bott, 1988; England, 1988). Without an expression of methodological integration of HF and SE embedded within these tools, future HF contributions would not be effective.
(b) that HF guidelines and methods should be developed and configured with respect to the design life-cycle, e.g. guidelines should be structured and tailored into design rules and algorithms (see Smith, 1986); methods should sufficiently cover the design scope, and their products should be contextualised to evolving design needs and be appropriately represented.
(c) that the recruitment of currently available HF methods (e.g. Meister, 1984; Kloster and Tischer, 1984), guidelines (e.g. STARTS Guide, 1987) and tools (e.g. England, 1988) should be contextualised to the design life-cycle. In other words, appropriate roles for these approaches for design inputs could be identified (as in (a)) such that their complementarity is maximised; i.e. the weaknesses of one approach are compensated by the other, and strengths are jointly exploited.
(d) that specific inadequacies of existing design supports should be easier to identify. Measures may then be taken to rectify inadequacies, e.g. bespoke tools may be developed to meet functional supports which are lacking; specific research areas may be identified and undertaken to develop better declarative support; the adequacy of existing methods may be assessed with respect to their coverage of concerns throughout the design life-cycle, etc.
(e) that the problem of facilitating HF inputs into design should not be confined to ensuring that SE designers are aware of HF design concerns. Mutual understanding of stage-wise design needs should be encouraged and accommodated within the design agenda. In this regard, the uptake of HF throughout the system development life-cycle may be facilitated by the provision of familiar reference points to SE designers. This provision could be achieved by capitalising on existing design structures of SADMs.

In summary, methodological integration of HF and SE design is pivotal for ensuring efficient management of HF involvement *throughout* the system development life-cycle. To this end, research into the derivation of such an integrated conception may be undertaken sequentially as follows:

(a) for the short to medium term: the demonstration of HF and SE integration for particular SADMs, e.g. Structured System Analysis and Design Method (SSADM), Jackson System Development (JSD) method, etc.
(b) for the long term: the abstraction and synthesis of integrated conceptions associated with particular SADMs, to derive a generalised conception of design inter-relationships between SE and HF over the entire system development life-cycle.

The preceding assertions and research schedule set the background against which the current research into HF

integration with the JSD method was undertaken[5] (i.e. this integration of HF with JSD falls within the short to medium term research category as described by (a) above). In the following sections, the management of this research as well as a conception of HF integration with JSD (subsequently referred to as JSD*), will be described.

2. Managing the Research of HF Integration with the JSD Method[6]

Before considering aspects of research management, it is necessary to set out the requirements of the undertaking with respect to the JSD method. In this regard, the points made earlier about methodological integration are now tailored as follows:

(a) an extension of the JSD design scope to include HF concerns (not necessarily as a simple addition, since JSD and HF concerns may overlap e.g. requirements analysis).
(b) inter-weaving of JSD and HF design processes, that is the location and timing of HF design stages within JSD. The objective is to construct an integrated process from disparate HF and JSD design streams, such that the timing of stage-wise design concerns (such as scope and inter-dependencies) would be appropriately accommodated. This process may require the configuration of stage-wise HF design processes.
(c) extension of the JSD notation for the description of HF inputs (not necessarily as an addition, since JSD notations may be sufficient for expressing HF inputs (see Carver and Cameron, 1987; Carver, 1988; Sutcliffe, 1988)).
(d) retention of JSD as essentially unchanged within JSD*, with the exception of design inter-dependency requirements (this 'modified' SE component will be referred to as JSD*(SE)).

In other words, the achievement of these goals primarily requires the development and integration of a structured HF analysis and design stream (subsequently referred to as JSD*(HF)) with JSD. The function of such a stream would be the derivation of HF specifications expressed as a set of tightly constrained stage-wise descriptions (ranging from intermediate to final/desired design products). The development of this stream would require the following sequential processes, all performed with due considerations of the characteristics of JSD[7]:

(a) definition of the stage-wise design scope and requisite range of products associated with the HF specification stream
(b) proceduralisation of design transformation processes between design products
(c) representation of stage-wise design products using appropriate notations of JSD and extensions thereof
(d) definition of contact points between structured JSD*(HF) and JSD*(SE) design streams, that are considered obligatory for ensuring convergence of the scope of design address; i.e. the identification and setting down of inter-dependencies between the two design streams

Based on these considerations, an overall plan for managing the research and development of JSD* can be formulated. Figure 1 shows the schematic representation of such a plan.

Insert Figure 1 about here.

[5] The choice of JSD is due to two principal reasons, namely it is a well accepted SADM, and is the method favoured by the sponsors of the current research. Research efforts in a similar direction have also been made elsewhere for other SADMs, e.g. Damodaran et al, 1988, for SSADM; and Hakiel, 1990, for Yourdan.
[6] Although this is described with respect to JSD, generalisation to other SADMs is possible. Some of these have been suggested in Lim et al (1990).
[7] It cannot be over-emphasised that at the outset of the research, an adequate understanding of the stage-wise concerns of JSD (specifically its stage-wise scope, process and notation) is a pre-requisite.

The plan is essentially divided into management modules concerning method specification and its implementation in case-study tests. These modules in turn comprise the following stages:

(a) an assessment of the design coverage by JSD, focussing on the identification of its potential requirements for HF support. The resulting enhancement generates a preliminary or high level conception of JSD*.
(b) a more detailed assessment of the methodological characteristics of JSD, i.e. its stage-wise scope/ product, process and notation. This assessment is required to develop further the preliminary conception of JSD*, i.e. for subsequent configuration of HF support that could be integrated in a stage-wise fashion with JSD. Potential areas for HF support are then identified and further developed. For example, it may be observed that an explicit user interface (UI) design process is not included in the scope of JSD. Thus, integration would entail the development of an HF method for UI design within the design framework of JSD. As shown in Figure 1, development of the stage-wise HF method is undertaken in three parts: (i) notation development; (ii) scope/product and process development; and (iii) modular specification of HF support activities following the development of an acceptably 'stable' conception of the HF method. Parts (i) and (ii) could proceed concurrently by focussing initially on the identification and characterisation of descriptive aspects (as opposed to derivative aspects) of the HF method[8]. Progressively detailed conceptions of the HF method are iteratively developed and tested using case-studies. When a well-formed low level conception of the JSD*(HF) method is derived, integration of the JSD*(SE) and JSD*(HF) streams is then explicitly formalised; i.e. inter-dependencies between the two design streams are specified. This specification is achieved by identifying intersecting stage-wise design concerns, and hence information inter-dependencies between the two design streams. Obligatory contact points[9] are thus set down. This constitutes the first conception of JSD*. Progressively improved conceptions of JSD* are then derived via further case-study tests and refinements.
(c) the low level conception of JSD* is then ready for testing with respect to its capability to address derivative aspects of a design method. In addition, questions of scale are also addressed. Subsequent enhancements would then lead to the derivation of an acceptable expression of the final JSD* method[10].

It was suggested earlier that in support of the research plan, strategies should be adopted to facilitate a systematic management of the complex issues involved in this undertaking. The function set for these strategies is three-fold: (i) to arbitrate between alternative ways of conducting the research; (ii) to support the derivation of methodological conceptions; and (iii) to support the testing of these conceptions. Three strategies (SS*, SS1 and SS2), together, address the concerns of the research plan outlined in the preceding section. Figure 2 describes how these strategies may be exploited to progress the research plan shown in Figure 1.

Insert Figure 2 about here.

The super-ordinate strategy (SS*) comprises the iteration of specification followed by case-study

[8] These aspects will be expanded later when strategic support is described.
[9] Note that informal exchanges between streams at other design stages are not precluded. As these exchanges tend to be situation-specific, they are not appropriate to be set out for JSD*. The requirement that a method be general (with respect to design subject matter and circumstances), dictates that only obligatory contact points or exchanges be made explicit.
[10] It should be noted that the time frame of the current project limits the research scope to the development and demonstration of an integrated methodology. It does not include the validation of its efficacy in ensuring improved design. Nevertheless, since methodological integration is built upon previous HF research and design practice, it is reasonable to expect that better designs would result from its application.

implementation or test. This strategy is similar to that used in hypothesis testing; i.e. an iterative process comprising a literature review is followed by a specification (hypothesis) of a stage-wise JSD*(HF) conception, which is then implemented (tested) using case-studies (an experimental rig). The conception of JSD*(HF) is upgraded at the end of each cycle, and the procedure is repeated. An instantiation of this strategy is shown in the top part of Figure 2. This may be illustrated by the following scenario. Two research schemes may be applied for the development of the JSD*(HF) method. First, to tailor independently JSD*(HF) within JSD and to validate its efficacy in the field. Second, to survey existing HF literature concerning extant design methods and practice and maximise their recruitment to JSD*(HF). Tailoring (comprising modifications and extensions) to specific aspects of JSD is then undertaken[11]. The latter scheme would be expected to improve interactive computer design and to reduce the need for subsequent iterative testing[12]. In other words, this scheme allows more rapid convergence onto a satisfactory conception of JSD*(HF), since it builds upon past knowledge[13] (i.e. substantiated conceptions provide more assurance of conceptual validity). Furthermore, early derivation of a JSD*(HF) conception allows more case-study test cycles to be undertaken within the restricted project time frame. Thus, the second scheme is preferred (see Figure 2).

A second research strategy (sub-ordinate to SS*) is backwards before forwards engineering (SS1 - see Figure 2). This is instantiated during both method specification and test/implementation as follows:

(a) separate treatment of descriptive aspects (addressed by backwards engineering) and derivative aspects (addressed by forwards engineering) of a design method. Research would initially focus on the identification and characterisation of descriptive aspects of the JSD*(HF) method. On attaining an acceptable conception of JSD*(HF), tests and further development on its capability for supporting derivative aspects of a design method would then be undertaken. Such a scheme in turn allows separate/parallel development of JSD*(HF) as indicated earlier; i.e. development of notational aspects as one part, and scope and process aspects as the other. This is instantiated by the initial templating[14] of the JSD*(HF) method with respect to its stage-wise design scope and process (see Figure 2). Preliminary templates of JSD*(HF) are derived by porting and tailoring appropriate aspects of current HF design methods and formalisations of its practice. At the same time, notational development and testing could also proceed in parallel, namely: (i) testing the power of existing JSD notations for describing stage-wise design products corresponding to preliminary conceptions or templates of JSD*(HF); (ii) enhancing existing JSD notations by porting other notational schemes associated with the original source(s) of the JSD*(HF) template (either whole or parts thereof); and (iii) independent formulation of proposed notational extensions to JSD and their testing for JSD*(HF) as in (i).
(b) the application of backwards engineering during templating is supported by the choice of small and simple (followed by larger and more complex) case-studies with well defined design start- and end-points (i.e. SS2. See later for expansion on this inter-play of SS1 with strategy SS2). Stage-wise design processes and scope can then be worked out to extend preliminary conceptions of JSD*(HF). Support for the application of forwards engineering is provided by strategy SS2 (see later expansion of SS2).

---

[11] Some flexibility should also be allowed within the method to avoid stifling designer creativity; i.e. excessive premature fine tuning should be avoided.

[12] Note that this expectation should be tempered by other factors, e.g. the composition and abilities of the design team members. A method alone, cannot guarantee superior design. Thus, an assessment of the efficacy of integration would require field validation regardless of which research scheme is pursued.

[13] For the same reason, it should also support positive transfer of training with respect to the application of the JSD*(HF) method by designers.

[14] The separation of descriptive from derivative aspects of a design method is supported by the manner in which the template scheme is configured. The separation is in turn suggested by the separate treatment of design scope and process, and notation. In the present case, separate treatment is made possible by the original separation of descriptive and derivative development of a design method. In other words, the scheme is recursive.

Questions to be answered during the application of SS1 are, for example, whether the stage-wise conception of JSD*(HF) is sufficiently well specified:

(a) to support advancement of the design in terms of descriptions of its design processes (and intermediate design products), such that they are adequately defined to facilitate the generation of HF (e.g. UI) specifications (Figure 2, left side).
(b) to support intermediate design descriptions (such that inter-designer communication is facilitated), and final product descriptions (such that UI specifications may be implemented unambiguously) (Figure 2, right side).

Answers to such questions are then interpreted, extrapolated and incorporated as re-specifications and/or further extensions of the JSD*(HF) conception.

In was indicated earlier that iteration in SS1 proceeds in accordance with a second sub-ordinate strategy which prescribes the testing of the method using case-studies of increasing complexity and size (SS2). More needs to be said about this inter-play between SS1 and SS2. SS2 is invoked to impose a range of case-study test demands on conceptions of JSD*(HF) developed using strategy SS1. For instance, during early JSD*(HF) development and forward engineering tests on JSD*, small and simple case-studies would be prescribed as initial test-beds by SS2. In other words, SS2 is a strategy for controlling complexity, and hence facilitates its management during research.

To summarise, the scope of SS1 includes the development and testing of proposed conceptions of JSD*(HF). SS2 then supports SS1 by initially recommending simple case-study systems with well (i.e. easily) defined design products, corresponding to the start and end points of the design process. The overall coherence and completeness of the proposed stage-wise scope, process and notation associated with the particular conception of JSD*(HF), are thus developed and assessed. The super-ordinate strategy, SS*, then prescribes further iterative loops of the case-study/case-studies as necessary. Inter-dependencies between HF and SE design streams are then made explicit to generate the first conception of the JSD* method. This conception is then tested under a forwards engineering scenario (SS1). During this test of the JSD* method, roles similar to those used for JSD*(HF) development are assigned for strategies SS2 and SS*.

This completes the account of the management aspect of JSD* research and development. A brief overview of the product of the research, i.e. the current JSD* conception, will now be presented. Desired aspects of methodological integration of HF and SE, as outlined in Section 1, will also be highlighted.

3. A Brief Overview of the Current Conception of JSD*

Figure 3 shows the current JSD* conception.

Insert Figure 3 about here.

It can be seen that JSD* comprises parallel JSD*(SE) and JSD*(HF) design streams. These design streams are structured into stage-wise processes, and design inter-dependencies occur across several of their stages. It should be emphasized at this juncture that Figure 3 should not be taken to imply that JSD*(HF) is more complex than JSD*(SE). The intention is to show the former in greater detail as it constitutes the focus of this section. General characteristics of JSD* may now be described:

(a) all stages of JSD* are defined with respect to their scope (product), process (procedure) and notation

(representation). These aspects have been specified at a level considered to be adequate for an HF designer with a working knowledge of JSD. Although JSD* does not prescribe a particular composition for the design team, it may be assumed to reflect the two design streams; i.e. SE and HF designers working together, with the possibility of additional support from domain experts.
(b) the methodological structure of JSD* facilitates the application of accepted design principles such as: the delaying of design commitments (Thimbleby, 1987) as instantiated by the setting of conceptual design before detailed design; the conduct of early evaluation, either analytically by the designer or through feedback (made possible by the explication of stage-wise design products), or empirically through prototyping; the application of iterative design or successive design refinement throughout the design cycle; the adoption of modular design options (within each of the two JSD* design streams) following the laying of foundations for conceptual design at the CTM stage.
(c) JSD*(HF) may be considered to comprise three phases, namely: elicitation phase (generation of preliminary design information within the Extant Systems(HF)[15] System Analysis (ESSA) stage; the analysis and synthesis phase (development of conceptual design spanning the ESSA and Composite Task Model (CTM) stages); and the design specification phase (detailed design undertaken from the System(HF) Task Model (STM) and User Task Model (UTM) stages onwards). The scope of these stages may be characterised briefly as follows:

(i) the ESSA stage generates background information for design, e.g. capturing details of the current system(HF). This includes information on user needs and problems, extant tasks, design features and rationale, etc. In addition, extant design characteristics are assessed for their potential for porting to the system(HF) to be designed (or the target system(HF)), as well as the consequential effects thereof, e.g. possible transfer of learning by end users from the extant system(HF) to the target system(HF) (both positive and negative effects).
(ii) the Generalised Task Model (GTM) stage processes information gathered in the ESSA stage into predominantly device independent descriptions. These descriptions facilitate analytic evaluation for subsequent mapping between extant design features and those desired of the target system(HF), i.e. the objective is to facilitate design porting. In addition, new and/or salient characteristics desired of the target system(HF) are exposed.
(iii) the Statement of User Needs (SUN) stage draws together conclusions from the analysis of extant designs and summarises them with respect to the target user. The purpose of the SUN stage is to augment the initial statement of requirements for the target system(HF), and sensitise it to target user needs and requirements, e.g. the identification of current problems experienced by the end-user, a summary of the rationale, requirements and constraints underlying extant design features, the expression of design criteria (e.g. for performance) and domain semantics, etc. The resulting description may be regarded as the HF view of the statement of requirements for the target system(HF).
(iv) the CTM stage generates the conceptual foundation for the target system(HF) via the synthesis and/or arbitration of GTM descriptions; i.e. it provides the basis on which subsequent decisions on the allocation of function are made. Lower level details may also be documented, e.g. design rationale, demarcation of on- and off-line tasks destined for the target system(HF), etc.
(v) the System and User Task Model (STM and UTM) stage is concerned with the decomposition of on-line and off-line components of the CTM product, such that high level functions may be identified. The STM product describes the user-computer (UC) interactive task, while the UTM product describes off-line tasks, i.e tasks which are unsupported by the computer. Desirable extant design features are also carried forward from the ESSA stage and incorporated on the basis of the SUN.

[15] System(HF) refers to the person-machine system which performs tasks (and achieving work in the process) in a particular environment. In the context of this definition, HF is primarily concerned with end-user behaviours with respect to device, i.e. UI behaviours, while SE is concerned with device behaviours, i.e. UI behaviours with respect to user behaviours.

(vi) the Interaction Task Model (ITM) and Interface Model (IM) stages generate descriptions which specify the manner in which inputs are to be effected, and the behaviour of UI objects (in response to user inputs and changes in attribute states of domain entities for either or both system/representation and real world entities) respectively. In other words, the IM product describes the appearance and behaviour of UI objects in terms of actions and associated state changes, while the ITM product is a 'keystroke' level description derived mainly from the STM description. The latter is either expressed in terms of object/action primitives of the adopted UI environment and basic keystrokes of the designated hardware, or at a level that is easily understood among design team members and end-users as appropriate.
(vii) the Display Design (DD) stage extends the descriptions of the IM and ITM products to determine the composition and layout of display screens; and the content and location of error, feedback and help messages. Screen actuations are also contextualised to the progress of the UC dialogue and on-line task, e.g. points at which particular system messages are initiated, and where particular sets of functions are presented to the user in accordance with the support requirements intended for the on-line task.

As the stage-wise design scope of JSD*(HF) is thus explicitly defined (so providing guidance on the granularity and timing of design concerns), the method facilitates the recruitment of HF guidelines and tools at appropriate stages of the design cycle. Its stage-wise notations also constrain the expression of HF design inputs. Communication between designers with respect to both intermediate and final design products, is thus supported.
(d) as part fulfilment of the constraints of this project (see Lim et al, 1990), the JSD*(SE) stream corresponds to an essentially unchanged JSD method; i.e. with the exception of design inter-dependency requirements. It suffices to say for this paper that within JSD*, two obligatory contact points are identified during design specification (points of contact following the design specification phase are not discussed in this paper). These contact points correspond to the two sequential specification stages of JSD*(SE), namely JSD*(SE) Modelling and Functions stages. The nature of the contact points are as follows:

(i) as the JSD*(SE) Modelling stage is concerned with capturing the subject matter of the target system, some overlap will occur with the design information captured by the JSD*(HF) stream. In general, JSD*(HF) products derived at the CTM and SUN stages are discussed with SE designers. The discussion comprises the agreement and exchange of information such as attributes of objects and actions, user needs and problems, lists of events, user task semantics and intended user task support. A Functions List intended for the system being designed, is then identified and carried forward by both design streams.
(ii) the JSD*(SE) Functions stage is concerned with the specification of information flows between JSD Function and Model processes, as well as with the external world. As such, there is some intersection of design concerns with the STM stage of the JSD*(HF) stream. Design information of common interest may include the functional sequencing embedded in STM descriptions corresponding to the JSD*(HF) design stream; and input and output streams, descriptions of JSD Function and Model processes, and the input subsystem (with respect to error handling), from the JSD*(SE) design stream. In general, at these points in the design life-cycle (i.e. JSD*(HF) STM stage and JSD*(SE) Function stage), both streams of the JSD* method are concerned with functional definition and conceptual design extrapolation. To ensure efficient convergence of the design scope close contact should therefore be maintained, i.e. this inter-dependency between the two design streams is an essential one. Failure to maintain close contact may result in the need for additional design iterations; e.g. changes would have to be reflected in an updated Functions List, and further JSD Function processes have to be specified in the Functions Stage of the JSD*(SE) design stream.

This completes the brief overview of JSD*. For a more detailed stage-wise account of JSD*, the reader is referred to Walsh et al (1988), Silcock et al (1990) and Lim et al (in press).

4. Conclusion

It has been argued that the integration of HF and SE methods within an overall design life-cycle, potentially provides a framework, on the basis of which, other means of ensuring HF inputs may be appropriately located and facilitated. In addition, HF design needs may then be accommodated explicitly within the design agenda. To achieve this objective, research into methodological integration should initially focus on particular HF and SE methods. On the basis of this research, a generalised conception of HF and SE design integration may then be attempted in the longer term. In this regard, it was further suggested that initial research into HF integration would be facilitated by SADMs as they are methodologically well defined and structured. Integration may thus be achieved by configuring and structuring an HF design stream within the chosen SADM. Methodological integration of SE and HF design is explicitly expressed in terms of the stage-wise scope, process and notation; as well as required design inter-dependencies between SE and HF streams, over the entire system development life-cycle. An instantiation of such a research undertaking was given with reference to the JSD method. This example includes an account of the management strategies adopted for the research. In general, JSD* development may be divided into two sequential phases, namely a phase for the development and testing of JSD*(HF), followed by a similar phase for JSD* (this includes formalised inter-dependencies between JSD*(SE) and JSD*(HF) streams). As JSD*(SE) is well-defined and is essentially left unchanged, work on the first phase dominated the present research. This research into JSD*(HF) was undertaken in separate modules, namely the scope and process module; and the notation module. The three strategies adopted for the research were found to be instrumental in making tractable, the complex nature of the undertaking. During the research, conceptions of HF and SE integration were developed and subjected to case-study tests. These indicated the current JSD* conception to be promising. This JSD* conception was described briefly to provide an indication of the research undertaking. Work on the research is ongoing, and further reports will be made in the near future.

Acknowledgements

The research associated with this paper is being carried out for the Procurement Executive, Ministry of Defence (Royal Armaments Research and Development Establishment) under Contract No. 2047/130 (RARDE). Views expressed in the paper are those of the authors and should not necessarily be attributed to the Ministry of Defence. Special acknowledgements are also due to M. Carver (LBMS) and to individual members of staff at CA2 (RARDE).

References

Bell, D., Green, E., Murray, F., Owen, J. and Pain, N.,1988, Human-Centred Design: A Methodology for Action. In Information Technology for Organisational Systems, eds. Bullinger et al, (Elsevier Science Publishers, North Holland) pp. 230-234.

Bott, M.F., 1988, Systems Analysis and Design Methods and Integrated Development Environments: The State of the Art and the Future. In Contemporary Ergonomics 1988, 'Ergonomics Giving Quality to Life', Proceedings of the Ergonomics Society 1988 Annual Conference, ed. E. D. Megaw (London: Taylor Francis), pp. 164-170.

Carver, M. K., 1988, Practical Experience of Specifying the Human-Computer Interface Using JSD. In Contemporary Ergonomics 1988, 'Ergonomics Giving Quality to Life', Proceedings of the Ergonomics Society 1988 Annual Conference, ed. E. D. Megaw (London: Taylor Francis), pp. 177-182.

Carver, M. K. and Cameron, J., 1987, The Jackson System Development Method: A Framework for the Specification of the Human-Computer Interface. Unpublished Internal Report of Michael Jackson Systems Limited.

Damodaran, L., Ip, K. and Beck, M., 1988, Integrating Human Factors Principles into Structured Design Methodology. In Proceedings of Eurinfo '88, First European Conference on Information Technology for

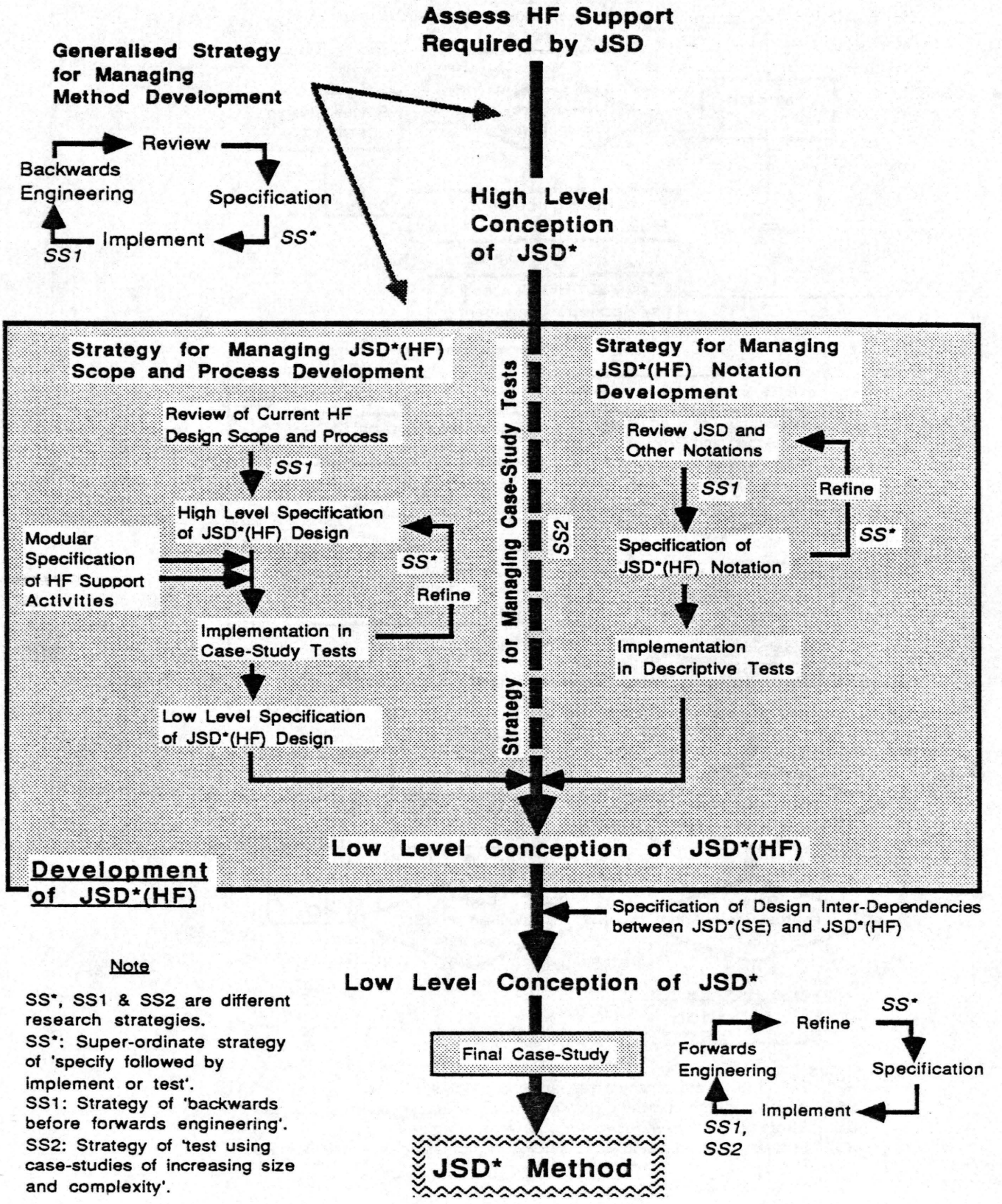
Organisational Systems, Athens, May 1988, (Elsevier Science Publishers, North Holland), pp. 235-241.
England, D., 1988, User Interface Design and Construction in ECLIPSE. Handout for the BCS HCI Specialist Group meeting, 31 October 1988, London.
Hakiel, S. R. and Blyth, R. C., 1990, Keeping the Human in Context. In Contemporary Ergonomics 1990, 'Ergonomics Setting Standards for the '90s', Proceedings of the Ergonomics Society's 1990 Annual Conference, ed. E. Lovesey, Leeds, 3-6 April, (Taylor and Francis) pp. 123-128.
Hartson, H. R. and Hix, D., 1989, Towards Empirically Derived Methodologies and Tools for Human-Computer Interface Development. In Int. J. Man-Machine Studies, **31**, (Academic Press) pp. 477-494.
Hewett, J. and Durham, T., 1987, Computer-Aided Software Engineering: Commercial Strategies, (Ovum Ltd.).
Hirsch, R. S., 1984, VDTs and the Human Factors Community: Tipping the Iceberg. In Human Factors Society Bulletin, **27**, No. 6, pp. 1-3.
Holman, R., 1988, Integrating Human Factors into the Systematic Development Process. In Contemporary Ergonomics 1988, 'Ergonomics Giving Quality to Life', Proceedings of the Ergonomics Society 1988 Annual Conference, ed. E. D. Megaw (London: Taylor Francis), pp. 183-188.
Kloster, G. V. and Tischer, K., 1984, Man-Machine Interface Design Process. In Proceedings of the First IFIP Conference on Human-Computer Interaction — INTERACT'84, ed. B. Shackel, London, 4-7 September, (Elsevier Science Publishers, North Holland), pp. 889-894.
Lim, K. Y., Long, J. B. and Silcock, N., 1990, Requirements, Research and Strategy for Integrating Human Factors with Structured Analysis and Design Methods: The Case of the Jackson System Development Method, In Contemporary Ergonomics 1990, 'Ergonomics Setting Standards for the '90s', Proceedings of the Ergonomics Society's 1990 Annual Conference, ed. E. Lovesey, Leeds, 3-6 April, (Taylor and Francis), pp. 38-43.
Lim, K. Y., Long, J. B. and Silcock, N., in press, Integrating Human Factors with Structured Analysis and Design Methods: An Enhanced Conception of the Extended Jackson System Development Method, In Proceedings of the Third IFIP Conference on HCI (INTERACT '90), ed. D. Diaper, Cambridge, 27-31 August, (Cambridge University Press).
Long, J. B. and Dowell, J., 1989, Conceptions of the Discipline of HCI: Craft, Applied Science, and Engineering. In People and Computers V, Proceedings of the Fifth Conference of the BCS HCI Specialist Group, eds. A. Sutcliffe and L. Macaulay, 5-8 September, Nottingham (Cambridge University Press), pp. 9-34.
Meister, D., 1984, A Catalogue of Ergonomic Design Methods. In Proceedings of the International Conference on Occupational Ergonomics, pp. 17-25.
Mumford, E., 1988, Designing Systems: Influences, Processes and Methods. In Proceedings of the Int. Conference on the Joint Design of Technology, Organisation and People Growth, 12- 14 October, Venice, Instituto RSO Milan.
Newman, W., 1989, User Interface Design Practice. Tutorial Notes for Fifth Conference of the BCS HCI Specialist Group, 5-8 September, Nottingham, pp. 6.
Potosnak, K., 1988, What's Wrong with Standard User Interfaces ? In IEEE Software, **5**, September 1988, pp. 91-92.
Rogers, J. G. and Pegden, C. D., 1977, Formatting and Organisation of a Human Engineering Standard. In Human Factors, **19**, pp. 55-61.
Silcock, N., Lim, K. Y. and Long, J. B., 1990, Requirements and Suggestions for a Structured Analysis and Design (Human Factors) Method to Support the Integration of Human Factors with System Development,In Contemporary Ergonomics 1990, 'Ergonomics Setting Standards for the '90s', Proceedings of the Ergonomics Society's 1990 Annual Conference, ed. E. Lovesey, Leeds, 3-6 April, (Taylor and Francis), pp. 425-430.
Smith, S. L., 1986, Standards versus Guidelines for Designing User Interface Software. In Behaviour and Information Technology, 1986, **5**, No 1, pp. 47-61.
STARTS Guide, 1987, Software Tools for Application to large Real-Time Systems. Second Edition, 1989, National Computer Centre Ltd, Manchester.

Sutcliffe, A., 1988, Some Experiences in Integrating Specification of Human Computer Interaction within a Structured System Development Method. In Proceedings of the Fourth Conference of the BCS HCI SIG Conference, eds. D. M. Jones and R. Winder, Cambridge, (Cambridge University Press), pp. 145-160.

Thimbleby, H., 1987, Delaying Commitment. Internal Working Paper, University of York.

Walsh, P., Lim, K. Y., Long, J. B. and Carver, M. K., 1988, Integrating Human Factors with System Development. In State of the Art Report 15:8, "Designing End-User Interfaces", eds. N. Heaton and M. Sinclair, (Pergamon Infotech Limited), pp. 111-120.

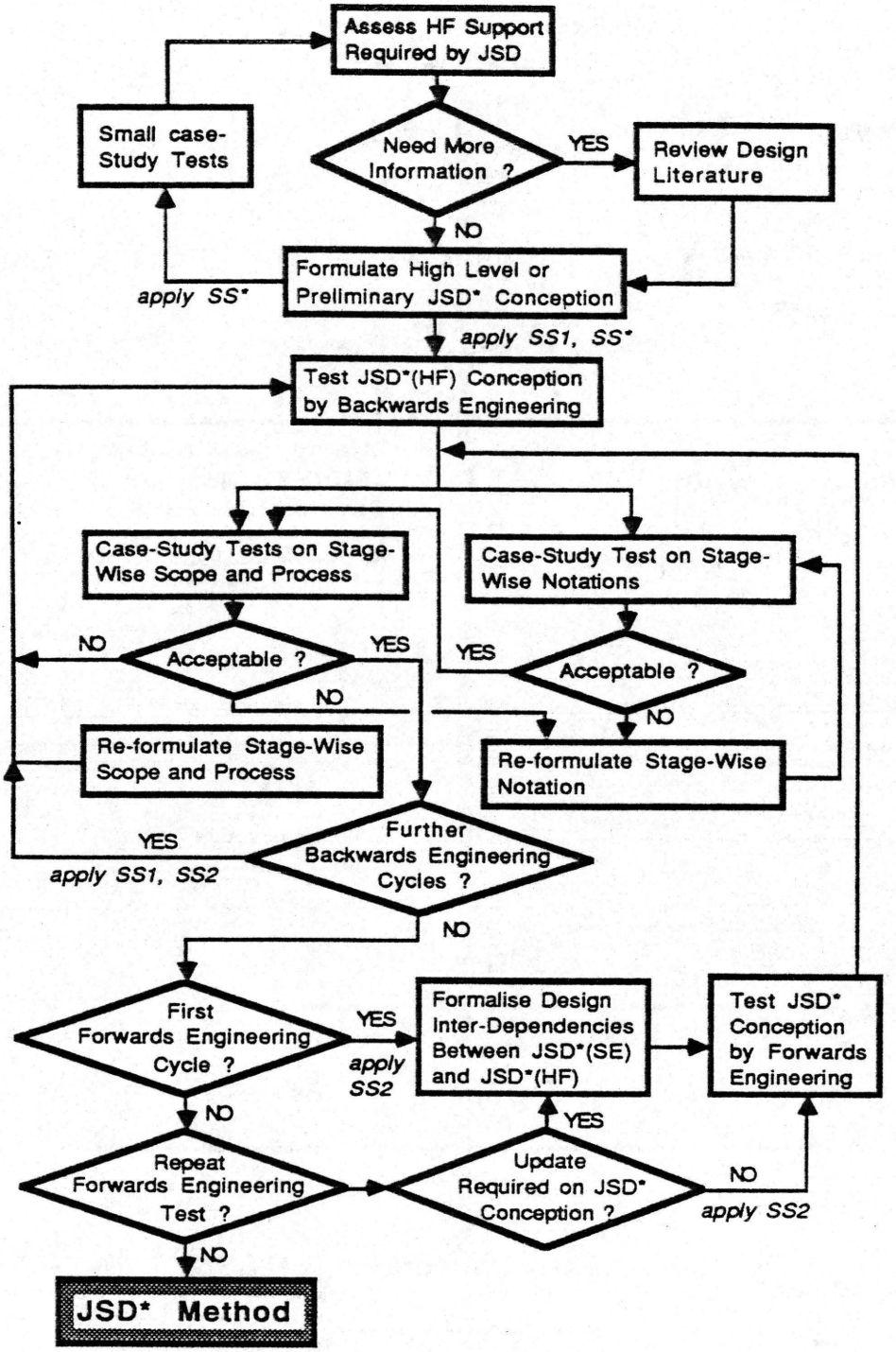
Figure 1: A Schematic Description of the Plan Adopted for Managing JSD* Research



Note

SS*, SS1 & SS2 are different research strategies.
SS*: Super-ordinate strategy of 'specify followed by implement or test'.
SS1: Strategy of 'backwards before forwards engineering'.
SS2: Strategy of 'test using case-studies of increasing size and complexity'.

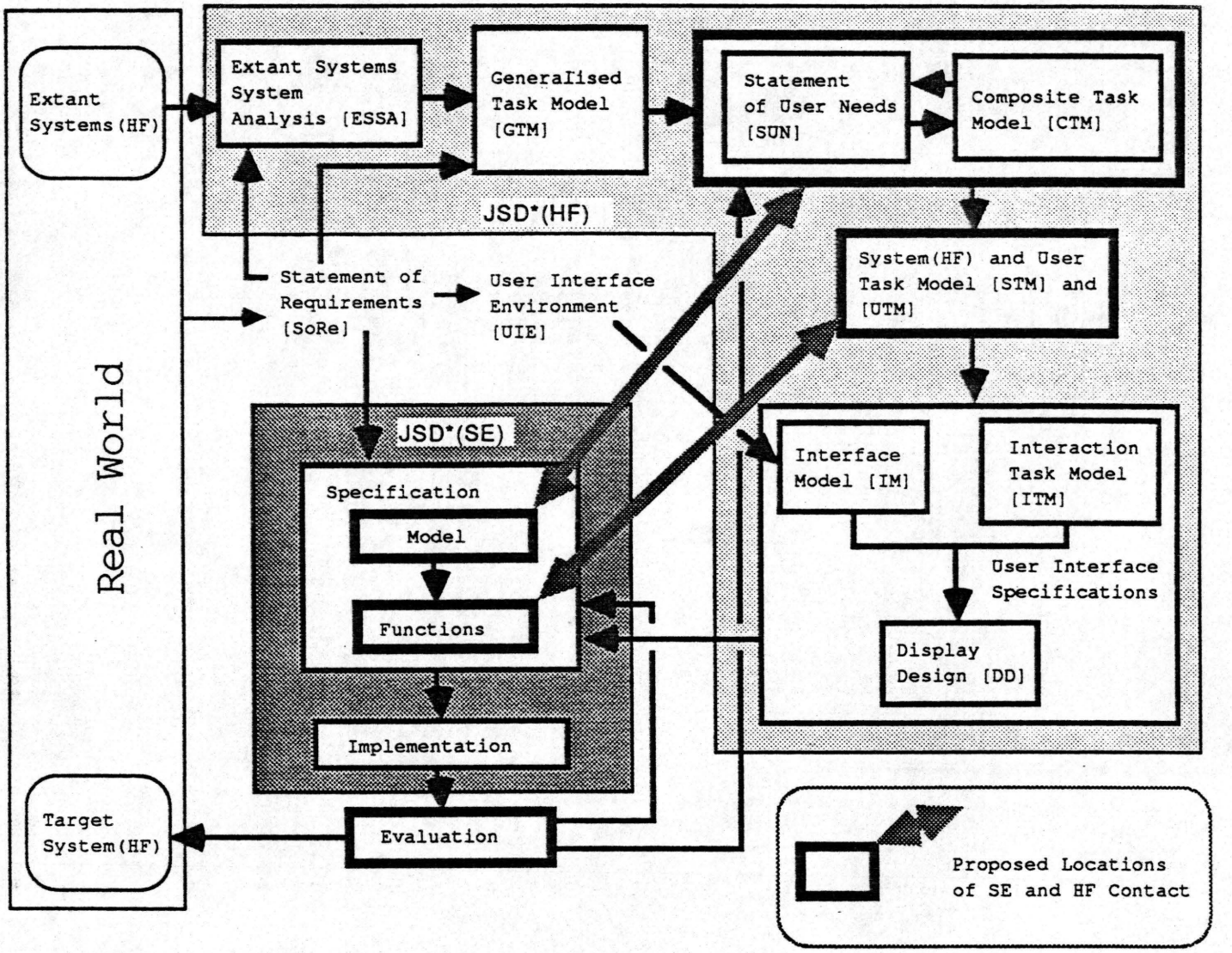
Figure 2: A Schematic Description of the Configuration of Management Strategies Associated with the Research Plan as Shown in Figure 1.



Note

SS*, SS1 & SS2 are different research strategies.
SS*: Super-ordinate strategy of 'specify followed by implement or test'.
SS1: Strategy of 'backwards before forwards engineering'.
SS2: Strategy of 'test using case-studies of increasing size and complexity'.

Figure 3: A Schematic Representation of a Simplified Conception of JSD*



SOFTWARE USABILITY: A USER-CENTRED PROCESS OF EVALUATION

Oronzo Parlangeli

OSN-Olivetti System & Network
Via Jervis 77, 10015, Ivrea, (TO)
Italy

Summary. The aim of the methodology described in this article is to evaluate the usability of software products within a laboratory environment. This methodology has been developed as part of an ESPRIT Project (Hufit, n. 385), which concluded in November 1989.

The first commitment for this process of evaluation was to satisfy the requirements of a commercial environment. That is to say that results coming from the process of evaluation must be comprehensible, useful and expressive of the end user reaction to the product. Consequently both objective and subjective variables have been considered. A procedure to convert data coming from objective variables to the same scale of the subjective ones is described. Such a procedure is based upon criteria provided by the subjects during the interaction with the product.

As a result of the process of evaluation, a single value expressing the degree of usability of the product will be calculated. Data analysis will provide indications about the product improvement.

A case study about the evaluation of the usability of a Word-Processor has been conducted. Data coming from this study demonstrates the feasibility and the adequacy of the proposed user-centred evaluation process.

1. INTRODUCTION

In the past, supplier companies of information technology tended to offer more competitive products by adding functionality not provided by competitors. But some researches (see for example Eason 1988) have shown that increasing functionality of software products often leads to a poor usage of many of the facilities available.

In addition, as a consequence of the great diffusion of information technology, end users can no longer be considered as specialists (Eason and Harker 1988; Nickerson 1986).

The awareness of changes in the market and in the users requirements, has forced to a deeper consideration of end users characteristics. It has also implied a growing attention to the degree of usability of a particular product, starting

from the early stages of its development (Eason and Harker 1988).

Evaluation of software usability is essentially a process that parallels each phase of the product life cycle (PLC). It is necessary to establish, at a particular stage, the difference between what has been planned and what has been developed.

Many suppliers are now trying to satisfy this need of evaluating products under development within a laboratory environment using methodologies.

Any such methodology must not only supply valid and useful data (Stahl 1987; Thomas and Kellogg 1989), but must also satisfy specific requirements of the commercial environment in which it is used.

The criteria that guided the development of the methodology described here were:

1 - the possibility of obtaining a single value that allows the expression of the degree of usability of a product;

2 - the possibility of analysing data that allow the provision of precise indications for product improvement;

3 - a real consideration of the end users characteristics.

These three requirements guided us to elaborate a methodology in which:

- experimental tasks are categorised according to a principle relevant for the study that must be conducted (point 2);
- both objective and subjective variables are considered (point 3);
- all measurements coming from different variables are converted to a uniform scale using criteria supplied by subjects during their phase of interaction with the product (points 1 and 2);
- after the conversion of all the measurements to a uniform scale, data is averaged progressively in order to obtain a single value (point 1);
- a top-down analysis of data, in order to find out parts of the product needing improvements, is adopted (point 1).

A brief description of the steps characterizing each procedural phase follows (see fig. 1).

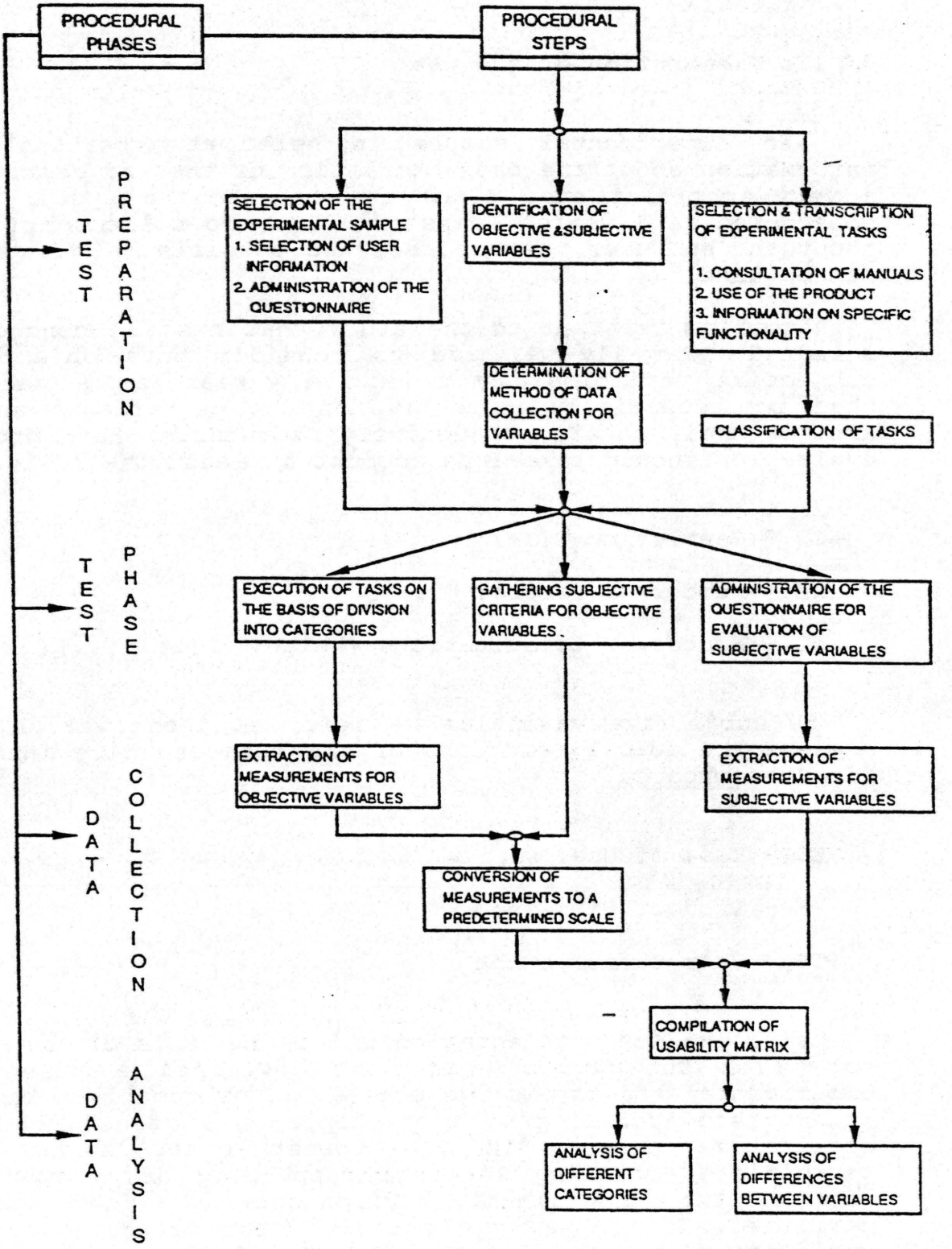


Figure 1: Procedural Phases and Steps

2. PROCEDURAL PHASES

2.1 Preparation of the test

An experimental sample is selected according to the information about the characteristics of the end user. This is a very important step since the man-machine system must be considered as a whole (Meinster 1986). To collect information about the end user, as precisely as possible, a questionnaire can be used.

Variables relevant to the study that must be conducted are selected. Normally we have to consider both objective and subjective variables. By objective variables we mean those that are essentially expressive of the performance and which are, by their nature, objectively measurable. Each process of evaluation should take into account at least the following:

CT - Completion Time;

NE - Number of Errors;

NC - Number of Consultations (helps, manuals,..).

By subjective variables we refer to those variables that express the actual reactions of the subject while interacting with the product:

EU - Ease of Use;

PF - Product Fitness;

US - User Satisfaction.

Using product documentation and by analysis of the activity for which the product has been developed a selection of experimental tasks is generated. Consequently, tasks are categorised according to a principle of a structural or even qualitative nature. In the former case, tasks can be categorised according to their complexity or by taking into account the programming environment in which they are performed. If instead the principle for categorisation is a qualitative one, then we can distinguish for example, between tasks requiring memory for a sequence of commands as opposed to tasks that are directed by messages from software.

This categorisation creates a general structure in which the guiding principle is easily analysed, emphasising intercategorical differences. Moreover information relevant to single tasks will not be lost if a deeper analysis is necessary.

2.2 Test phase

The test in the laboratory is video recorded. It is structured in such a way as to obtain:

- measurements for the objective variables (CT, NE, NC);
- measurements for the subjective variables (EU, PF, US);
- subjective criteria to convert the measurement of the objective variables onto a common scale with the subjective variables.

During this phase the subject interacts with the product performing a series of tasks grouped into categories. Each category contains a variable number of tasks. The subject usually begins by performing the first task in the first category. The test is concluded when the last task of the last category is completed.

Data coming from subjective variables are uniform. That is to say that they are expressed on the same scale. Measurements of these three variables are in obtained using a questionnaire based on a Lickert-type scale.

The questionnaire is administrated to the experimental subjects after they have completed each category of tasks. Before compiling the questionnaire it is clarified with the subjects that their replies given must refer only to the category of tasks just completed. Thus the data which refers to subjective variables are more summary than those referring to the objective ones.

In order to obtain measurements that are all of the same nature (satisfaction of the point 1, expressed in the introduction) a process of conversion is necessary for data coming from objective variables. This process of conversion can be carried out using some criteria. These criteria, according to a user-centred approach are provided by the experimental subjects during their interaction with the product.

The experimenter asks the subjects, essentially assisting and without causing them to feel under examination, opinions about their own performance. These judgements form the basis to be used to convert the objective measurements on a scale equivalent to that one used for subjective variables. For example, each subject's judgements of very-poor/very-good performance can be considered. By averaging measurements for the variables corresponding to these opinions, subjective values to be associated to the minimum/maximum of the scale can be calculated. As a consequence of these operations, it will be possible to determine the amplitude of the intervals composing such a scale, thus allowing the satisfaction of two crucial requirements:

- measurements coming from different variables will be expressed on the same scale. Consequently it will be possible to calculate a general average;

- each value obtained after conversion will reflect the experience, the tolerance, the subjective reaction to the product.

The experimenter needs to obtain a fare amount of judgments for each objective variable and from each subject involved in the test.

2.3 Data collection

As a result of the user's interaction with the product in the laboratory, the experimenter has to analyse many hours of video recording. Firstly he must determine operational criteria for the extraction of the measurements for the three objective variables. Then the actual analysis of the videotape must be conducted.

To retrieve data referring to the three subjective variables is a fairly simple task. Information about these variables has been collected by administrating a questionnaire after each category of tasks. The experimenter now only has to transpose the subjects opinions onto points on a scale that he has chosen previously.

At this point, criteria coming from subjects during the interaction with the product can be used to convert measurements for objective variables onto the same scale used for subjective ones. This operation produces uniform data for all the variables considered.

2.4 Data analysis

It must be remembered that, after the data collection phase, the experimenter has the following data available for each subject:

- measurements of the three objective variables (CT, NE, NC) converted according to the criteria provided by the subjects on a predetermined scale, for each task performed by each subject;

- measurements of the three subjective variables (EU, PF, US) on the same scale used for objective variables, for each tasks category.

After calculating the averages for tasks in the same category, data can be put into a matrix. Such a matrix contains, at the most detailed level of information, data coming from each category of tasks and referring to both objective and subjective variables. Progressively averaging this data, yields a unique value, expressing the degree of usability of the product evaluated.

A top-down analysis (from the most general result to the most detailed) can be conducted, to identify specific parts of the product needing improvements. The operation can be conducted from at least two different perspectives.

First, intercategorical differences can be analysed. Categories of tasks giving scores above the normal may identify ways in which to increase the usability of the product where scores are below the normal. Data interpretation should be facilitated by analysing in which way the principle that guided tasks categorisation, has affected intercategorical differences.

Second, the overall averages of the variables can be compared. Differences can emerge showing if developed functionality permit an efficient performance (objective variables) and/or if the the subjects have favourable reactions using the product (subjective variables) . Also in this case, analysis can be continued down to the categories level to obtain more detailed information and to find out where to improve the product.

3. A CASE STUDY: USABILITY EVALUATION OF A WORD-PROCESSOR

A study as been conducted to validate the methodology developed. The facilities and equipment used were those ones normally available in a Human Factors Laboratory: a testing room with three cameras (two remotely controlled) and microphones, and a control room with an observation screen, television monitors and video recording equipment.

3.1 Method

A word-processor application, running on a PC, was chosen for the study. This application was already commercialised and tested from a technical functionality point of view. On the whole, its characteristics are similar to those of other packages developed for the same purpose.

Subjects were selected, according to the characteristics of the product's target population, by means of a User Profile Questionnaire. The sample group was composed of 6 secretaries

aged between 30-40 years old. Of six, five already possessed a familiarity with IT products. The sixth had no worthy previous experience.

Before performing the test, subjects were trained in the use of the product. The course was the same normally offered with the product and lasted 5 days.

By use of the product and analysis of its documentation, a selection of tasks was created. These activities produced a first sample of tasks that was then reduced by taking into account the effective use of the product by the end users.

The structural principle guiding tasks categorisation "Complexity". Tasks were grouped in categories according to the number of commands required for their execution. In "category 1" were tasks requiring one command, "category 2" two commands, and so on up to "category 7".

Before starting the test each subject was informed exactly of what would be required of them, the purpose of the experiment and what procedures would be used. The experiment was repeated twice with each subject in four day cycle. The setup for each test was the same with a test duration of 60-90 minutes.

Tasks were performed one after the other. When all the tasks of one category were completed a questionnaire was administrated to collect evaluation of the three subjective variables (EU, PF, US) for that category.

The subject's opinions of their own performance were collected by the experimenter. Three anchor points were considered sufficient: minimum, medium and maximum. These were used to convert data coming from the three objective variables (CT, NE, NC) to the same scale adopted for the subjective variables.

Following to data collection, measurements of the objective variables were converted. A seven point scale was chosen as for subjective variables. Data was then summarised in a matrix and a single value expressing the degree of usability of the product was calculated.

3.2 Main results

The second cycle of tests with the same subjects has given results only slightly better than the first session. The data from the two session has thus been considered together and the evaluations given report an average.

Figs. 2 and 3 show evaluations obtained for each variable in each task category on a seven point scale.

By calculating the average of these values, a weight of 5.05 expressing the degree of product usability is established.

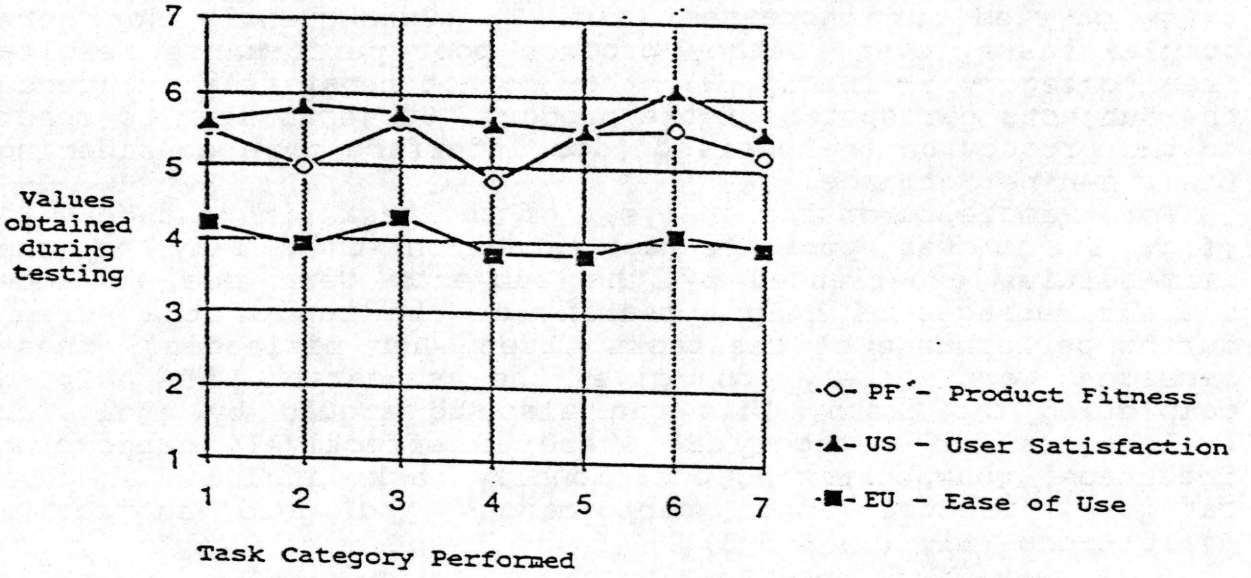


Figure 2: Overall Averages of the Three Subjective Variables

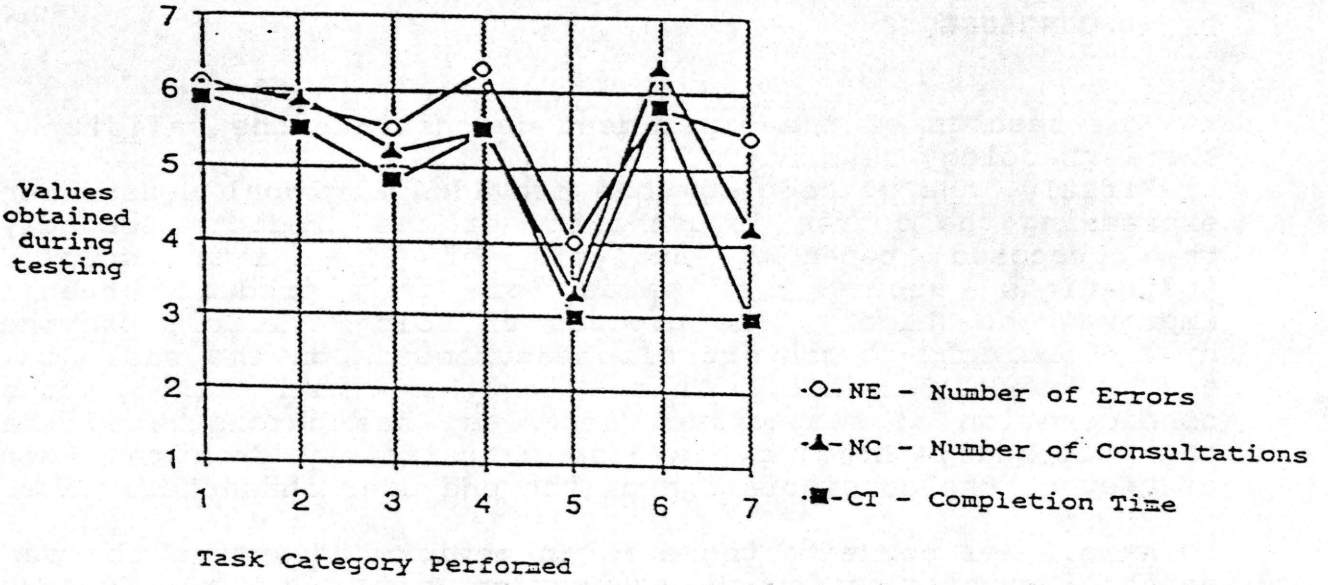


Figure 3: Overall Averages of the Three Objective Variables

Subjects do not show a decrease of their product acceptability scores when the structural complexity of the tasks carried out increases (fig. 2). Consequently for more complex tasks, even if they produce poor performance results (see "category 5" in fig. 3), they do not negatively influence the subjects perception of the product. Subjects seem to react to the product on the basis of its interface than considering their own performance.

For example, from the analysis of the tasks in "category 5" (fig. 3) it was possible to establish that many of the difficulties experienced by the subjects were due to some unclear messages of help appearing at the top of the screen during performance of the tasks. Even when misleading, these messages were likely to give the sensation of helping completing the tasks. This can also be argued by analysing evaluations for "category 6" where an effectively supporting interface, characterising occasionally tasks included in this category, produces good performances and good subjective evaluations (figs. 2 and 3).

Analysis of subjective variables scores indicates EU (Ease of Use) as the variable with the lowest scores. Going deeper, it is possible to find out that this is a consequence of the negative evaluation of the training course. Subjects consider five days as a too short period to learn all the functionality of the product. This probably means that the end user characteristics were not really considered elaborating the course program.

4. CONCLUSIONS

The results of the experiment demonstrate the validity of the methodology described.

Firstly, the methodology has provided a global measurement expressing the degree of usability of the product. Secondly, the proposed top-down analysis of the data supplied indications about the parts of the product needing improvement. Thirdly, the process of using criteria provided by the subjects to convert all measurements on the same scale has proved feasible. This, together with the actual consideration of subjective variables, has demonstrated that the proposed methodology is also satisfactory from the point of view of the consideration of the end user characteristics.

As a final comment: there is an obvious advantage to use a methodology for software evaluation early in the product development process. In this case, the product to evaluate is often a prototype in which we can incorporate improvements coming as a result from evaluation. Consequently, evaluation together with a prototyping approach, can lead to a clear, efficient an quicker communication with developers, saving time, resources and money (Benimoff and Whitten 1989).

5. ACKNOWLEDGEMENTS

This work is based upon my contribution to the Final Report of an ESPRIT Project (Hufit n° 385). The author would like to thank all the members of the Hufit team inside Olivetti, the System Test Department and the partners involved in the Project.

6. REFERENCES

Benimoff, N. I. and Whitten, W. B. 1989. Human Factor Approaches to Prototyping and Evaluating User Interface. AT&T Technical Journal, Vol. 68, n. 5, 44-55

Eason, K. D. 1988. Patterns of usage of a flexible information system. In The Application of Information Technology, edited S. D. P. Harker and K. D. Eason. London: Taylor and Francis

Eason, K. D. and Harker, S. 1988. The Supplier's Role in the Design of Product for Organisations. The Computer Journal, 5, 426-430

Meinster, D. 1986. Human Factors Testing and Evaluation. Amsterdam: Elsevier North Holland

Nickerson, R. S. 1986. Using Computers. Cambridge, Massachusetts: MIT Press

Stahl, B. 1987. Testing for Usability can head off disaster. ComputerWorld, December, 7, 83-92

Thomas, J. C. and Kellogg, W. A. 1989. Minimizing ecological Gaps in Interface Design. IEEE Software, January, 78-86

Is graphical notation really superior to text, or just different? Some claims by logic designers about graphics in notation

M. Petre

Institute of Educational Technology, The Open University

T.R.G. Green

MRC Applied Psychology Unit

ABSTRACT

Is graphical notation really superior to text, or just different? This paper reports observations of professional hardware designers and considers claims they make about graphics and text on a variety of issues: overviews, zooming, adjacency, detail, viscosity, searching, and space consumption. This paper concludes that the key factor in choosing between graphics and text is the accessibility of information demanded by the user's task.

1. INTRODUCTION

Some writers on visual programming assume that graphical representations are self-evidently superior to text in all conditions and for all purposes (Shu, 1988, is one instance). We doubt that. This paper asserts that graphics and text each have their places, and that the determining factor is the user's task (cf. Bouwhuis' goal-directed analysis of reading, 1989, and Pennington's suggestion that a programmer's task goals influence mental representations, 1987).

Electronics designers have been using a graphical notation for years. This paper considers in detail some of the claims about graphical notation made by expert hardware designers and CAD-E (Computer Assisted Design systems for Electronics) users. Where possible, relevant psychological considerations are brought to bear on those claims.

1.1. Introduction to CAD-E

Schematic drawings for electronics are largely, but not exclusively, graphical. (See Figures 1 and 1.1. Note: Although all of the figures given in this paper are manufactured, they are all based on actual (proprietary) examples shown to us by the experts we interviewed and are accurate reflections of their subjects.) The notation is multi-layered, although not strictly hierarchical. (Figure 2 gives a "block diagram", or top-level view, of the circuit expanded in Figure 1. Figure 3 gives a further expansion of a section of Figure 2.) Text is incorporated as annotation or addenda to diagrams, and sometimes as the lowest level in the design (e.g., parameters of a component). Often the schematic is only a partial representation of the final object, dependent on information held elsewhere for the complete implementation. (It is similar in this respect to declarative programing languages, for which the language implementation provides algorithmic information not specified in the program. cf. Petre and Winder, 1990) What began as a paper- and-pencil

notation has adapted well to computer technology, and some of the most advanced graphical interfaces have been developed for CAD-E. Although systems are available for PCs, those used by engineers in this study ran on workstations, with large, high- definition screens.

1.2. The nature of the experts and interviews

Five expert digital electronics engineers and CAD-E users, all with ten or more years of experience and all members of hardware design teams in industry, were interviewed and observed *in situ*. All use CAD systems (mainly commercially-available systems, but some 'home-grown') running on workstations, and all are familiar with more than one system.

The interviews were informal, set (confidentially) in the context of the engineers' current work-in-progress, so that the engineers provided their own examples. Having been asked to discuss and demonstrate the CAD-E systems they use and what they use them for, the engineers needed little further prompting, so that the interviews were led by the experts, supported by a fixed agenda of questions from the interviewer.

2. THEMES FROM THE INTERVIEWS

We report seven major themes, each mentioned in some form by every designer. All quotations are from these interviews.

2.1. Overviews

Claim:
Graphics is better for overviews.

> *"In a case where you've got things like the flow of a signal, the graphics makes it a hell of a sight easier to understand what's going on."*
>
> *"The shape of the diagram gives information about the content of the diagram, at least for the initiated."*

Comment:
There appear to be three explanations.

First, discussions about graphical notations frequently incorporate some notion of a 'gestalt', an informative impression of the whole diagram which provides insights to structure or flow and is useful as a reference for navigation. It is argued that this distinguishes graphical representations from text. Results in visual perception (e.g., Humphreys and Bruce, 1989, or Treisman, 1986) suggest that, for plain, unformatted text at least, this is true.

The effective visual field is determined largely by visual grain, by the number of visual elements in the objects under scrutiny. Because of the complexity and density of the characters, text is more affected by actual distance than pictographic information in terms of overall perception. Hence, 'stepping back' from a slab of text may well put it out of effective visual range. Moreover, because letters are not simple, and because the density of plain text minimizes the distinctions between global and local features, the very nature of plain text counteracts the simple perceptual interactions of visual elements (e.g., perception of texture, clustering, 'pop-out' of shapes and boundaries) that might produce 'gestalt' effects. Further, the learned associations with text disrupt automatic perceptual processing.

Second, Kosslyn (1978) and Rohr (1986) suggest that relations among objects are visually/spatially grasped, and so it is easier to derive a mental model of a system structure from

a graphical representation than from a textual one. In mental simulation, people often use visual/spatial models which they place themselves inside and move through mentally. In general, if mental representations are visual, there will be be less translation between external graphical and internal representation, at least in the early stages of comprehension and acquisition. It is worth noting that most electronics diagrams are fairly flat, usually with no more than two or three layers (although there may be more for more complex problems treated in more detail), so that the structures discussed are broad rather than deep–just the sort of map-like, mainly two-dimensional abstractions that people handle well (see, e.g., Gilhooly et al., 1988).

Of course, not everything is grasped visually/spatially, just as not every entity has spatial components. Rohr (1986) concluded that highly abstract and more exclusively event-related functions ("existential functions") are ill-served by visual concepts; these may be grasped verbally. Hence, this study observed the transition to text for component properties that are not spatial.

Third, an entirely different type of explanation relies not on matches between external representation and supposed internal representation, but on accessibility: with a graphical overview we can more readily scan elements within their structure of relations, because the search path to each of those relations is shorter and less difficult.

The three explanations offer different (and complementary) ways to improve systems and notations. There are two mechanisms provided in CAD-E that are relevant to providing information to help form overviews: zooming devices and the use of neighborhood. These will be considered in the next two sections.

2.2. Zooming

Claim:
Designers emphasize the need for the improved detail or the overall gestalt-like views obtained by changing scale, i.e., *zooming.*

> *"...Zooming for instance. There's something you can't do with text. You can get information standing back from a diagram, seeing the whole thing at once, but it doesn't tell you anything with text."*

Comment:
The term 'zooming' is applied to two sorts of facilities: *uniform zooming,* in which the content is uniform but the view varies, usually continuously, as though moved closer or farther away; and *stepped zooming,* in which 'pushing' on an object causes a move to a lower-level representation of that object in the same physical position, as though the object were opened up or expanded in place.

Uniform zooming (illustrated in Figure 4) is often used as an aid to navigation, with a 'far' view acting as a map and a position reference. This is often used in combination with windowing, so that different windows are opened on near and far views of the same diagram. Few, if any, text editors support uniform zooming.

Stepped zooming (illustrated in Figures 3 and 5) is something like macro expansion in programming, or, if the technique were available, like seeing a function expanded at the place where it is called. By unfolding objects in place, and allowing the engineer to trace from the outside of the object to the inside continuously, stepped zooming permits bridging among levels of abstraction by making it possible to interleave several different levels in a single view. Interleaving levels helps designers to perceive the functionality of lower-level components by showing them set in a higher level or, if necessary, in the level above that. (Conversely, design studies of programmers (e.g., Guindon, 1988) have shown the importance of discovering low-level consequences of high-

level decisions at an early stage without immersing the designer in unnecessary detail.) Importantly, the stepped zoom exposes calls to still lower levels. The nearest text equivalent is probably hypertext. However, in early hypertext systems users easily lost context, which was not mentioned as a problem in these graphic CAD-E environments, where the context is maintained in the same view.

2.3. Neighbourhood: adjacency and locality

Claim:
The nature of graphics allows *connectedness* to be represented by a line, and thus kept separate from *relatedness* (represented by adjacency). Hence, adjacency or position is always, even locally, a secondary cue, a manipulable variable, available for reinforcing associations, for suggesting structure, or for giving hints to the reader. Text, however, must either use adjacency to indicate *both* relationships, or introduce symbolic links (e.g., cross-references or variables).

> *"You get less clutter in graphics. There aren't arbitrary extra variables running around; you just link one thing explicitly to the next."*
>
> *"Adjacency is what you use to make things clear, it's not automatic. Think about the mess they got into with automatic generation of flow charts from code. There's implicit information in grouping and adjacency: long lines, short lines, which way round things go."*
>
> *"The difference in the design between mechanically identical designs done by a novice and an expert, is the novice engineer's will be more difficult to comprehend because of the layout.".*

Comment:
It appears that the *extra dimension* of adjacency is used to express something akin to the notion of *programming plan:* a description in a functional domain (e.g., Gilmore and Green, 1988, Pennington, 1987). Unenhanced text displays plan structures badly, precisely because no extra dimension is available.

Electronics design conventions attempt to reinforce meaningful use of adjacency and to avoid awkward distances between related things. Hence, signals tend to be local, appearing on a few pages of the many that specify a large system. But, as a manipulable quality, adjacency in graphics is subject to personal style and skill, and is not used well automatically.

Compare Figure 1, a "good" example, to Figure 6, the corresponding "awful" novice example. The two are informationally equivalent, that is, they "compile" to the same machine. The "novice" example fails in a variety of ways to exploit the available manipulable cues:
- neglect of adjacency cues: things are grouped visually which are not related logically,
- dispersal of related components, e.g., U8 and U10,
- substantial confusion of signal flow; unnecessarily convoluted wiring (compare Figures 1.2 and 6.2),
- neglect of active-low/active-high conventions (i.e., "bubbles" are used haphazardly).

Other *mis-cues* are noted on Figure 6.1.

"Engineers will discourse for hours on the disgustingness of badly done circuit diagrams." It is not unusual for engineers to re-draw a circuit from a document in order to put it into a form meaningful to them. Similarly, programmers often fail to take advantage of what local organization *is* available in text (cf. Waller's argument that text designers can make better use of the perceptual cues, 1980).

2.4. Shifts from graphics to text, or, what *text* is good for

Claim:
Text has its uses; even after reasonable evolution, electronics schematics remain a multi-level system with alternative representations in text.

> *"The purpose of the schematic is to show connectivity--what's used where and how it's used. What it's very bad at giving you is a description of the components therein."*
>
> *"[Schematic representation has] stood everybody in great stead and still does, apart from special cases where the implementation constraints start becoming important and you have to mention them on the same piece of paper as the schematic, by text...where the physical constraint is of the properties of the things you're indicating by objects on a schematic."*

Comment:
Three important shifts from graphical notation to text were noted, in addition to annotations to the diagrams (e.g., labels or component values, see Figure 1.1):

1. the lowest level of description, usually library information concerning properties of components (this may or may not be available from the CAD system);
2. appendices, often parts lists or textual descriptions (usually not handled within the CAD system);
3. the net list, a list of all connections among elements (produced automatically by the system from the schematics).

In each case, the text offered the most precise and detailed view of the aspects of the diagram covered. The net list, primarily an interchange mechanism between the graphic editor and all the other programs in the CAD suite, is still used by engineers in checking designs, as a sort of 'representation of last resort'.

Often the schematic is only a partial representation of the final object, dependent on physical details expressed in text. The graphic representation abstracts from physical details (e.g., physical packaging and implementation constraints), just as any formal notation is a deliberate abstraction. The persistence of text in CAD-E highlights the need in any formal notation for some underlying--and preferably accessible--universal form that can capture the details outside the formalism. In CAD-E, text provides information that graphics may not (e.g., component parameters, inter-relationships between component layout and physical packaging); "You can always say it in text, but not always in graphics."

Text is also used for compiling parts inventories and net lists, because there is no graphical equivalent for ordered look-up.

The engineers associated shifts from graphics to text with shifts from functional reasoning to implementation reasoning and with shifts from general functional and structural information to details and constraints. They refer to text more often with more difficult designs, ones that use the components closer to their specified limits.

2.5. Viscosity

Claim:
Massive changes to graphic representations are hard work: if a group of components is moved, connections have to be re-established. Similarly, the cases where moving a piece of text is difficult are where connectivity has been established explicitly; in other cases, blocks of text may be moved freely without difficulty.

> *"Small changes are small and large changes are large, but the definition of small is more strict than it is in text, by which I mean that a small change is a change to a few things connected to each other at one level in the hierarchy, whereas a large change would be a*

change at many levels of the hierarchy, or a similar change at lots of different places...That's where it's very different to text...which relies on symbolic similarity, not the semantic similarity."

Comment:
Replacement in CAD-E is nearly always manual, and global replacement is available only in the form of object re-definition: *"Change all somethings is usually done by changing the library entry, i.e., change not what is being called but the substance of the thing which is being called...There's a difference of opinion about whether it [global search-and-replace] should be do-able: it's like changing a system call on a multi-user operating system."*

Green (1989) points out the universal HCI problem of *viscosity:* resistance to change of a design. The apparent high viscosity of graphics systems provokes changes in the designers' strategies; because the absence of 'power tools' discourages the types of change that require global thought or global manipulation, designers cope by applying a more local method or a more hierarchical structure. For example, electronics, like software, has used modularity to control signal complexity. A hierarchy is applied to signals, so that the signals in a sub-component are local to the diagrams concerning that sub-component.

2.6. Search trails

Claim:
Searching is well supported in CAD-E in the domain of connectivity, but not in the domain of functionality.

"Schematics show connectivity, but they're bad at showing any other things, like the precise actions applied to the signals. Sure, the symbol suggests the function, like a print *statement suggests what it does, but the parameters may vary, and those may be fiendish to get at."*

"...It's possible to make assumptions about what a symbol means that turn out not to be true in practice. That's the sort of thing that results in masses of text appended or long amounts of time spent debugging or things going wrong in the field."

Comment:
Following a signal in a schematic is a matter of following a line or of following a signal name from one page to the next. The CAD system can help by highlighting the whole signal path (as illustrated in Figure 1.2) or by stepping along it. The search may be more complex than following a single line; some signals have many sources and many 'sinks'. Notable is the symmetry of the signal trail: a signal can be traced to its source or followed to its destination. This is in contrast, for example, to a spreadsheet program, which provides pointers only in one direction, and to many other information designs (Green, 1989).

Although the signal path is explicit (as wires), the action that a component has on a signal may be affected by a parameter buried in a lower level. Such side-effects may be difficult to unearth. Symbol shape is associated with component function. Implicit in this evolved use of perceptual cues (embodied in the symbol) is a notion of what information is of primary importance to designers, i.e., function over value. But the engineers also mentioned ambiguity resulting from this sort of abstraction, because parameters constraining the function are less accessible.

A single physical component may have more than one graphical representation giving more than one view of its functionality. For example, an 'and' gate may be viewed as the 'and' of true things or as the 'or' of inverted things. (See Figure 7, an example of a component "library", which shows dual views of many of the components.) Further, a given circuit may have more than one graphical representation (as illustrated in Figure 8). Used carelessly, this may

contribute to ambiguity of function, but, used well, it may be manipulated to provide additional perceptual cues to function. Compare Figures 1 and 6. Circle 2 of Figure 1.1 shows a "good" use of alternative representations; the same component (U1) is shown in two different ways as an aid to clarifying function. Circle 4 of Figure 6.1 illustrates a *mis-cue:* the two gates are erroneously made to look like a bi-stable flip-flop. As with adjacency, these manipulable cues or "extra dimensions" are subject to individual differences and tend to distinguish novices from experts.

2.7. Vocabulary and space consumption

Claim:
Although it is commonly believed that graphics is more compact than text, the comparison is not straightforward, particularly because most graphical notations rely to some extent on text to keep their vocabularies manageable. *Complete* system descriptions tend to be massive in either graphics or text.

> *"You can get more on a page [with graphics]..."*
>
> *"The early pictorial version of the schematic had the advantage of telling you what type of a component you were dealing with; the schematic now gives you no hint as to its physical representation...In the beginning, there were two or three types of resistor [and a symbol for each]. You're now dealing with several hundred, so the schematic representation has gone from three to one...what will help is the text normally appended to a schematic, a parts list or whatever."*

Comment:
A survey of the contents of the components libraries used by the engineers in this study suggested that CAD-E libraries used for designing large digital systems in a given technology are likely to contain in the range of 50 to 65 basic component symbols (see Figure 7), which are used with textual labels to represent 100 to 500 actual part types. Symbols for given functions are similar across technologies. Symbols for similar functions share some resemblance. A variety of attributes distinguishes the symbols: shape (with smaller units, like gates, having varied shapes, and composite units, like memories, typically being variations of rectangles), aspect ratio, size, numbers of inputs and outputs.

The resistor example that appears in the quote above illustrates an important trade-off in graphical notation: between graphical purity and vocabulary size. Should each sort of resistor have its own symbol, or, given that all resistors serve the same basic function, should they be represented by a single symbol annotated with text indicating their individual values? Conciseness and discriminability among symbols argue for the annotation.

The numbers of components employed in a single design varies tremendously, depending on the problem, but for medium to large digital designs the number of different component types (and hence different symbols) seems to range around 50 to 70. Although plotter size can be a variable in how the engineer lays out a design because of transitions from one page to the next, density apparently increases approximately linearly above the minimum useful page size, usually A3 in Britain. The engineers studied usually produce A3 drawings, on which they expect to place 20 to 60 components. Although symbol size varies somewhat with enlargement and reduction, and symbol size is affected by the numbers of input and output wires that must be accommodated, symbols tend to be roughly the same size (say, within a factor of 2), fitting in an area around 3 to 5 centimeters square.

Whether or not "you can get more on a page" using graphics (at least annotated graphics) rather than text, is uncertain. Arguments about relative usable density of graphics and text are inconclusive because of the difficulty in identifying comparable units in each style of notation.

Certainly, in schematics, reduction is limited by the textual annotations, whose smallest elements (the characters) are smaller than the graphical symbols, and which in the scale typically used are close to their minimum usable size. (Imagine trying to read a diagram with graphical symbols the size of text characters.)

3. CONCLUSION

> *"Forget* 'all the information'; *what you want is the* salient *information and let the rest go hang."*

A decade ago, Fitter and Green (1979) argued that the critical factor determining comprehensibility of notations was accessibility of information. They listed five criteria: "A good notation [1] should present *relevant* information in a perceptual form, or [2] should use *redundant recoding* to present important parts of the symbolic information in a perceptual code as well; [3] it should *restrict* users to objects that can readily be understood; [4] it should *reveal* the underlying mechanisms and be *responsive* to manipulation; and [5] it should allow easy and accurate *revision."* [Emphasis and numbers added.]

Seven themes emerged from the interviews. They can be related easily to these criteria. First, the designers favoured graphics for its better presentation of *overviews.* This is a standard claim. It indicates not only the undisputed importance of perceptual organization [1], but also the sometimes-overlooked importance of having all requisite information simultaneously available, as indicated by the next two themes: the increased *zooming* capabilities of graphics, and the 'extra dimension' of *adjacency,* used heavily by the designers as a perceptual cue to implicit organization [2]. Text was not just favoured but required for the next theme: the presentation and *manipulation of detail* (e.g., ordered sets, physical packaging, implementation constraints), elements excluded from the graphical formalism [4]. Theme five was the problem of changing designs, the *viscosity* problem that afflicts any notation, which is severe enough in these systems to make it difficult to rework poor designs [5]. Theme six was *search trails:* signal flow is well revealed and responsive to exploration [4], but constraints on functionality are not; text is required for these. Lastly, text annotation is used to restrict the *vocabulary* of graphical symbols [3], but no clear statement could be made about *space consumption.*

Fitter and Green's criteria do not cover two other aspects of the results. One is individual skill differences. Even a well-evolved notation is vulnerable to weaknesses in individual expressive skill. On the other hand, skill may help compensate for weakness in a notation (e.g., the engineers described how they use modularisation to help combat viscosity). The second aspect is the need to escape from formalism. Every notation, graphical or symbolic, is a formalism, designed to present a simplified view of the world. In laboratory study, formalisms may be adequate, but in real-life professional-level design there must be a mechanism to express arbitrary constraints outside the formalism.

Bouwhuis (1989), in his analysis of reading as goal-driven behaviour, characterizes the tradeoffs among criteria: "Reading is a fight for percentages." So it is with electronics schematics. Bouwhuis observes that text is well-evolved: "...there is an almost perfect correlation between the temporal organization of the reading process and the spatial organization of text." Similarly, electronics schematics have evidently evolved well in their own terms.

We argue that the observations presented indicate *not* that graphical systems are universally superior to text, but that the two systems, text and graphics, each have preferred uses; and, moreover, that the uses of each are exactly in line with the argument that accessibility of information is a key factor.

4. ACKNOWLEDGEMENTS

This paper expands work first presented at Interact '90 (IFIP Conference on Human-Computer Interaction, held in Cambridge, England, in August 1990). Much of this research was performed while MP held a Research Fellowship at the Institute for Perception Research (IPO) in Eindhoven, the Netherlands. Part of this research was performed while TG was seconded to Rank Xerox EuroPARC; we thank Tom Moran and Rank Xerox EuroPARC for providing facilities for MP.

We thank the engineers for their cooperation and enthusiasm, and Peter Eastty for invaluable comment.

5. REFERENCES

Bouwhuis, D.G. (1988) Reading as a goal-driven behaviour. In: B.A.G. Elsendoorn and H. Bouma (eds.), *Working Models of Human Perception.* Academic Press. 341-362.

Fitter, M., and Green, T.R.G. (1979) When do diagrams make good computer languages? *IJMMS,* **11** , 235-261.

Gilhooly, K.J., Wood, M., Kinnear, P.R., and Green, C. (1988) Skill in ma p reading and memory for maps. *Quarterly Journal of Experimental Psychology* **40A** , 87-107.

Gilmore, D.J., and Green, T.R.G., (1988) Programming plans and programming expertise. *Quarterly Journal of Experimental Psychology,* **40A** , 423-442.

Green, T.R.G. (1989) Cognitive dimensions of notation. In: A. Sutcliffe and L. Macaulay (eds.), *People and Computers V.* Cambridge University Press.

Guindon, R., Krasner, H., and Curtis, B. (1987) Breakdowns and processes during the early activities of software design by professionals. In: G.M. Olson, S. Sheppard, and E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop.* Ablex. 65-82.

Humphreys, G.W., and Bruce, V. (1989) Visual processes in reading. In: *Visual Cognition: Computational, Experimental, and Neuropsychological Perspectives.* Lawrence Erlbaum. 227-286.

Kosslyn, S.M. (1978) Imagery and internal representation. In: E. Rosch and B.B. Lloyd (eds.), *Cognition and Categorization.* Lawrence Erlbaum. 217-257.

Pennington, N. (1987) Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive Psychology,* **19** , 295-341.

Petre, M., and Winder, R. (1990) On languages, models and programming styles. *The Computer Journal,* **33** (2), 173-180.

Rohr, G. (1986) Using visual concepts. In: S.-K. Chang, T. Ichikawa, and P.A. Ligomenides (eds.), *Visual Languages.* Plenum Press.

Shu, N. C. (1988) *Visual Programming.* Van Nostrand Reinhold.

Treisman, A. (1986) Features and objects in visual processing. *Scientific American,* November 1986, 106-115.

Waller, R.H.W. (1980) Graphic aspects of complex texts: typography as macro-punctuation. In: P.A. Kolers, M.E. Wrolstad, and H. Bouma (eds.), *Processing of Visible Language 2.* Plenum Press. 241-253.

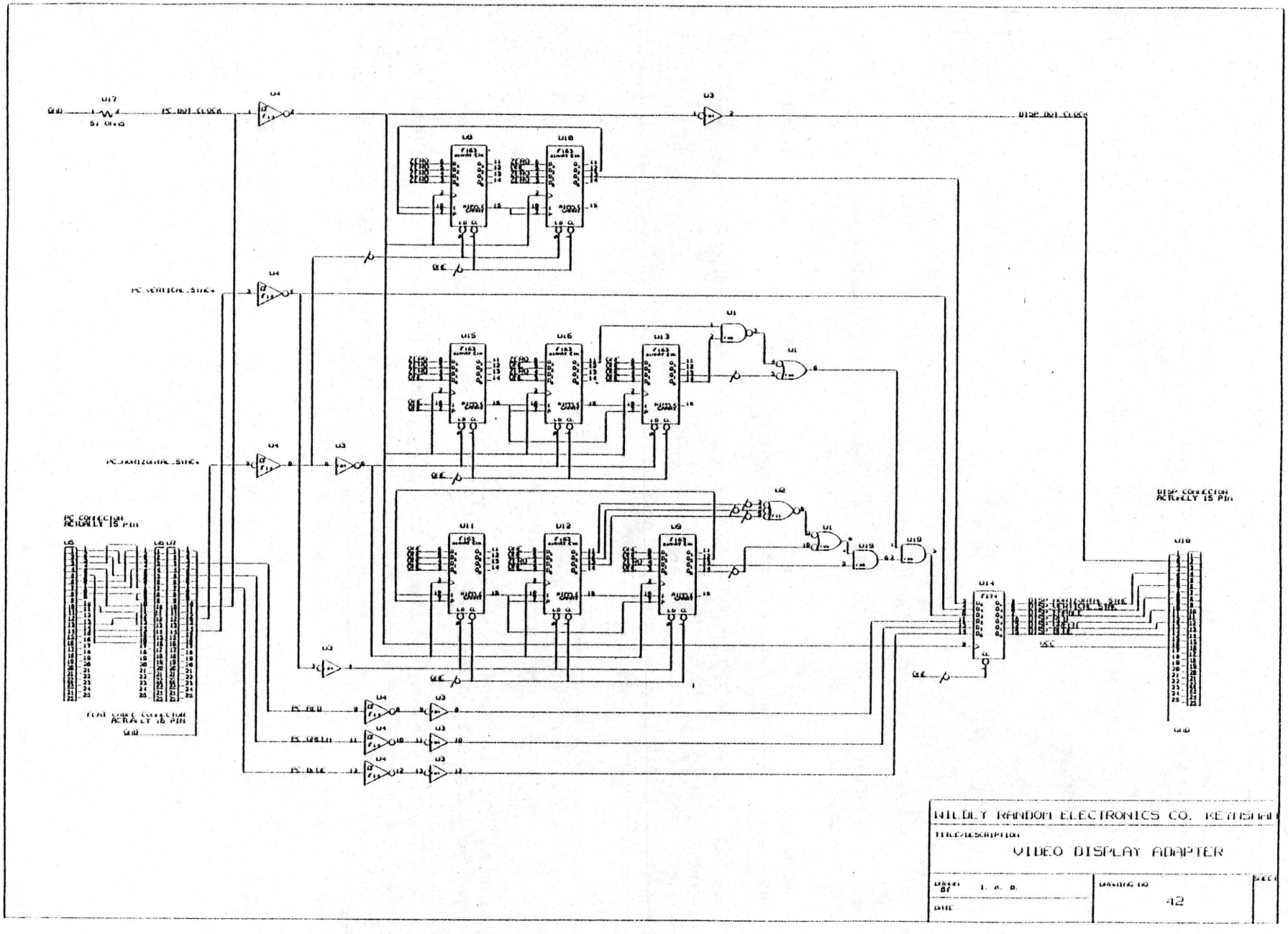


Figure 1: This is example circuit is typical, although the symbols are not as densely packed as in many diagrams. Compare this to Figure 2 (the corresponding "block diagram") to see how adjacency and clustering are used to associate components of "blocks".

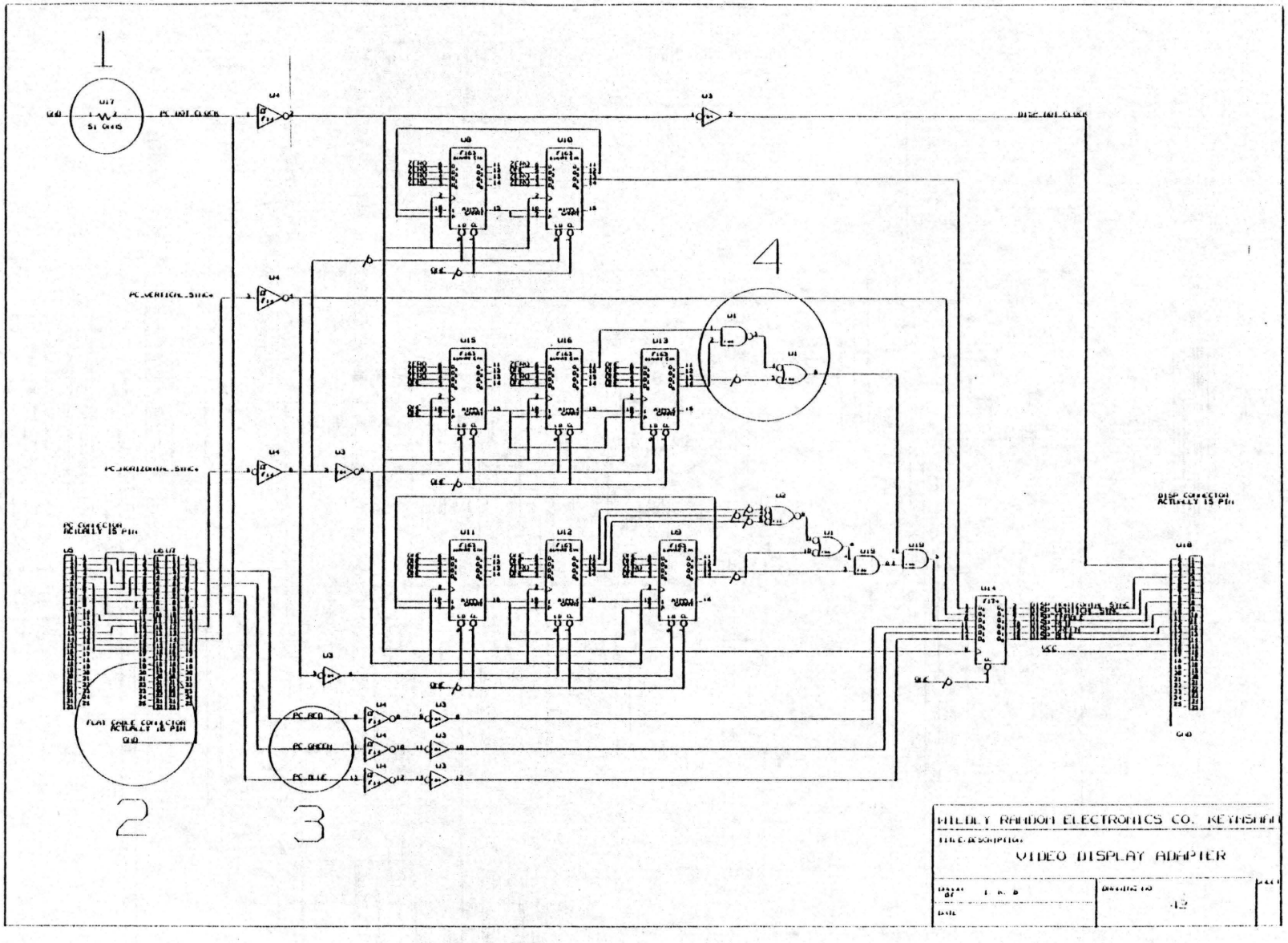


Figure 1.1: This variation of Figure 1 illustrates some notable details about the use of text and choice of graphical representation.

Circle 1: The text annotation below the resistor symbol gives the resistor value.

Circle 2: This annotation is a comment; the designer did not have the required component in the component library and so used something close.

Circle 3: These annotations are signal names.

Circle 4: The same component is shown two different ways as an aid to clarifying the function.

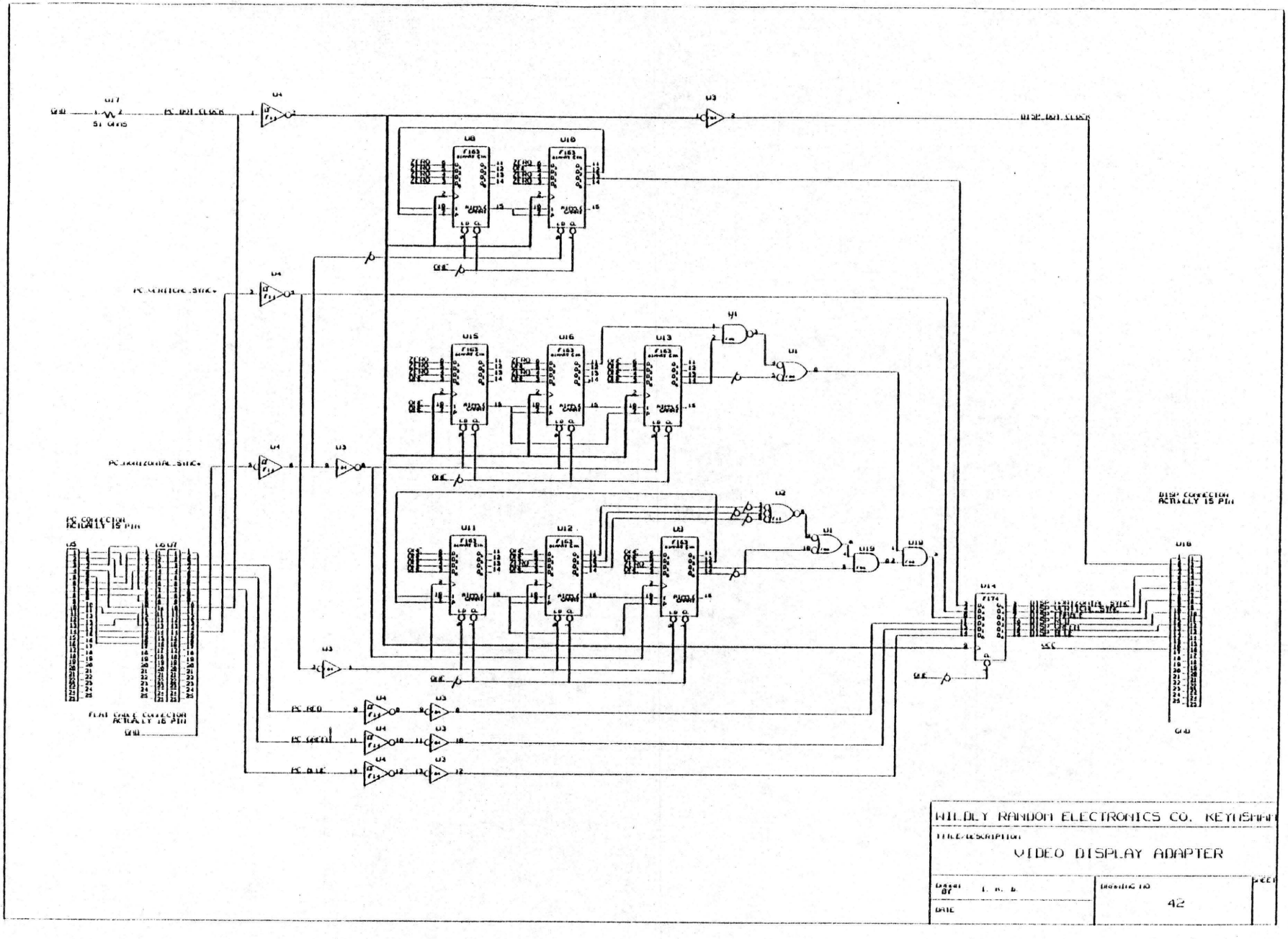


Figure 1.2: This variation of Figure 1 highlights a signal path.

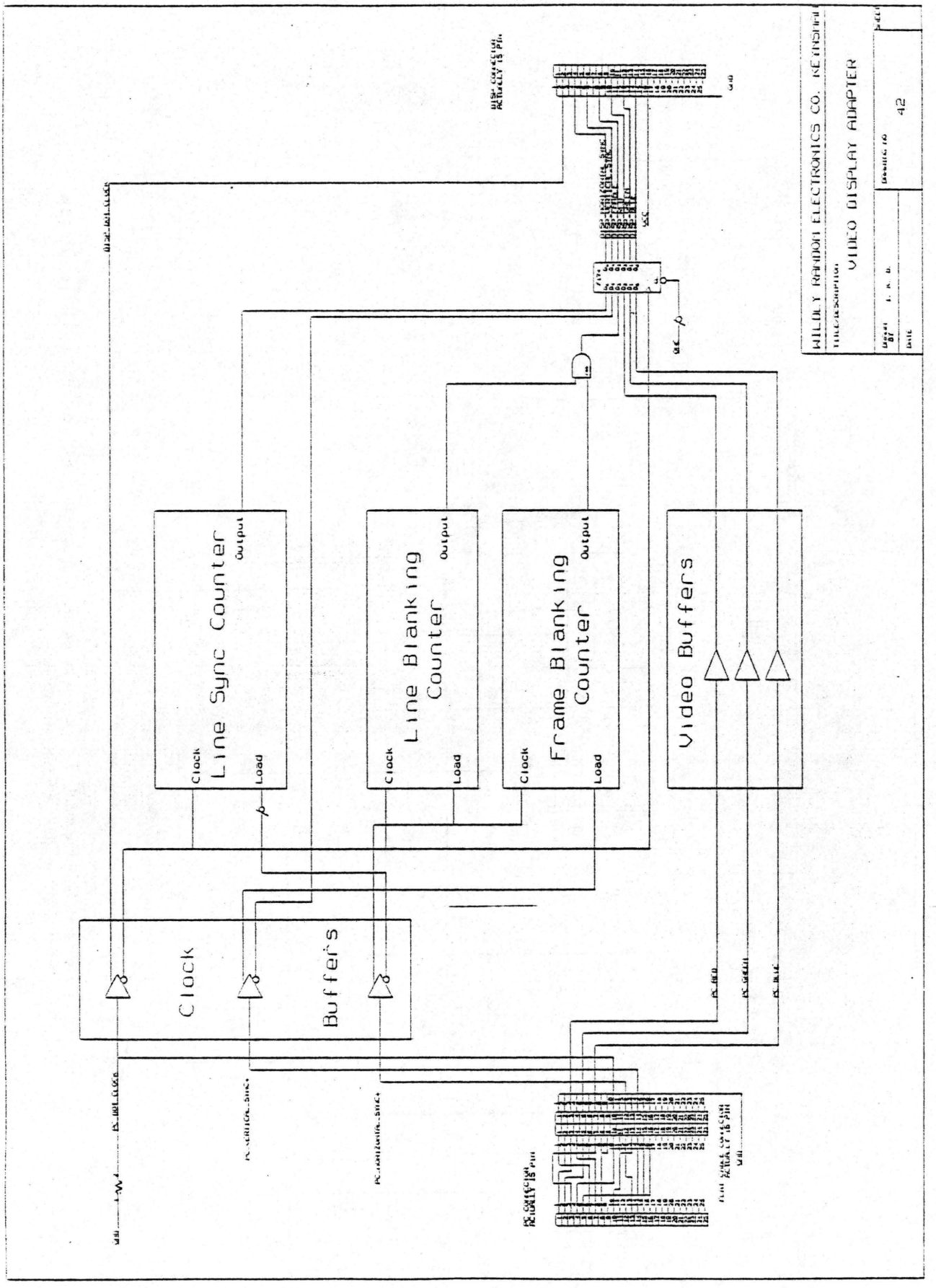


Figure 2: This is a "block diagram", a higher-level version of the circuit expanded in Figure 1.

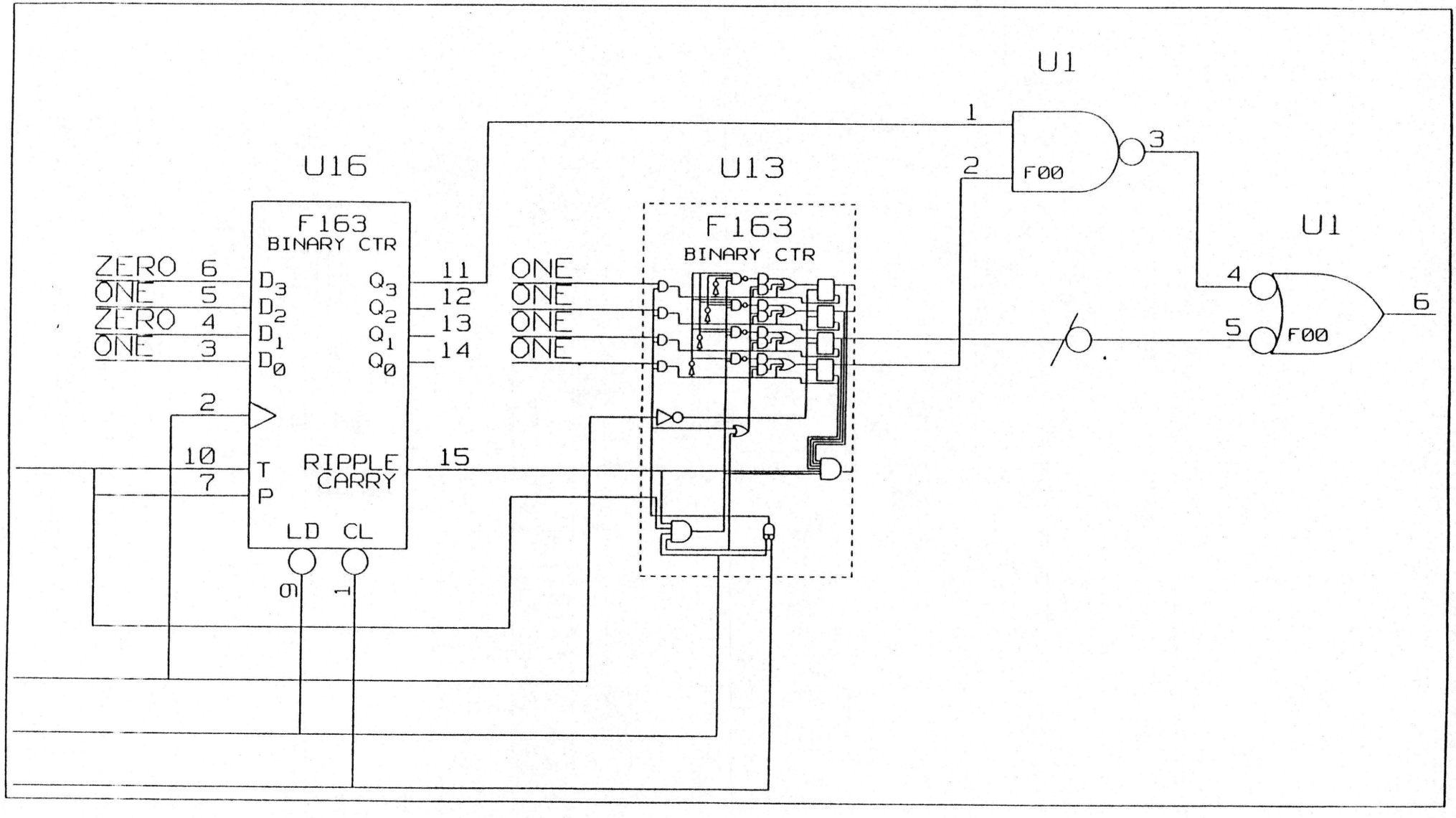


Figure 3: This fragment of Figure 1 shows one component (U13) expanded in place to show a lower-level, more detailed representation.

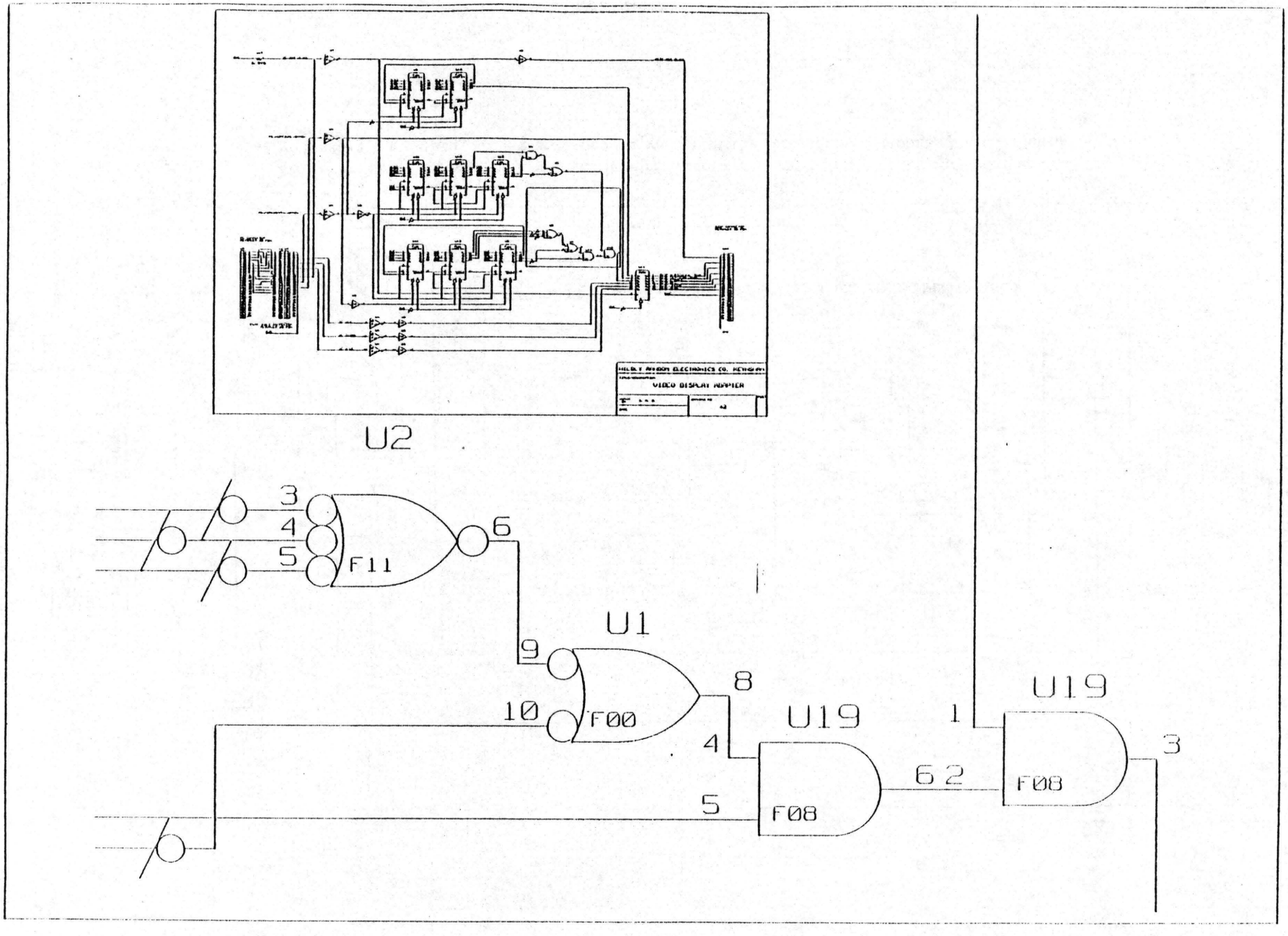


Figure 4: This mock-up illustrates *uniform zooming*. The "far view" in the top, left corner gives the context from which the "near view" has been enlarged (see the small box in the middle-right of the far view).

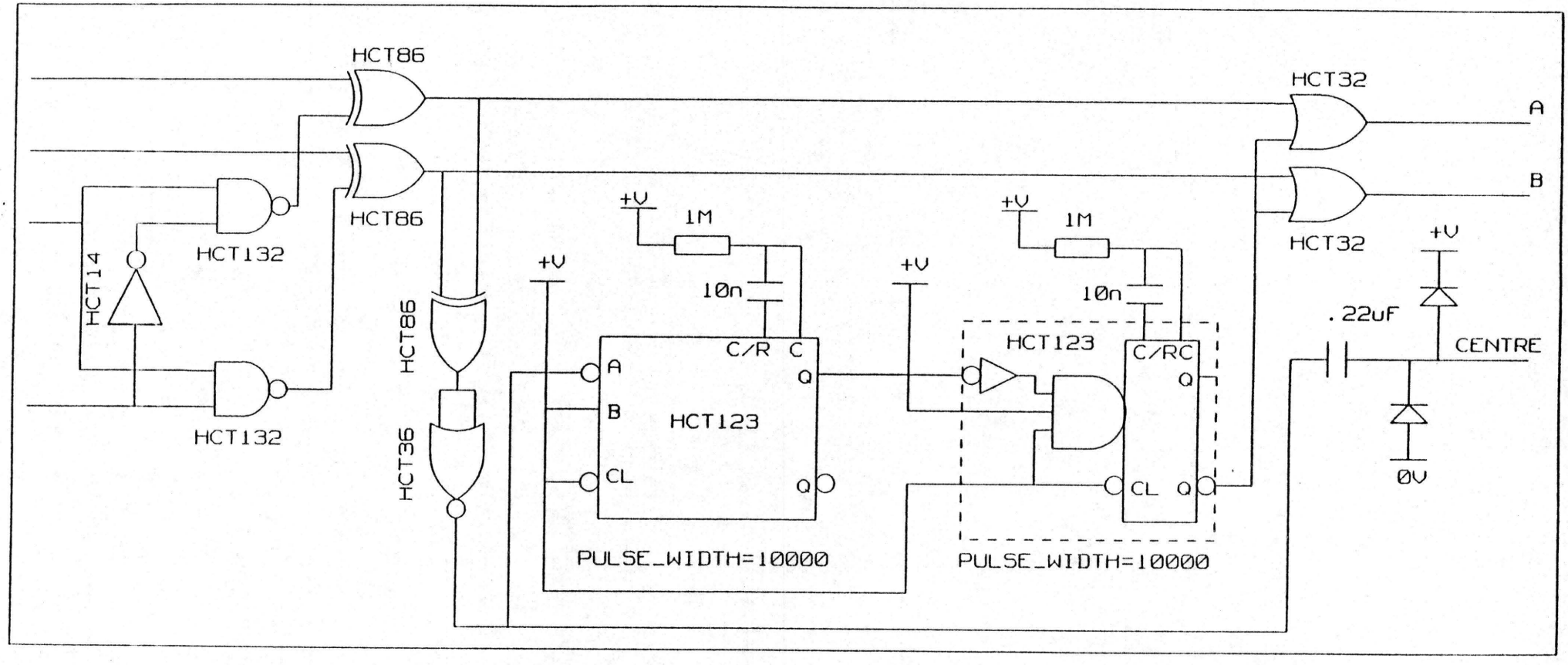


Figure 5: This fragment of a schematic illustrates a simple *stepped zoom:* the components enclosed in the dashed box are an expanded version of the rectangular component (also labelled HCT123) to their left.

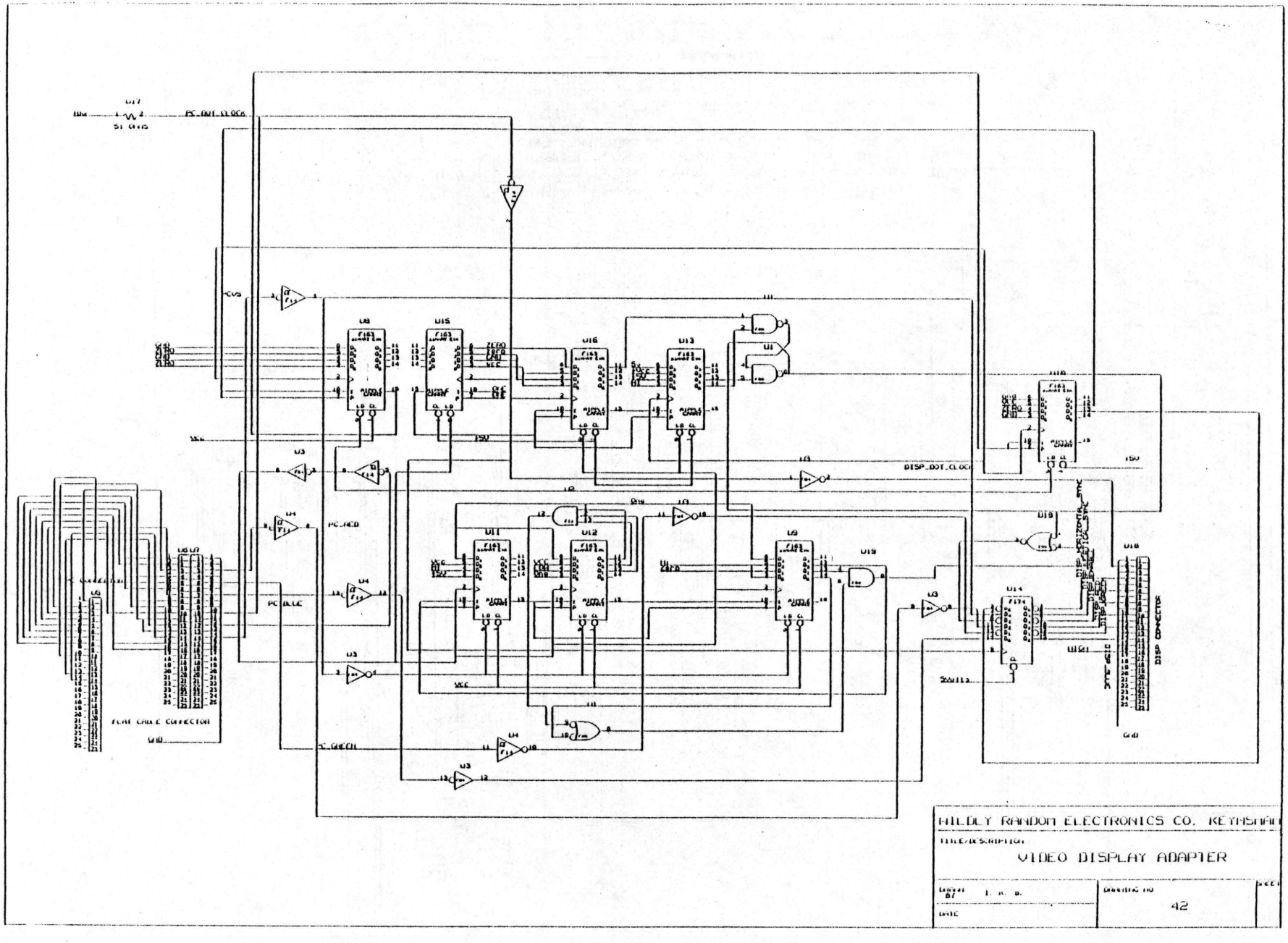


Figure 6: This "awful" diagram is informationally equivalent to Figure 1 but includes a number of *mis-cues*, such as neglect of adjacency (dispersal of related components and grouping of unrelated ones) and confusion of signal flow.

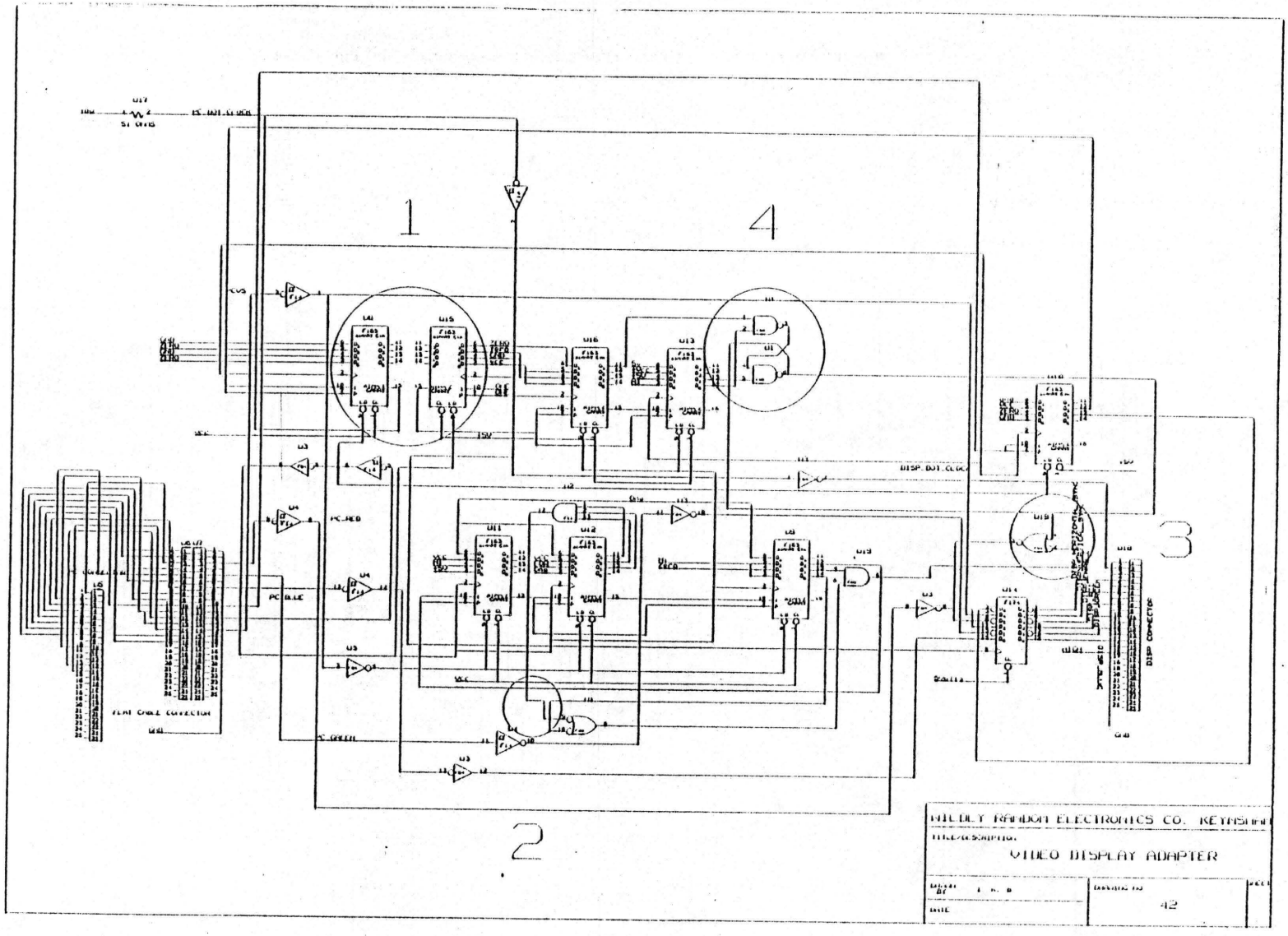


Figure 6.1: The circles on this variation of Figure 6 enclose some significant *mis-cues:*

Circle 1: Although these two components are of the same type, one has been reflected (its inputs and outputs rearranged), giving a misleading impression of symmetry. Their visual symmetry and proximity do not reflect logical relatedness; on the contrary, these components are unrelated in their use.

Circle 2: The extra bends in this wire are probably an editing fossil. Notice also the convoluted wiring at the left of the diagram.

Circle 3: The wires to this gate have two peculiarities: the wire enters input 1 at an awkward angle; and there are two routes (one redundant) into input 1.

Circle 4: These two gates have erroneously been made to look like a bi-stable flip-flop. This is a striking misuse of a strong perceptual cue.

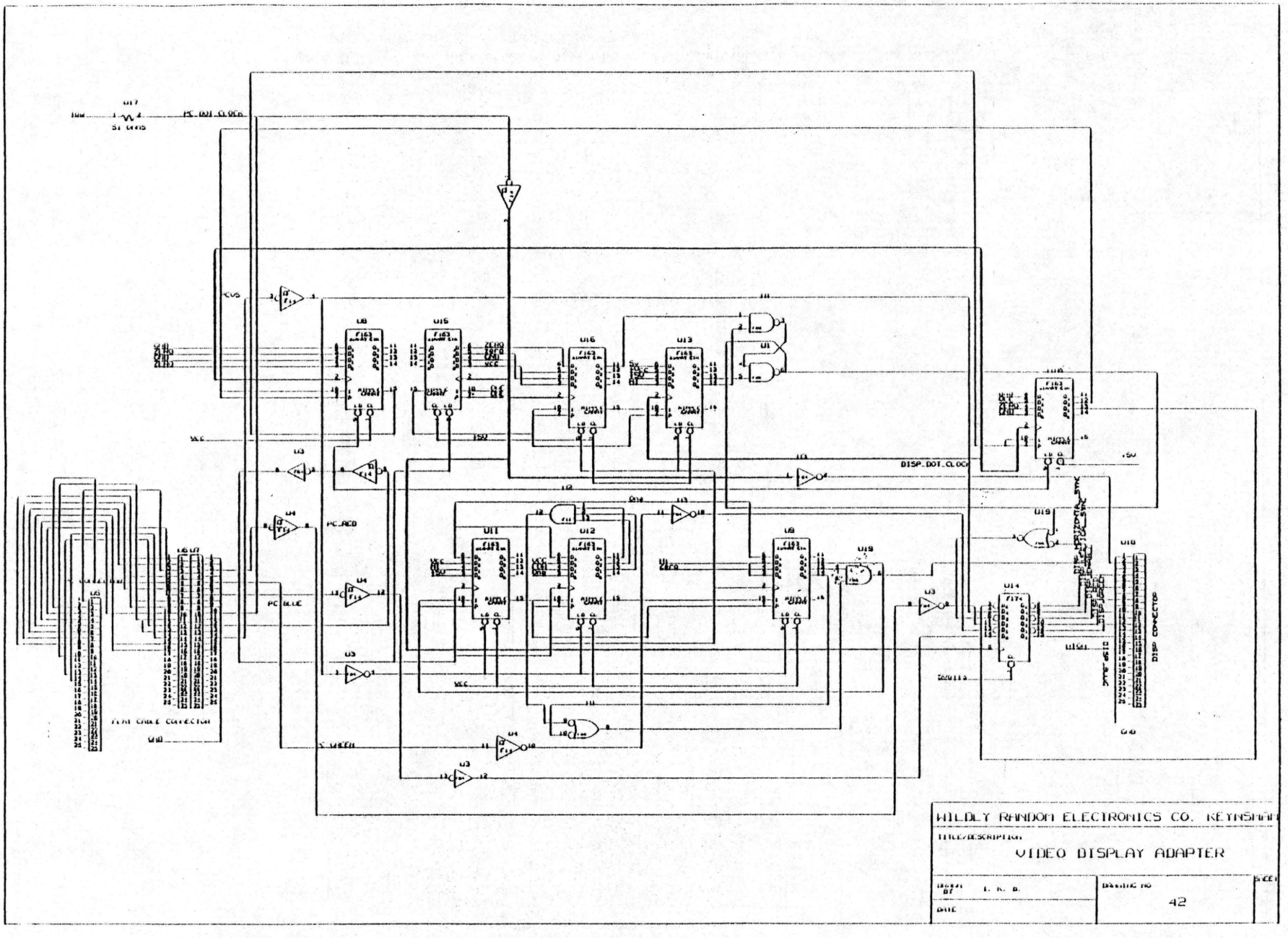


Figure 6.2: This variation of Figure 6 highlights a signal path and so illustrates the convolution of the wiring. Compare to Figure 1.2, which highlights the corresponding signal path.

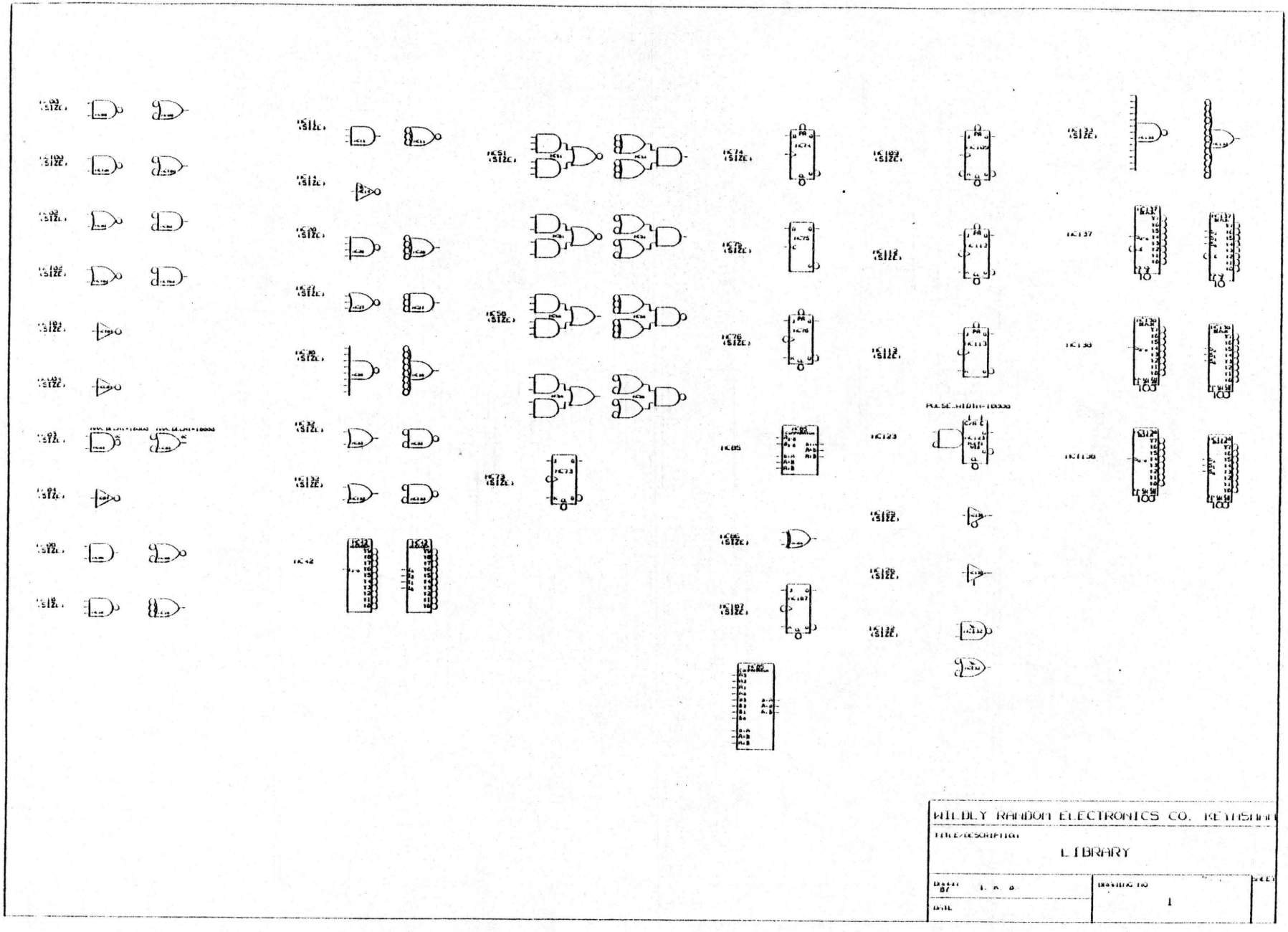


Figure 7: This is one page of a two-page "library" of components. Notice that the first three columns of components all have two representations.

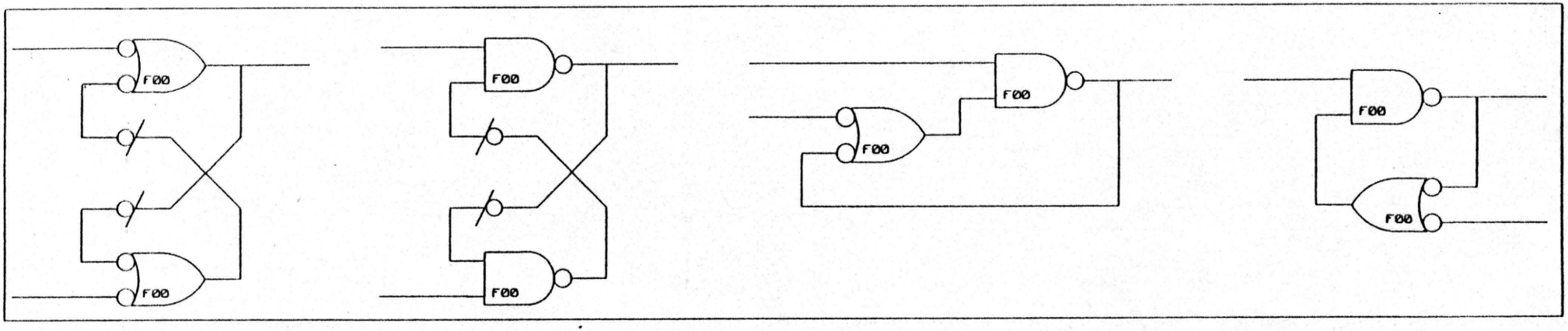


Figure 8: This example illustrates four informationally equivalent representations of a bi-stable flip-flop. A "good" designer would choose one of the four based on the particular context and use. The two on the right are rarely used. The two on the left would both "pop out" of a typical circuit diagram, where most wires are parallel or perpendicular to all other wires.

Using CLG and ETAG in designing software systems
Possible extensions
-Steps toward dynamic interfaces-

Horia Pitariu[*], *Cristian Valcea*[**], *Cristian Rapauzu*[***], *Lucian Lapadatu*[***]

[*] Cluj University, Department of Psychology,
Kogalniceanu 1., 3400 Cluj-Napoca, Romania.

[**] Institute for Power Studies and Design (ISPE)
Lacul Tei Blvd. 1., 7000 Bucharest, Romania.

[***] Research Institute for Computer Technique and Informatics,
Calea Floreasca 167., 7000 Bucharest, Romania.

ABSTRACT

This paper describes the work in progress on a project which concerns the implementation of a methodology to design Human Computer Interfaces. We introduce a formalism to investigate the mental and the conceptual model of a computer user, based mainly on ETAG and CLG. In our view the decoding of the mental model must be focused on structuring a metacommunication system which has the purpose to form an optimal mental model at the user's level. In the first stage of the project we outlined the formalism we sustain for investigating the mental model. We also tried to include some adaptive features to the system. Future work will imply the engagement of experiments to validate the psychologic element to the system.

1. Introduction

Cognitive Ergonomics has brought a series of concrete solutions to the design of software systems and of Human Computer Interaction (HCI). The role of Psychology, especially of Cognitive Psychology can not be denied (Hoc, 1989). Many discoveries in Psychology proved themselves to be valid in time and were integrated in the design of HCI, while others, being incompatible with the principles and the solutions of Cognitive Psychology, were left apart (Stagner, 1988). Boring (1929) a well known psychological historian underlined that the aristotelian model prevailed, for a long period of time, the domain of psychical phenomena. The classical Psychology is limited in labeling the psychical processes, without explaining them (Bunge & Ardila, 1987). The aristotelian way of thinking was opposed, according to Levin, to the "galilean" one. The behavior was explained by the quantification of the psycho-social scope, which puts together the actual person and its environment, the external events and internal vectors (goals, aspirations). Cognitive Psychology has renounced to explain the behavior by descriptive concepts, and has ascertained its position on discovering the structures and mental processes responsible for one's behavior. Cognitive Psychology has made the offer to illuminate the behaviourist "black box" intending to explain the behavior. In order to achieve such a performance it designs models of cognitive processes and replaces the classical concepts with these models (Miclea & Radu, 1989).

The design of a HCI implies the analysis of the user's mental model and of the conceptual model generated by the designer (van der Veer et al., 1989). A crucial problem of Cognitive Ergonomics is the accurate description of the above mentioned models. This is made possible by means of a formal language. In this perspective we adopt Piaget's position: "the employment of a logico-mathematical language does not imply the reduction of natural thought to formal models, but instead enables us to use the most precise language to describe natural structures" (Piaget, 1960, p. 182).

Our intention was to think out an efficient methodology for designing an expert system able to tutor and to learn. This intention confronted us with a series of problems that could not be solved by means of the analysis methods offered by CLG or ETAG (Moran, 1981; Tauber, 1988). The formalism we need should encompass the following requirements:

- it should offer a methodology for the software engineer to design his own application taking account of some psychological aspects of the end product.
- it should allow the possibility for the software product to manipulate the communication with the user in such a way that the latter should attain an optimal skill in using it as soon as possible.

We tried to envisage a formalism characterized by dynamism as opposed to a static one, which is not reduced to ascertain facts. This is one of the requirements imposed to a cognitive model, namely to explain and predict the computer system user's behaviour.

2. CLG vs. ETAG

Command Language Grammar (CLG), was designed by Moran (1981). It uses top-down functional analysis, and employs a semi-formal language which can be applied to describe the Human Computer Interface, to analyze the interaction and the psychological processes implied in the use of a computer system. By means of CLG the description of the conceptual model and the mental model of the user has become possible. Moran distinguishes six levels grouped in three components:

Conceptual Component
  Task level
  Semantic level

Communication Component
  Syntactic level
  Interactive level

Physical Component
  Spatial-Layout level
  Method level

An important feature of CLG is its multilevel nature which enables it with a great flexibility and a increased ease of use.

Extended Task-Action Grammar (ETAG) (Tauber, 1988) is an extension of the formalism designed by Payne and Green (1986), the Task-Action Grammar (TAG). Tauber adds to the description of the concept contained in the mental model a series of references to the basic ontological concepts, from the theory of conceptual structure (Jackendoff, 1983). Using ETAG "we have been able to describe some of the consequences of design decisions that were not treated by the original version of TAG - especially decisions about how much information to reveal to the user" (Tauber, 1988).

3. Possible extensions

The idea formulated in the introduction sets forth a different point of view from those stated by CLG and ETAG. Our intention is to decipher the mental / conceptual model in order to build a metacommunication system able to form an optimized mental model to the user. This implies an extension of the formalisms mentioned above with more general notions, closely related to the inner mechanisms of human thought. A synthetic representation of those general notions is presented by Innocent, Van der Veer, and Waern (1988) (Fig. 1).

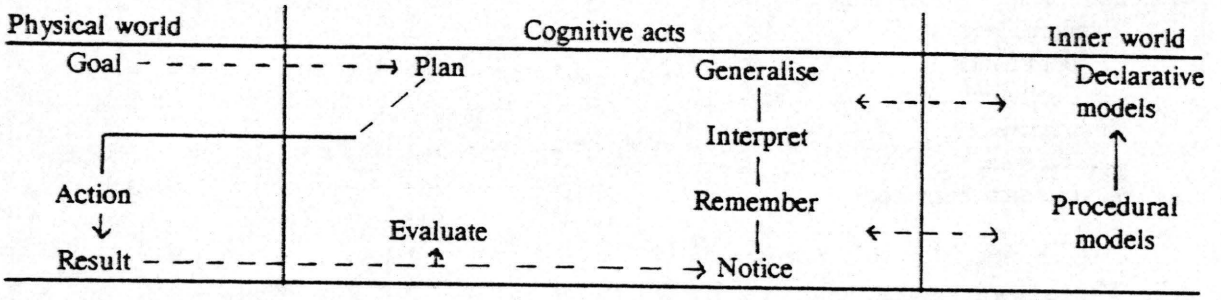


Figure 1. The interaction between physical world, inner world and cognitive acts (Innocent, van der Veer, Waern, 1988).

Starting from this elements the mental model can be structured on five major components:

a. Metaknowledge component
- Metalevel description
  - Concepts
  - Cognitive Acts
- Semantic Metalevel
  - Conceptual entities
  - Conceptual operations
  - Semantic methods

b. Metacommunication component
- Learning level
  - Learning strategies
  - Entities
- Learning operative level
  - Learning methods
  - Learning heuristics
  - Entities

c. Personality component
- Personality level
  - Personality strategies
  - Entities
- Personality operative level
  - Methods
  - Entities

d. Conceptual component
    Task level
        Conceptual entities
        Tasks
    Semantic level
        Conceptual entities
        Conceptual operations
        Semantic methods

e. Communication component
    Syntactic level
        Commands
        Command methods
    Interactive level
        Interactors
        Interactive methods

3.1. Metaknowledge component

The metaknowledge component makes the connection between all the other components, being, on one hand their prototype (ontological concepts), and on the other the conductor of the entire activity of the system (cognitive acts). This means it contains:

- the description of the basic ontological categories (object, place, etc.).
- the description of the other components and their levels (this is the self-consciousness of the system).
- the description of the cognitive acts specific to the human thought.

This component includes Innocent, van der Veer and Waern's (1988) description, and also Jackendoff's (1983) ontological categories.

3.1.1. Metalevel description

3.1.1.1. Ontological concepts

The ontological concepts are those enumerated in the theory of conceptual structure by Jackendoff (1983), namely object, place, etc.

```
CONCEPT
 GENERIC TYPE          = ONTOLOGICAL CATEGORY
 SUBTYPES              = ([PLACE], [PATH], [STATE], [SPACE], [EVENT])
 PROPERTIES            = ( A PAIR ([PROPERTY-NAME], [PROPERTY-VALUE]))

PLACE
 GENERIC TYPE          = ONTOLOGICAL CATEGORY
 SUPERTYPE             = CONCEPT
 REPLACEMENT RULE      = ( place.FUNCTION([OBJECT]))

SPACE
 GENERIC TYPE          = ONTOLOGICAL CATEGORY
 SUPERTYPE             = CONCEPT
 REPLACEMENT RULE      = (extend.FUNCTION([OBJECT]))
```

LEVEL
GENERIC TYPE = SPACE
SUPPORTS = ([CONCEPT], [GOAL], [METHOD])
STRUCTURE = SET-OF ([CONCEPT], [GOAL], [METHOD])
SUBTYPES = ([TASK-LEVEL], [SEMANTIC-LEVEL], [METALEVEL], [PERSONALITY-LEVEL], ...)

3.1.1.2. Cognitive acts

Now we will introduce the description of the basic cognitive acts which appear in the human thought process (Fig. 2).

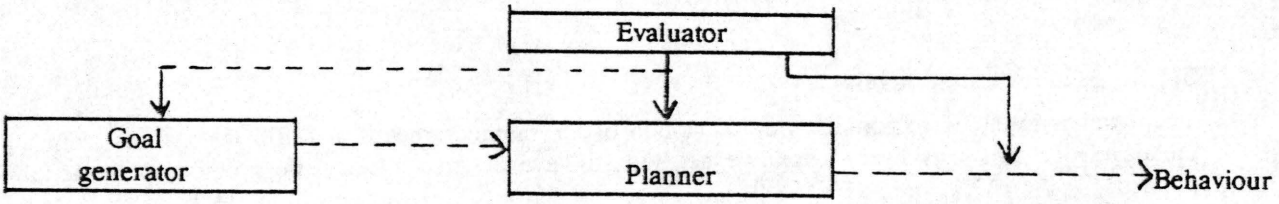


Figure 2.

It is noticeable that apart from Innocent et al.'s schema we do not have any learning mechanism. Furthermore the goal generator shifts from the physical world to the domain of cognitive acts. How do these components work together?

- the goal generator produces a desired possible state of the system;
- the planner works out a strategy to attain the desired state, being controlled step by step by the evaluator;
- the evaluator validates or refuses a goal and controls the planner's activity;
- finally, if the plan is "approved" by the evaluator it becomes behaviour.

ACTION
GENERIC TYPE = ONTOLOGICAL CATEGORY
PROPERTY = (A PAIR ([PROPERTY-NAME], [PROPERTY-VALUE])
HAS = ([A-SEMANTIC-METHOD])

SEMANTIC-METHOD
GENERIC TYPE = ONTOLOGICAL CATEGORY
SUBTYPES = ([SEMANTIC-METHOD-FOR-LEARNING], ...)

GOAL GENERATOR
GENERIC TYPE = ACTION
DESCRIPTION = ("Find a purpose to be achieved")
PRECONDITION = ([ACTUAL-STATE], [EURISTIC-BASE])
RESULT = ([GOAL])
SIDE-EFFECT = ([NONE])
INFLUENCED-BY = ([PERSONALITY])

PLANNER
GENERIC TYPE = ACTION
DESCRIPTION = ("Find a path to achieve a goal")
PRECONDITION = ([GOAL])
RESULT = ([PLAN])
SIDE-EFFECT = ([NONE])

EVALUATOR
GENERIC TYPE = ACTION
PRECONDITION = ([PLAN])
RESULT = ([PLAN])
INFLUENCED-BY = ([PERSONALITY])
SIDE-EFFECT = ([NONE])

3.1.2. Semantic metalevel

This level is intended to contain the description of the methods meant to fulfill the actions from the metalevel. In this way every cognitive act should have at least a semantic method to be accomplished.

3.1.2.1. Methods

The semantic method for the evaluator consists of a so called "belief system". This type of system simulates the way people relate beliefs, how beliefs work and the way they influence thought.

BELIEF-SYSTEM
GENERIC TYPE = ACTION
DESCRIPTION = ("Belief systems are not consensual, they
simulate how we reason with beliefs, how
beliefs function as prejudices, and how
emotions affect reasoning")
PRECONDITION = ([MASTER-SCRIPT], [POSSIBLE-GOAL])
RESULT = ([GOAL])
METHOD-FOR = ([GOAL-GENERATOR])
DO = (FOR ALL [POSSIBLE-GOAL] DO
(COMPUTE [PRIORITY] ACCORDING TO [MASTER-SCRIPT])
(SELECT A [GOAL] WITH HIGER [PRIORITY])
SIDE-EFFECT = ([POSSIBLE-GOAL])

It is obvious that any computer system will imply a certain planning activity from behalf of the user, because running a complex task forces the user to take decisions about the priority of subtasks, and about resources. Even simple tasks, if they offer the possibility to decide on what next action should be undergone, induce a certain activity of planning. HCI generates specific conditions for the human behaviour, because of the computer's nature which takes part in the interaction (Young, Green & Simon, 1989). For Young, Green, and Simon the following factors should be considered:

- a HCI environment must be safe. Trying different variants in such an environment must not cause unrecoverable or expensive errors. Modern software should allow certain safety facilities, such as an UNDO option which will permit the user to get back to a former state if an action causes an undesired effect.

- the nature of an interactive environment qualifies the user's behaviour as a situated action. That means that the mental and physical behaviour of the user should be considered as an interrelation between a goal oriented action and the actual state of the physical and functional environment, including the computer.

We tried to cover these requirements by a continuous interdependency between the goal generator, planner and evaluator.

```
HIERARCHICAL-PLANNING
 GENERIC TYPE          = ACTION
 DESCRIPTION           = ("The majority of plans have nested subgoal
                         structures")
 PRECONDITION          = ([GOAL], [ACTUAL-STATE])
 RESULT                = ([PLAN])
 METHOD-FOR            = ([PLANNER])
 DO                    = (REPEAT
                            (EXPAND [GOAL])
                            (RESOLVE CONFLICTS)
                            (ELIMINATE-REDUNDANT-PRECONDITIONS)
                            (USE-EXISTING-OBJECTS)
                         UNTIL (FAILS EXPAND))
 SIDE-EFFECT           = ([NONE])

EVAL
 GENERIC TYPE          = ACTION
 DESCRIPTION           = ("Evaluates the desirability of the plan")
 PRECONDITION          = ([PLAN], [ACTUAL-STATE], [GOAL])
 RESULT                = ([PLAN])
 METHOD-FOR            = ([EVALUATOR])
 DO                    = (FOR ALL [ELEMENTARY-ACTION] IN [PLAN] DO
                            (COMPUTE [PRIORITY] ACCORDING TO [MASTER-SCRIPT]))
 SIDE-EFFECT           = ([PLAN])
```

3.1.2.2. Conceptual entities

The concepts used by the semantic methods will be defined at this point.

```
STATE
 GENERIC-TYPE   = SPACE
 DESCRIPTION    = ("A description of the objects and their places
                   at a certain moment")
 SUBTYPE        = ([GOAL], [POSSIBLE-GOAL], [ACTUAL-STATE])
 STRUCTURE      = ( SET OF ([OBJECT], [PLACE]))

POSSIBLE-GOAL
 GENERIC TYPE   = SPACE
 DESCRIPTION    = (" A node in the space of possible space")
 SUPERTYPE      = ([STATE])
```

ACTUAL-STATE
GENERIC TYPE = SPACE
DESCRIPTION = (" The present state of the world")
SUPERTYPE = ([STATE])

The entities described above (state, possible goal, actual state) are used in common by the goal generator, planner and evaluator.

The belief system needs a particular entity in order to fulfill its mission. The Master-script is a kind of personal pattern which contains statements referring to an individual's beliefs. It is one of the critical points of our system. For a certain user the Master-script will be formed according to the type of personality the user exhibits. That is, to form the Master-script we should combine the actual descriptions of each trait that composes the user's personality. The descriptions of traits are given in the personality component. For better understanding we will give an idea about the system's way of doing these things:

- first it should determine the structure of the user's personality;
- it should choose among the descriptions from the personality component those which best fits the user;
- it will compose a Master-script by combining the chosen descriptions;
- it will use the Master-script as a pattern for the belief system.

The Master-script is formed of:

- implicational molecules with the following form:
  [A does X, X causes Y, A wants Y]
- themes, which represent an interaction between the goals and the plans of two autonomous actors. They constitute themselves in a taxonomy of themes (Table 1).

Feelings toward other	Influence of an actor over the other		
	No influence	One influences the other	Both influence one another
Positive	Admiration	Devotion Appreciation	Cooperation Love
One actor negative	Alienation Freedom	Dominance Victory	Rebellion
Both actors negative	Mutual antagonism	Oppression Law and order	Conflict

Table 1. A taxonomy of themes (Abelson, 1973)

- scripts, sequences of themes which come one after another in a psychological plausible way.

These more general motions that form together the master-script can be described with our formalism as follows:

MASTER-SCRIPT
GENERIC TYPE = SPACE
DESCRIPTION = ("Spells out several general contingencies for the fate of the system or user. It is influenced by personality")
STRUCTURE = (SET OF ([IMPLICATIONAL-MOLECULES], [THEMES], [SCRIPT]))

IMPLICATIONAL-MOLECULE
GENERIC TYPE = OBJECT
DESCRIPTION = ("A clause related by psychological implication")
STRUCTURE = ([A does X], [X causes Y], [A wants Y])

SCRIPT
GENERIC TYPE = OBJECT
DESCRIPTION = ("Sequences of themes that follow each other in some psychologically plausible fashion. Again influenced by personality")
STRUCTURE = (SEQUENCE OF [THEME])

THEME-TAXONOMY
GENERIC TYPE = OBJECT
DESCRIPTION = ("Represents a relation between a theme and the position of the actors in direct interaction")
STRUCTURE = (SET OF ([SENTIMENT-TOWARD-OTHERS], [INFLUENCE-OF-OTHERS], [THEME])

THEME
GENERIC TYPE = OBJECT
DESCRIPTION = ("Represents an interaction of the purposes an plans of autonomous actors")
STRUCTURE = ([TRAIT])

PRIORITY
GENERIC TYPE = OBJECT
DESCRIPTION = ("A value (qualitative or quantitative) that asses a certain probability for an action to be undertaken or for a goal to be achieved")
ATTRIBUTE = ([VALUE])

PLAN
GENERIC TYPE = PATH
DESCRIPTION = ("An action-state sequence where each state enables a subsequent action until a final goal state is obtained")
STRUCTURE = (SET OF ([ACTION], [STATE]))

3.2. Metacommunication component

The actions achieved by this component are:

- the simulation of the user's behaviour while his learning of the system, in order to predict his future actions for suppling him in every moment with the knowledge he needs in his reasoning (something like "I know, that you know, that I know, that you know");
- the structuring of an actual mental model of the user (a dynamic model which changes in time).

In other words the metacommunication component builds and uses the mental model of the user. (Here by the mental model of a user we understand a data structure in the memory of a computer which mimics as closely as possible the real user's mental model). This component as described above has two functions:

- a tutoring function (teach the user about the system).
- a learning function (induces the user's knowledge about the system).

3.2.1. Learning level

3.2.1.1. Learning strategies

The learning strategies we have considered are:

- rote learning;
- inductive learning;
- deductive learning;
- learning by taking advice.

```
ROTE-LEARNING
 GENERIC TYPE          = ACTION
 DESCRIPTION           = ("Memorization")
 INFFERENCE-TYPE       = ([EXPLICIT-INSTRUCTIONS])
 RESULT                = ([CONCEPT])
 HEURISTIC-RULES       = ([LRU-HEURISTIC])

INDUCTIVE-LEARNING
 GENERIC TYPE          = ACTION
 DESCRIPTION           = ("Induces a new concept from an example")
 PRECONDITION          = (A PAIR ([CONCEPT], [EXAMPLE]))
 INFFERENCE-TYPE       = ([SPECIFIC-TO-SPECIFIC])
 SUBTYPES              = ([LEARNING-FROM-EXAMPLES])
 RESULT                = ([CONCEPT])
 SIDE-EFFECT           = (BIND ([CONCEPT], [EXAMPLE])
 HEURISTIC-RULES       = ([RULE1],[RULE2],[RULE3],[RULE4],[RULE5],
                          [RULE6], [RULE7])

DEDUCTIVE-LEARNING
 GENERIC TYPE          = ACTION
 DESCRIPTION           = ("Acquires a new concept from a context, like
                          proving a theorem)
 INFFERENCE TYPE       = ([GENERAL-TO-SPECIFIC])
 PRECONDITION          = (A [NETWORK] TO CONTAIN [CONCEPT])
 RESULT                = ([CONCEPT])
```

LEARNING-BY-ADVICE
GENERIC TYPE = ACTION
DESCRIPTION = ("Accepts advice and makes use of it. Conveys expert advice into performance")
PRECONDITION = ([REQUEST-FOR-ADVICE])
RESULT = ([CONCEPT])
INFFERENCE-TYPE = ([EXPLANATIONS-TO-SPECIFIC])

3.2.1.2. Conceptual entities

EXAMPLE
GENERIC TYPE = OBJECT
DESCRIPTION = ("Spells out a concept and several of its properties")
STRUCTURE = ([CONCEPT], [PROPERTY])

NETWORK
GENERIC TYPE = OBJECT
DESCRIPTION = ("A set of concepts that are intercorelated and their relations")
STRUCTURE = (SET OF (SEQUENCE ([CONCEPT], [RELATION], [CONCEPT])))

RELATION
GENERIC TYPE = OBJECT
DESCRIPTION = ("A description of the reference between two concepts")
STRUCTURE = ([PROPERTY])

PROPERTY
GENERIC TYPE = OBJECT
DESCRIPTION = ("A property of a concept")
STRUCTURE = ([PROPERTY-NAME], [PROPERTY-VALUE])

3.2.2. Learning operational level

At this level we will describe the semantical methods for the achievement of learning strategies.

3.2.2.1. Semantic methods

ROTE-LEARNING-METHOD
GENERIC TYPE = ACTION
PRECONDITION = ([EXAMPLE])
RESULT = ([CONCEPT])
METHOD-FOR = ([ROTE-LEARNING])
DO = (MAKE [CONCEPT] FROM [EXAMPLE])
(ASSOCIATE [USING-TIME] TO [CONCEPT])

ADVICER
GENERIC TYPE = ACTION

```
PRECONDITION    = ([REQUEST FOR ADVICE])
RESULT          = ([CONCEPT], [ACTION])
METHOD FOR      = ([LEARNING-FROM-TAKING-ADVICE])
DO              = ([INTERPRET ADVICE])
                  ([OPERATIONALIZE])
                  ([INTEGRATE])
                  ([EVALUATE])
```

3.2.2.2. Learning heuristics

The learning process is mainly a heuristic driven one. Some of the heuristics employed in this process are here to follow:

```
LRU-HEURISTIC
 GENERIC TYPE     = ACTION
 DESCRIPTION      = ("Last recently used - selective forgetting")
 PRECONDITION     = (NUMBER OF [CONCEPT] GREATER THAN
                    MAXNUMBER OF CONCEPTS)
 HEURISTIC-FOR    = ([ROTE-LEARNING])
 DO               = (IF [PRECONDITION] THEN
                    REMOVE [CONCEPT] LAST RECENTLY USED)

RULE1
 GENERIC TYPE     = ACTION
 DESCRIPTION      = ("The dropping condition rule")
 PRECONDITION     = ([CONCEPT], [EXAMPLE])
 RESULT           = ([CONCEPT])
 METHOD-FOR       = ([INDUCTIVE-LEARNING])
 DO               = (IF [EXAMPLE] CONTRADICTS [PROPERTY] OF [CONCEPT] THEN
                    REMOVE [PROPERTY])
 SIDE-EFFECT      = ([CONCEPT])

RULE2
 GENERIC TYPE     = ACTION
 DESCRIPTION      = ("The adding alternative rule")
 PRECONDITION     = ([CONCEPT], [EXAMPLE])
 RESULT           = ([CONCEPT])
 SIDE-EFFECT      = ([CONCEPT])
 METHOD-FOR       = ([INDUCTIVE-LEARNING])
 DO               = (IF [EXAMPLE] NOT IN [PROPERTY] LIST OF [CONCEPT] THEN
                    ADD [PROPERTY] TO [CONCEPT])

RULE3
 GENERIC TYPE     = ACTION
 DESCRIPTION      = ("The extending reference rule")
 PRECONDITION     = ([CONCEPT1], [CONCEPT2], [EXAMPLE])
 DO               = (IF [EXAMPLE] RELATES [CONCEPT1] AND [CONCEPT2] THEN
                    (ADD [PROPERTY] TO [CONCEPT1]
                    ADD [PROPERTY] TO [CONCEPT2]
                    WITH [PROPERTY-NAME] [RELATES-TO]))
```

```
RULE4
 GENERIC TYPE    = ACTION
 DESCRIPTION     = ("The climbing generalization tree rule")
 PRECONDITION    = ([CONCEPT1], [CONCEPT2], [EXAMPLE])
 DO              = (IF [EXAMPLE] RELATES [CONCEPT1] TO [CONCEPT2]
                    AND [CONCEPT1] ANALOGOUS TO [CONCEPT2] THEN
                    [CONCEPT1] IS SUPERTYPE FOR [CONCEPT2]
                    OR ADD NEW [CONCEPT] SUPERTYPE FOR BOTH)
```

3.3. Personality component

This component influences all the other components by means of the master-script which will be built with pieces of information supplied by the personality level. We have chosen a model of personality based on the California Psychological Inventory (C.P.I.), because it offers an elegant way for the diagnosis of the user's personality type (Gough, 1987). A few words about this model.

Gough has developed three relatively uncorrelated vector scales combined into a single "cuboid model": the first scale (v.1) assesses the interpersonal factor; the second scale (v.2) assesses the normative factor; the v.3 scale was developed to index the level of self-realization and competence. The v.1 and v.2, taken conjointly, define four life-styles or ways of living. For convenience, these four cells have been labeled as the Alpha, Beta, Gamma and Delta. Figure 3 presents the v.1 and v.2 axes of extraversive to introversive interpersonal orientation, and norm-questioning to norm-accepting perspectives on social values. The descriptions of the four types are for persons at the highest level of fulfillment and integration (Gough, 1989).

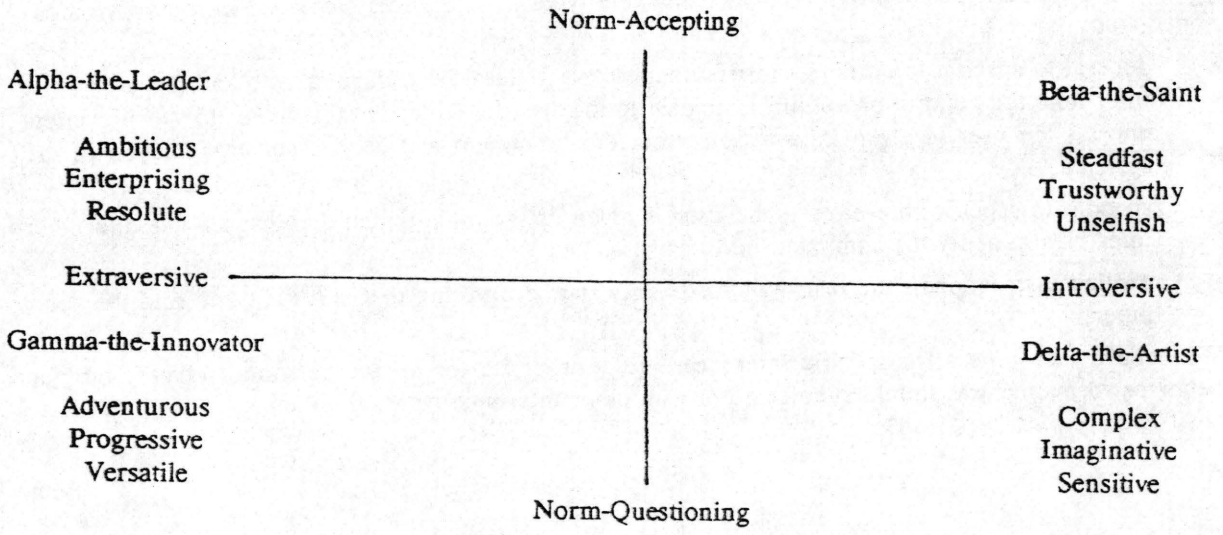


Figure 3. The typology formed by the v.1 and v.2 scales of the C.P.I. (Gough, 1989).

The findings for the third scale (v.3) are compatible with the interpretation that the scale assesses feelings of self-realization and psychological integration. Because of the minimal correlations it is possible for a person classified into any one of the four cells to score high, average or low on the third scale; that is, it is possible for any Alpha, Beta, Gamma or Delta to manifest any level of the self-realization.

Next, we made some statements about how we will use this model.

Alpha Hierarchical help on request	Beta Hierarchical help continuously displayed
Explicit help on request Gamma	Explicit help continuously displayed Delta

This classification is valid only at the beginning of the human computer interaction on a new system, when the metacommunication has the leading role. After a period of interaction the tendency is to use hierarchical help on request.

In a study developed by Black and Carroll (1989), the general to specific type of inference is proved to cause the most effective transfer of knowledge. The conclusion is statistically proved so we considered ourselves entitled to extend it. We consider that this type of inference is indeed the most effective for the learning process, but the other types can also add to the quality of learning if correlated with the user's personality. We will review the types of inferences employed in the above mentioned study:

- general to specific inferences - instructions state a general procedure or rule and the user must infer how the general procedure is applied to the specific context; explanation to specific inferences - the user is given information about the functional and conceptual organization of the system;
- specific to specific inferences - the user is given an example of the command's use and must infer how to apply the command in different context;
- explicit instructions - the instruction explicitly state everything that is to be typed into the computer.

We believe that general to specific inferences are appropriate for all the personality types, but for better performances we should supplement it with other inference types, as follows:

Alpha Explicit instructions	Beta Specific to specific inferences
Explanation to specific Explicit instructions Gamma	Explanation to specific Delta

By this assumptions we intend to influence the metacommunication, that is the learning strategies. It can be observed that we have a correspondence between the types of inference and the learning strategies defined at the metacommunication component:

- general to specific inference → deductive learning
- explanation to specific inference → inductive learning
- specific to specific inference → learning by advice
- explicit instructions → rote learning.

Certainly, the personality influences other traits of human interaction with the computer. We will try to include as much as possible in the assessments made in the formalism.

3.3.1. Personality level

This is the place were some examples will be shown of the way we write the above hints into the formalism we developed.

3.3.1.1. Personality entities

```
PERSONALITY MODEL
 GENERIC TYPE        = SPACE
 SUPPORTS            = ([PERSONALITY-TYPE])

PERSONALITY-TYPE
 GENERIC TYPE        = SPACE
 SUPPORTS            = (SET OF [TRAIT])
 SUBTYPES            = ([ALPHA], [BETA], [GAMMA], [DELTA])

ACTOR
 GENERIC TYPE        = SPACE
 ATTRIBUTES          = ([PERSONALITY-TYPE])
```

3.3.1.2. Personality strategies

The purpose of the personality strategy is to determine the type of personality the user has and to make use of this information in order to build the master-script and to select the appropriate strategies for metacommunication.

```
PERSONALITY STRATEGY
 GENERIC TYPE        = ACTION
 DESCRIPTION         = ("Selects a strategy to suit a personality type")
 PRECONDITION        = ([VALUE FOR [PERSONALITY TYPE])
 RESULT              = ([LEARNING-STRATEGY], [MASTER-SCRIPT])
```

3.3.2. Operative personality level

According to Gough (1987) we have affected four different types of psychological organization of personality. Here are their transcriptions in formalism.

```
USER
 GENERIC TYPE          = SPACE
 SUPERTYPE             = ([ACTOR])
 INSTANCES             = ([PERSONALITY-TYPE] ONE OF ([ALPHA],...[DELTA])

ALPHA
 GENERIC TYPE          = SPACE
 DESCRIPTION           = (" ...")
 LEARNING STRATEGY     = ([DEDUCTIVE-LEARNING],[ROTE-LEARNING])
 HELP-STRATEGY         = ([HIERARCHICAL HELP ON REQUEST])

BETA
 GENERIC TYPE          = SPACE
 DESCRIPTION           = (" ...")
 LEARNING STRATEGY     = ([DEDUCTIVE-LEARNING],[INDUCTIVE-LEARNING])
 HELP-STRATEGY-B       = ([HIERARCHICAL HELP ALL THE TIME])
 HELP-STRATEGY         = ([HIERARCHICAL HELP ON REQUEST])

GAMMA
 GENERIC TYPE          = SPACE
 DESCRIPTION           = (" ...")
 LEARNING STRATEGY     = ([DEDUCTIVE-LEARNING],[LEARNING-BY-TAKING-ADVICE])
 HELP-STRATEGY-B       = ([EXPLICIT-HELP-ON-REQUEST])
 HELP-STRATEGY         = ([HIERARCHICAL HELP ON REQUEST])

DELTA
 GENERIC TYPE          = SPACE
 DESCRIPTION           = (" ...")
 LEARNING STRATEGY     = ([DEDUCTIVE-LEARNING],[LEARNING-BY-TAKING-ADVICE])
 HELP-STRATEGY-B       = ([EXPLICIT-HELP-ALL-THE-TIME])
 HELP-STRATEGY         = ([HIERARCHICAL HELP ON REQUEST])
 MASTER-SCRIPT-ENTRY
   THEME               = ([USER] INFLUENCES [COMPUTER])
                         [APPRECIATIVE]
                         [POSITIVE])
   IMPLICATIONAL-
    MOLECULE           = ([USER] DOES [VERIFY]
                         [VERIFY] CAUSE [EVALUATE]
                         [USER] WANTS [EVALUATE])
```

We do not believe that the model we choose is the best. Yet, we think that while working on a computer the user affirms his personality, and by using this model we have tried to offer a type of interaction which is more suitable with the user way of doing things, and which will allow him to easier accommodate with a computer system. Our future applications will validate or refute these hypothesis.

We tried to induce a flexible structure to the conceptual and mental model used in HCI by enlarging the previous models with personality driven cognitive acts, that is we tried to give a human face to computer.

3.4. Conceptual component

This component has a great similarity with the synonymous one described by Moran in CLG. The conceptual and communication components are specific to the application developed. In developing them we will refer to a particular application, namely an interpreter for the C.P.I.

3.4.1. Task level

3.4.1.1. Conceptual entities

SELF-REPORT-STUDIES
GENERIC TYPE = SPACE
DESCRIPTION = ("Tools to induce some aspects of personality from the answers to a number of items/question")
SUBTYPES = ([INVENTORY], [QUESTIONNAIRE])

INVENTORY
GENERIC TYPE = SPACE
SUPERTYPE = ([SELF-REPORT-STUDIES])
EXAMPLE = ([MMPI], [EMPI], [CPI])

QUESTIONNAIRES
GENERIC TYPE = SPACE
SUPERTYPE = ([SELF-REPORT-STUDIES])
EXAMPLE = ([CATTELL-16PF])

CPI
GENERIC TYPE = SPACE
DESCRIPTION = (".....")
SUPERTYPE = ([INVENTORY])
THEMES = ([ITEM], [ANSWER-SHEET], [CHARACTERIZATION], [NORMS], [RAW-SCORES], [TRUE-SCORES], [FOLK-CONCEPT], [VALIDATION-CONCEPT], [SCALE], [FACTOR], [PROFILE], [CUBOID-MODEL], [STRUCTURAL-SCALES])

ITEM
GENERIC TYPE = OBJECT
DESCRIPTION = ("An assessment with which the subject has to say his position - whether he agrees with it or not")
ATTRIBUTE = ([ANSWER-VALUE])

ANSWER-SHEET
GENERIC TYPE = OBJECT
DESCRIPTION = ("A place where the subject marks down his answers [ANSWER-VALUE]-s to the inventory")
STRUCTURE = ([PLACE] FOR [ANSWER-VALUE])

CHARACTERIZATION
GENERIC TYPE = SPACE
DESCRIPTION = ("The purpose of CPI administration namely a characterization for a certain subject")
SUPPORTS = ([PLACE] FOR [CONCLUSION])

CONCLUSION
GENERIC TYPE = OBJECT
DESCRIPTION = ("An assessment concerning the subjects personality")
STRUCTURE = ([TEXT])

3.4.1.2. Tasks

ADMINISTRATION
GENERIC TYPE = ACTION
USES = ([ITEM])
PRECONDITION = ([NONE])
GOAL = (FILL [ANSWER-SHEET])

INTERPRETATION
GENERIC TYPE = ACTION
USES = ([CUBOID-MODEL], [PROFILE], [KNOWLEDGE])
PRECONDITION = ([CUBOID-MODEL], [PROFILE])
GOAL = ([CHARACTERIZATION])

KNOWLEDGE-ACQUISITION
GENERIC TYPE = ACTION
USES = ([PROFILE], [KNOWLEDGE])
PRECONDITION = ([PROFILE])
GOAL = (FILL [KNOWLEDGE])

There are other two tasks: SCORING and VALIDATION, but it is not our purpose to give a complete description of the application.

3.4.2. Semantic level

According to Moran the tasks described at task level should be mapped on the semantic method responsible for their fulfillment. The same holds true for us.

3.4.2.1. Conceptual entities

KNOWLEDGE-BASE
GENERIC TYPE = SPACE
MAPPED FROM = ([KNOWLEDGE])
STRUCTURE = (SET OF [RULE])

RULE
GENERIC TYPE = OBJECT
STRUCTURE = (SEQUENCE ([R-CONDITION], [R-CONTEXT], [R-CONCLUSION]))

3.4.2.2. Conceptual operations

```
MODIFY
 GENERIC TYPE    = ACTION
 PARAMETER       = ([RULE])
 SIDE-EFFECT     = ([KNOWLEDGE-BASE])
 EFFECT          = (DO
                    (REMOVE [RULE] FROM [KNOWLEDGE-BASE])
                    (CHANGE STRUCTURE OF [RULE])
                    (INSERT [RULE] IN [KNOWLEDGE-BASE]))

INVALID
 GENERIC TYPE    = ACTION
 PARAMETER       = ([RULE])
 SIDE-EFFECT     = ([KNOWLEDGE-BASE])
 EFFECT          = (REMOVE [RULE] FROM [KNOWLEDGE-BASE])
```

3.4.2.3. Semantic methods

```
RULE-EDITOR
 GENERIC TYPE                = ACTION
 PRECONDITION                = ([PROFILE])
 RESULT                      = (A NEW [RULE])
 METHOD-FOR                  = ([KNOWLEDGE-ACQUISITION])
 DO                          = (REPEAT (CASE (MODIFY [RULE])
                                   (INVALID [RULE])
                                   (SEE-NEXT [RULE]))
   UNTIL (NO-MORE [RULE]))
 SIDE-EFFECT                 = ([KNOWLEDGE-BASE])
```

3.5. Communication component

3.5.1. Syntactic level

In contrast to CLG where each task is mapped on a syntactic method, we have mapped the conceptual operations from the semantic level; that is, we have a hierarchy of the following type:

- tasks map on semantic methods;
  - semantic methods decompose into conceptual operation;
    - conceptual operation map on syntactic methods.

3.5.1.1. Syntactic operations

```
MODIFY3
 GENERIC TYPE    = ACTION
 PARAMETER       = ([TEXT])
 EFFECT          = (EDIT [TEXT])
```

3.5.1.2. Syntactic methods

```
MODI
 GENERIC TYPE    = ACTION
 METHOD-FOR      = ([MODIFY])
 CONTEXT         = ([RULE-EDITOR])
 PRECONDITION    = ([RULE])
 EFFECT          = (REPEAT (CASE (MODIFY1 [R-CONDITION])
                           (MODIFY2 [R-CONTEXT])
                           (MODIFY3 [R-CONCLUSION])
                           (QUIT))
                   UNTIL (QUIT))
```

3.5.2. Interaction level

At this level we introduce a new concept: the interactor.

3.5.2.1. Interactors

An interactor (Myers, 1989) incapsulates interactive behaviours. It is based on the observation that there are relatively a few interactive behaviours in a graphic interface environment. These interactors cover almost all of the interaction types excluding those for text editing and those based on commands. Future work will offer a possibility to include these types of interactions in the class of interactors. For the time being a description like the one used in CLG suits our purpose.

```
TEXT-EDITING-INTERACTOR
 GENERIC TYPE    = ACTION
 DESCRIPTION     = ("This is used to input a one line edited text")
 DO              = (PRESS [KEY] ACCORDING TO [TEXT])

MENU-INTERACTOR
 GENERIC TYPE    = ACTION
 DESCRIPTION     = ("This is used for choosing one or more from a
                    set of items. A feedback object is usually
                    shown over the selected item")
 ITEM-SET        = (SET OF [VARIANT])
 MOVING-KEYS     = ([^], [v])
 SELECTION-KEY   = ([<CR>]

VARIANT
 GENERIC TYPE    = OBJECT
 DESCRIPTION     = ("A text/icon explaining the action undertaken")
 STRUCTURE       = ([TEXT], [ICON])
```

3.5.2.2. Interactive methods

```
CASE
 GENERIC TYPE    = ACTION
 SUPERTYPE       = ([MENU-INTERACTOR])
 ITEM-SET        = (("Modify condition")
                    ("Modify context")
                    ("Modify conclusion"))

EDIT
 GENERIC TYPE    = ACTION
 SUPERTYPE       = ([TEXT-EDITING-INTERACTOR])
```

Having these interactors a task can map through the semantic and syntactic level to a multitude of sequences of interactors. It is up to the system to decide which sequence is better suited for a certain type of personality, or at a certain time in the communication with the user.

4. Conclusions

The mental model we presented tried to cover a greater area of cognition than the previous ones. It includes a series of general traits exhibited by the human way of reasoning. Its core could be used as a basis for developing a great number of applications. The personality model is central for the dynamic behaviour of the system. The use of a more sensitive one could bring surprising performance.

The continuous refinement and improvement of the system's core is a task we will undergone in time, hoping it is worthwhile.

5. References

Black, J.B., Bechtold, J.S., Mitrani, M., Carroll, J.M. (1989). Online tutorials: what kind of inference leads to the most effective learning? CHI'89, Wings for the Mind. Conference Proceedings, Austin, Texas, April 30 - May 4, pp. 81-83.

Boring, E. (1929). History of Experimental Psychology. The Century Co., New York.

Bunge, M., Ardila, C. (1987). Philosophy of Psychology. New York, pp. 275.

Gough, H.G. (1987). CPI Administrator's Guide. Consulting Psychologists Press, Palo Alto, CA.

Gough, H.G. (1989). The California Psychological Inventory. In: Major Psychological Assessment Instruments. C.S. Newmark (Ed.), Vol. 2, Boston: Allyn and Bacon, pp. 67-98.

Hoc, J.M. (1989). What is Cognitive Ergonomics? EACE Newsletter, 3, pp. 2-5.

Innocent, P.R., Veer, G.C. van der, Waern, Y. (1988). Experiments with the user interface for UNIX mail. Conference Proceedings EUUG September 1987, Trinity College, Dublin, Ireland. Buntingford, U.K.; European UNIX systems User Group.

Jackendoff, R. (1983). Semantics and Cognition. MIT-Press, Cambridge Mass.

Miclea, M., Radu, I. (1989). Modelarea in sistemul psihologiei cognitive. Revista de psihologie, 35, 1, pp. 5-15.

Moran, T.P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. International journal of Man-Machine Studies, 15, pp. 3-50.

Myers, B.A. (1989). Encapsulating interactive behaviours. CHI'89, Wings for the MIND.

Conference Proceedings, Austin, Texas, April 30 - May 4, pp. 319-324.

Payne, S.J., Green, T.R.G. (1986). Task-action grammars: a model of the mental representation of the task languages. Human Computer Interaction, 2, 2, 93-133.

Piaget, J., (1960). Les modeles abstrats sont-ils opposes aux interpretation psycho-physiologique dans l'explication en psychologie? Revue suisse de Psychologie Pure et Appliquee, 1.

Stagner, R. (1988). A History of Psychological Theories. MacMillan Publishing Company, New York.

Tauber, M.J. (1988). On mental models and the user interface. In: Working with Computers. Computer and People Series. G.C. van der Veer et al (Eds.). Academic Press: London.

Veer, G.C. van der, Guest, S., Haselager, P., Innocent, P., McDaid, E., Oestreicher, L., Tauber, J., Vos, U., Waern, Y. (1990). Designing for the mental model. An interdisciplinary approach to the definition of the user interface for the electronic mail system. In: Mental Models and Human Computer Interaction 1. D. Ackermann and M.J. Tauber (Eds.), Elsevier Science Publishers B.V. (North-Holland).

Young, R.M., Green, T.R.G., Simon, T. (1989). Programmable user models for the predictive evaluation of interface designs. CHI'89, Wings for the Mind. Conference Proceedings, Austin, Texas, April 30 - May 4, pp. 15-19.

An object-oriented semantic description of procedures for evaluation of interfaces

Poitrenaud, S., Richard, J.F. & C. A. Tijus

Laboratoire de Psychologie Cognitive du Traitement de l'Information Symbolique
Université de Paris 8
93326 Saint Denis Cedex 02

This paper addresses the central issue of evaluating interfaces with a new approach of declarative and procedural knowledge representation. We demonstrate how knowledge involved in using a system can be represented into semantic networks (where classes factorize common structure of objects through associated procedures with simple and multiple inheritance principles). In such semantic networks, objects are instances of classes and subclasses represented as nodes of a directed graph where the vertices indicate an inclusion relation structure which is defined by the procedures (considered as properties of classes of objects) to which they may be applied. From the analysis of two different technical devices, a new telephone terminal and a text-editor, the cognitive relevance of our approach as a model for human knowledge representation is shown (i) by representing and simulating within this model the behavior of subjects learning elementary devices functions and (ii) by analyzing the "cognitive complexity" of a system. For a device and a novice, comparing and evaluating graphs or subgraphs isomorphism of semantic networks provide a good prediction of learning difficulties. For two given devices, a good prediction of differences in cognitive complexity is provided as well.

Course of learning is described by the enrichment (construction of a new class, addition of a procedure to a class) and the restructuring (displacement of a class, transfer of a procedure from a class to a superclass or the reverse) of a semantic network that is first related to a familiar device which stands as a source for analogical transfer and learning appears to be the progressive construction of a semantic network which is consistent with the device to be used.

Cognitive complexity of a system is described by its semantic network structure. Experimental results show that simplifying the semantic network structure of a system makes the new version of this system more usable in reducing its cognitive complexity.

1. Introduction

The paper presents how the knowledge involved in operating a device and how the knowledge of how to accomplish the task can be both represented in a same formulation using an object-oriented knowledge-based system called Procope in which the hierarchical structure of procedures, which is the basis for the decomposition of goals into subgoals up to the level of primitive actions, is transposed into networks that describe the semantic structure of the objects of the microworld in which the procedures are valid. The hierarchy of decomposition distinguishes four levels: tasks, functions, commands and primitives. The format of a procedure is the same at any level and is composed of three parts: the conditions of execution, the procedure itself and the exit conditions. The semantic network of objects is structured by the identity and the inclusion relation. Procedures are linked up to the most general class of objects to which they may be applied. Procope has been tested through the analyzing novice's knowledge of how to accomplish job-tasks (i) of typing on a text-editor and (ii) of dialing and storing telephone numbers on a new technological telephone terminal.

First, we present how to describe tasks to be accomplished on devices by presenting relationship between the functioning of the device and the task at some intermediary levels of the decomposition of goals into subgoals and we will focus on Procope's description of declarative and procedural knowledge involved in using a device. Secondly, subjects data analysis will be reported to demonstrate how our formalism enables us to simulate subjects behavior or to reduce systems complexity. Finally, we will discuss Procope methodology (i) to simulate the carrying out of tasks issued of varied semantic descriptions and their interactions with the external world and to compare the results of automatic performance with human performance (ii) to be used in the conception of "on-line aid" for effective planning and technical decision support and (iii) to be used in the design of new devices.

1. Task decomposition and the semantic structure of the objects

To describe the user's task representation, Card, Moran and Newell (1983) distinguish, by the GOMS model, four levels of analysis: goals, operators, methods and selection. Another models come from the task action-grammar approach (Reisner 1981, 1984; Payne and Green 1983, 1986; Green, Schiele & Payne, 1988). These models use a BNF formalism and describe decomposition in the form of rewriting rules. A need for introducing semantic aspects has been manifested in several ways. Green and Payne (1983) introduce metarules which concern classes of objects and make explicit similarities between rules. Kieras and Polson (1985) propose to describe the task with a system of productions and the model of the device by an Augmented Transition Network and apply this approach to the evaluation of devices. Another description based on the decomposition of procedures and applied to the conception of devices has been proposed by Scapin (1988): an object-oriented formalism is used in order to make explicit similarities between procedures (but not the structure of the objects to which procedures are applied). Tauber (1988) introduces explicitly a description of the model of the machine: decomposition of procedures into actions involves changes of state in the device, which specify which types of objects may be affected. Until now, the emphasis has been put essentially on the decomposition of goals. Usually, intermediate levels occur between the basic level of semantic propositions and the cognitive level of representation allowing to work out the procedure. This point of view has been mainly concerned by the learning of procedures from texts and the definition of text organizations adequate for learning (Frederiksen, 1975, 1989). As a difference with previous studies, our approach makes explicit the semantic structure of the objects of the domain and the selection of a procedure to complete a task is based on this structure.

The general cognitive hypothesis underlying our approach is that procedures are tied to objects, so that they are accessed through as properties of classes of objects. In other words, procedures are stored in memory in the same way as structural properties of objects in a semantic network. We consider that the availability of a procedure for a given type of object is recognized by a very early decision: it precedes the processing of how the procedure will be worked out. This hypothesis has some cognitive relevance (Richard, 1983).

Following theoretical foundations for human semantic of action (Richard, 1990) and for object-oriented languages (Goldberg & Robson, 1983), PROCOPE was designed as a computational architecture to unify declarative and procedural knowledge in semantic networks in order to provide a framework for the development of procedural knowledge-based systems. There are two basic steps in domain description. The language of description is first based on task decomposition that provides elementary procedures. Then, procedures are used to construct the semantic network of the task microworld (Objects, Procedures, Relations). The first step method is quite similar to some other task decomposition formalism that have been developed. The second step corresponds to a new approach of knowledge representation.

1.1. Framework of a decomposition of goals into subgoals

In order to achieve the first step of the application of the method, we describe procedures as follows. A procedure generates a decomposition of a goal into subgoals connected by temporal constraints: these subgoals have to be decomposed or they are primitive actions. We distinguish four levels in the task decomposition. The lowest level corresponds to the elementary actions (primitives) which one can apply to the device. The next lowest level consists of commands which regroup elementary actions. The third level consists of sequencing commands which correspond to the functionality of the device. Finally, the highest consists of user's defined tasks.

(i) Tasks: this level is that in which the actions have a meaning which is independent of the device being used (moving text, permuting two phrases).

(ii) Functions: functions are operations of the device in which several commands cooperate in the realization of a global result.

(iii) Commands: commands are requests by the user to which the system responds by particular modifications of the data or by modification of the parameters which control the data. The submission of a command implies generally the execution of a set of one or more primitives.

(iv) Primitives: These are the actions of the user which trigger the smallest event which the system can identify and process. However, the result of the primitive is never by itself a modification of data manipulated by the user. Not having a goal-related result, the goal in which it participates is the goal of the command in which it is inserted.

A procedure has a single goal and several results. The goal is a target state which one attempts to attain with an appropriate procedure. Among the results, it is necessary to distinguish between a principal result, which corresponds to the achievement of a goal, and secondary results (side effects). Several procedures can share a same goal but may differ in their side effects. On the other hand, carrying out a procedure can be considered acceptable when it is chosen either for its side effects or its role in the principal result.

We distinguish between three phases in the implementation of a procedure: verification of the conditions for the execution of the procedure (prerequisites and preconditions); execution of the procedure body and the verification of the exit conditions (post-conditions) and execution of any associated subsidiary procedures (post-requisites).

A procedure can contain prerequisites and preconditions which make reference to states enabling the execution of the procedure body. The prerequisites are states that have to be realized in order to carry out the procedure body. A prerequisite possesses its own subordinate procedure which has to be executed each time the state is not realized. Similarly, preconditions consist of a test of states. A precondition is the entry condition into the procedure. However, there is no subordinate procedure for realizing this state. Either this does not depend on the user's actions, or the utilization of the procedure makes it impossible to attain a waiting goal. When the precondition is not satisfied, one must abandon the procedure for which it is an entry condition.

Execution, strictly speaking, is described by the activation of a set of subgoals. These actions are temporally structured by a sequencing relation. We distinguish between five cases of sequencing relation: ordered execution, linked execution, linked and ordered execution, unconstrained execution and parallel execution.

After the carrying out of a procedure body, tests may have to be effectuated in order to assure that one has really attained the sought after goal and/or that no subsidiary action is required. These are the post-conditions and the post-requisites. Post-conditions are conditions for ending the procedure. They serve to evaluate the principal result. They permit, in particular, the expression of iteration conditions in the procedure body as "repeat until". The post-requisites are auxiliary actions which complete the execution of the procedure. They are not strictly necessary for the attainment of a goal. In a sense, they are what remains to be done after one has finished.

1.2. Semantic networks of object types

Procedures are applied to objects and objects may be classified as a function of the procedures which may be applied to them. To achieve the second step of the method, we propose to build a semantic network of object types to which are associated the procedures that apply to them. This network is structured by the inclusion relation. Using this relation, given the procedures available in a device for the different objects, it is possible to build a semantic network of object types in the manner of the class and method hierarchies in an object-oriented language. To do so, it is necessary to identify the different object classes of the system and bring out the general/specific relations in examining the procedures that are applicable for those classes.

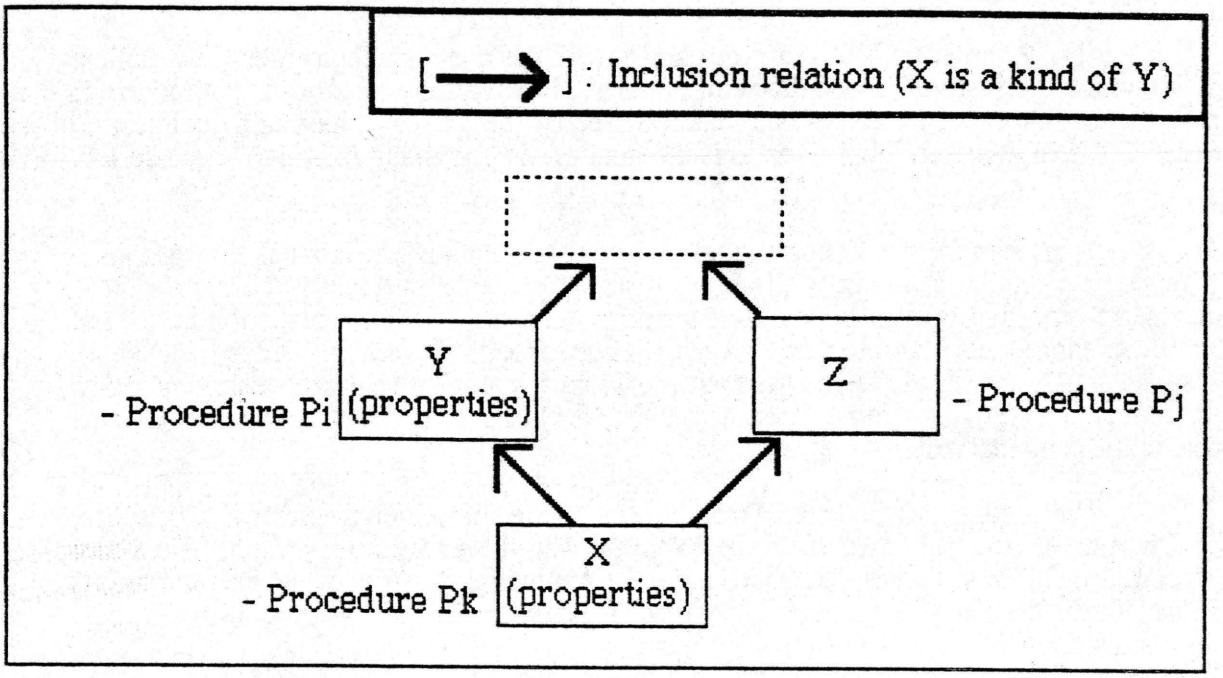


Figure 1 - Procope semantic network representation with inclusion relations. At this point, procedural & declarative semantic of objects merges in regard of applied procedures. Classes Y and Z inherite of procedures of subordinate classes as class X inherites of procedures of both Y & Z classes (multiple inheritance).

A PROCOPE microworld description is a semantic network of task objects (figure 1), in regard to the procedures applied to them. This network is structured by the identity relation ("X" and "Y" belong to the same class if all that can be done on "X" can be done on "Y", the reverse being true as well) and by the inclusion relation (" X" is a kind of "Y", if it is possible to do on "X" what is possible to do on "Y"). Classes factorize common object structures through associated procedures with simple and multiple inheritance principles.

Given the objects of a domain with associated procedures, PROCOPE produces a hierarchy of abstract and concrete classes represented as a directed graph [G: < N, P, R>] which is defined to consist of a set of nodes "N" (representing classes of objects), a set of procedures "P" and a set of inclusion relations "R" between nodes. Node n_i corresponds to a subclass label assignment of node n_j if and only if $p(n_i)$, procedures of node n_i, are included in the set of $p(n_j)$, procedures of node n_j. In such a semantic network, procedures are domain-dependant and object-specific.

1.3. Cognitive relevance of the PROCOPE description

The basic cognitive hypothesis underlying the PROCOPE description is that procedures are tied to objects, so that they are accessed through as properties of this class. In other words, procedures are stored in memory in the same way as structural properties of objects in a semantic network. We consider that the availability of a procedure for a given type of object is recognized by a very early decision : it precedes the processing of how the procedure will be worked out.

Learning to solve familiar tasks with a new device consists of two steps: learning which procedures are available for the different types of objects and learning how each procedure has to be decomposed up to the level primitive actions. In other words what has to be learned first is the semantic network of objects which is required by the new device. A direct test of this hypothesis may be made by teaching subjects procedures applied to specific objects in the new device and testing their generalization to new objects, some of them being categorized in the same way in both devices, other ones belonging to different semantic hierarchies.

A text-editor is an adequate new device to be compared to an electronic typewriter taken as the familiar device. A letter and a digit are both a character in both environments. They belong to the same class from the point of view of procedures which may be applied to them: they may be typed and deleted in the same way. So we may expect that a procedure learned to delete a letter will be transferred immediately to a digit. On the other hand, while a space is treated as a character in the environment of the text-editor, since everything which can be done on a digit or letter may be done on a space, it is not categorized in the same way in the microworld of the typewriter. It is categorized as a blank. A blank belongs to the category of blank space: blank space cannot be deleted, the only thing which can be done with it is to move things in, by changing their location. So when given a task which can be interpreted as deleting a space in the semantic network of the text-editor (to modify a word with a blank in the middle), a subject familiar with the sole environment of the typewriter will not even ask himself how to delete a blank. Hence a deletion procedure learned for a letter can not be transferred to a space.

Reducing cognitive complexity of a device can be undertaken in first describing with Procope method the semantic network one must acquired in order to carry out tasks with the device and secondly in conceiving a new device that share the same functions but where the how-to-use-it can be described in a simpler semantic network.

A new technological telephone with multiple storage functions might be simplified if the how-to-use memory locations is changed in such a way the corresponding semantic network of memory location appears to be well structured in regard to the number of classes and the simplicity of the classes and subclasses inclusion relation.

2. Experimenting Procope's method

2.1. The course of subjects learning elementary functions of text-editing

This experiment has been conducted in our laboratory (Pichancourt, 1989). Ten adult subjects familiar with electronic typewriters learned how to delete a letter in Word (by moving the cursor behind the letter, clicking and pressing the BS key), to delete a word (by a double-click inside the word and pressing the BS key) and to duplicate a word (by delimiting the word, that is to say by clicking before and shift-clicking after the word, then by selecting the copy item, moving the cursor to the insertion point and selecting the paste function in the Edition menu). Immediately after learning they were tested on the following tasks: delete a letter, a digit, a space, a word, a number and a sequence of spaces; duplicate a word, a number, a sequence of spaces, an alphanumeric sequence and a line of text. In a third phase, subjects not having transferred the learned procedures were invited to find another procedure for each task and in the case of failure received prompts from the experimenter.

Using the PROCOPE methodology, it has been possible not only to identify procedures but also, considering the different types of objects to which they are applied, to build the semantic hierarchy of objects with the procedures attached to each class. By analyzing how each task was solved in the transfer phases, it has been possible to identify two semantic organizations for the elementary objects (letter, digit, space) and six for composite ones. For simple objects the two structures are:

(i) a letter and a digit are the same type of objects (a character): they can be typed and deleted. Every subject in the test phase applied to a digit the "delete" procedure learned with a letter. A space is a different type of object: it cannot be deleted, it can only be typed. This representation corresponds exactly to the microworld of the typewriter. Five subjects out of ten showed the behavior consistent with this representation during the test phase.

(ii) a space is categorized as a character in the same way as a letter or a digit. This is manifested by the transfer of the deletion procedure to a space as well as subject's comments: "I could not imagine that a blank could be deleted in the same way as a letter, because it is a blank, and I thought that this (BS key) deleted only letters". Five subjects discovered the solution by themselves during the test phase, four subjects became aware of it only in the interview phase with the help of the experimenter and one subject did not discover it at all.

For the composite objects, the semantic structure required by the device is: a string of words or numbers, a string of alphanumeric characters, a sequence of spaces and a line of text are all strings. As such they may be delimited (by clicking before the first character, pressing the shift key and clicking the last character), deleted and duplicated. In addition an alphanumeric string may be designated by a double click. All these procedures apply to the objects considered as wholes. But as these objects are made of characters, procedures which apply to characters are available. We have identified six steps in the construction of this expert semantic network:

(i) a word is designated by a double click and deleted, a word is delimited and duplicated as in the learning phase: there is no transfer at all to other objects;
(ii) only the deletion procedure (double click and BS key) is transferred to a number;
(iii) in addition a number is delimited and duplicated;
(iv) in addition an alphanumeric string is delimited and duplicated;
(v) in addition a sequence of spaces is designated by a double click and deleted;
(vi) a sequence of spaces and a line of text is delimited and copied and is also delimited and deleted.

For each step it has been possible to find out a structure of the objects consistent with the behavior:

At step (i) there is no structure at all: each object is independent of each other.

At step (ii) words and numbers belong to a class, which can be defined as the class of homogeneous alphanumeric strings. The procedure of designation by double clicking and deletion is attached to this class.

At step (iii) no new class is created but the procedure "delimit and duplicate", is attached to the generic class.

At step (iv) a superordinate class is linked to homogeneous alphanumeric strings: the class of alphanumeric strings. The procedure of delimit and duplicate which was attached to the class of homogeneous alphanumeric strings is transferred to this class.

At step (v) the class of homogeneous strings is created: it has as subclasses the homogeneous alphanumeric strings and the sequences of spaces. The procedure of double-click and deletion is transferred to this new class. Actually in the expert network the sequences of spaces have to be linked up to the class of strings: in general a double click does not select (designate) a sequence of spaces, it selects in addition the word before, but the spaces are selected if they are at the beginning of the line, as it was the case in one task. This caused some subjects to link up the spaces to an inappropriate class.

At step (vi) the class of strings is created with homogeneous strings, alphanumeric strings and lines of text as subclasses. To the class of strings are attached the procedures of deletion, duplicate and selection by delimitation. To the class of homogeneous strings is attached the procedure of selection by designation (double-click). Remark that at the previous steps, selection by "delimitation and duplication" was considered as a single procedure, as well as "selection by designation and deletion". Step (vi) is close to the expert level. There are two differences: the procedure for selection by designation (double-click) is applied at the expert level to both alphanumeric strings and homogeneous strings classes, the class of sequences of spaces is directly linked up to strings.

Considering the semantic network of elementary objects and the semantic network of composite objects, we simulate the behavior of each subject in the test phase. It was possible to assign every subject, except two, to a given semantic network for simple and complex objects in such a way that the structure of these two networks predicts exactly the different procedures which have been used by the subject. The first exception is a subject who was unable to recall the learned procedure to copy a word. The second exception is a subject whose behavior is exactly represented by a graph which has the same classes as graph (vi), the same procedures associated to the classes (in particular the procedures are dissociated into functions) with one exception: there is no link between lines of text and the class of strings. These exceptions are minor ones, so that we may consider that the behavior of subjects in the course of learning is adequately represented by the six graphs.

This analysis was similarly applied to the results of the third phase in which the subjects were invited to find other procedures and, in case of failure, received prompts from the experimenter who rephrased the task by explicating the goal associated to the procedure to be discovered: for instance subjects who tried to displace a text in order to eliminate a space in the middle of a word were said "you have to delete the space". There was two exceptions in the assignment of subjects to graphs: two subjects have discovered how to copy a sequence of spaces but have not discovered how to delete a sequence of spaces. If they were at step (v), they should be able to do both; if they were at step (iv) they should not be able to copy a sequence of spaces.

The consideration of response latency provides additional information about the elaboration of the semantic network consistent with the new environment. If we consider the transfer tasks to elementary objects, the mean response time in the task requiring the deletion of a character is very close for letters (7.81 sec.) and digits (7.28 sec.) and much larger for spaces (22.2 sec.). There is no difference whether there is one or several characters of the same type (repetition of the procedure). If we consider the task of reproducing a text (by duplication), we observe again that there is no difference between a word (12.5 sec.) and a number (13.15 sec.) and that there is a large difference for a sequence of spaces (29.6 sec.). The class of strings of words or numbers is already existing (except for one subject) from the beginning of the test, as it was the case for letters and digits.

The overall results show that learning the basic procedures in text editing is adequately described by the acquisition of two semantic structures in a sequence of orderly steps. Each new step is characterized on the behavioral side by the addition of a new procedure and on the cognitive side either by the enrichment of the network (creation of a new class) or reorganization of the network (displacement of a class), addition of procedures to a class or transfer of a procedure from a class to another one. This illustrates the notion of learning by accretion and learning by restructuring developed by Rumelhart and Norman (1978). There are very few discrepancies with this order and only minor discrepancies between the predictions of the model and observed behavior.

2.2. Reducing cognitive complexity of a device

This second experiment was designed to show that it might be possible to provide ergonomic specification in the design of devices-users interfaces or to evaluate the cognitive complexity of two given devices with same functions. The experiment has been conducted in our laboratory using our software FRIDA that allow to model on a computer the functioning and the interface of various kind of devices. Furthermore, actions on the device with screen localization and response latency are automatically recorded using FRIDA.

Starting first with an existing new technological telephone terminal R.N.I.S. that was presented to subjects on the computer screen, the declarative and procedural Procope semantic network that describes the how-to-use knowledge to perform job-tasks on the telephone has been represented. Secondly, we reduced the cognitive complexity of the telephone in regard of how-to-use memory function with the two following principles: (i) reducing the number of classes in assigning similar procedure to objects that share same properties and (ii) avoiding multiple inheritance from superordinate classes make the system more usable. Finally, a new conception of the telephone with the same functions and capabilities has been designed with FRIDA software.

Sixteen subjects were divided in two groups. Each group performed the same job-tasks on one of the two versions of the simulated device with the corresponding reference documentation of how-to-use it: Group 1 with the existing telephone terminal and Group 2 with the ergonomic conception. Results show that both errors analysis (mean of 11 versus 16 errors) and latency to perform the task (mean of 48 mn versus 67 mn) were compatible with the hypothesis that cognitive complexity is reduced when the semantic network is restructured with these two principles.

Furthermore, analyzing performance of group 1 with Procope's method in order to describe novice knowledge of how-to-use the device showed that novice subjects have a very simple and interesting representation of a sophisticated telephonic device. Subjects semantic network of a device might also be used to conceive ergonomic specifications of interfaces.

3. Conclusion

These results are a proof of the importance of considering the semantic structure underlying the available procedures and of developing a formalism which allows a systematic investigation of this structure through analysis of procedures available for different types of objects in transfer tasks.

With PROCOPE formalism, description for carrying out of procedures allows: to track learning with assimilation of new information that might change the network structure, to compare and evaluate graphs or subgraphs isomorphism (for instance between expert and novice or between devices) and to evaluate what has to be learned in order to apply correct procedures to objects, given that procedure applicability is justified by the properties of objects. As well, PROCOPE formalism allows: to simulate the carrying out of tasks issued of described procedures and their interactions with the external world and to compare the results of automatic performance issued from various semantic networks (Cerf & al, 1990). Thus, knowledge representation underlying PROCOPE seems suitable to be used in devices evaluation and conception of "on-line aid" for effective planning and technical decision support.

Evaluation of a device from the point of view of its compatibility with the mental model of the user by a comparison of the semantic network of objects required by the device, and of the semantic structure of these same objects in the familiar environment.

Documentation about a device from the procedures to be used and from the point of view of the subjects representation of how-to-use the device and consequently to interpret the actions of the user. The description of procedures by the PROCOPE formalism articulates both types of information : documentation and on-line assistance to the user. The Procope description contains both types of information and in addition provides a tool for making a diagnosis of the subject's goal and of the subject's knowledge about the device. The diagnosis of subject's goal at a given moment consists in generating hypotheses about what command, what function, what task is intended by the user and in testing these hypotheses. The generation of hypotheses is based on the representation of the user's knowledge.

References

CARD, S.K., MORAN, T.P., NEWELL A. (1983). The Psychology of Human-Computer Interaction. Hillsdale, N.J.: Erlbaum

CERF, M., POITRENAUD, S., RICHARD, J.F., SEBILLOTTE, M., TIJUS, C.A. (1990). Décisions et imprécisions dans la conduite des systèmes de cultures: la conception ergonomique des objets agronomiques. 3ème conférence IPMU. ENSTA. Paris.

FREDERIKSEN, C. H. (1975). Representing logical and semantic structure of knowledge acquired from discourse. Cognitive Pssychology, 7, 371-458

FREDERIKSEN, C. H. (1988). The representation of procedures: Acquisition and Application of Procedural Knowledge, Actes du Colloque International "Informatique Cognitive des Organisations". ICO 89. Québec. 12-15 juin 1989

GOLDBERG, A., ROBSON, D. (1983). Smalltalk-80, the language and its implementation, Reading Mass. Addison-Wesley

GREEN, T.R.G. (1980). Programming as a cognitive activity in Human Interaction with computers SMITH H.T., GREEN T.R.G. (eds). London. Academic Press

GREEN, T.R.G., PAYNE S.J. (1984). Organization and learnability in computer languages. International Journal of Man-Machine Studies, 21(1) 7-18

GREEN, T.R.G., SCHIELE F., PAYNE S.J. (1988). Formalisable models of user knowledge in human-computer interaction in Working with Computers: Theory versus Outcome VAN DER VEER G.C., GREEN T.R.G., HOC J.M., MURRAY D.M. (eds) Londres Academic Press
KIERAS, D.E., POLSON, P. (1985). An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22 (4) 365-394
NORMAN, D.A., RUMELHART, D.E. (1975). Memory and knowledge. In
NORMAN, D.A., RUMELHART, D.E. (1975). Explorations in cognition, San Francisco, Freeman and Co.
PAYNE, S.J., GREEN, T.R.G. (1986). Task action grammars: a model of the mental representation of task languages, Human Computer Interaction 2, 93-133
POITRENAUD, S., RICHARD J.F., DENHIERE G. (1989). La description des procédures: une approche "orientée objets". Actes du Colloque International "Informatique Cognitive des Organisations". ICO 89. Québec 12-15 juin 1989
REISNER, P. (1981). Formal grammar and human factors design of an interactive graphics system. IEEE Trans. Software Engineering, SE-7 (2), 229-240
REISNER, P. (1984). Formal grammar as a tool for analyzing ease of use: Some fundamental concepts, Human Factors in Computer Systems, Thomas J., Schneider M. (eds.) Norwood, N.H.:Ablex
SCAPIN, D. (1988). Vers des outils formels de description des tâches de bureau orientés conception d'interfaces. Rapport de recherche INRIA N¡ 893.
TAUBER, M. (1988). On mental models and the user interface. Human-computer interaction in Working with Computers: Theory versus Outcome VAN DER VEER G.C., GREEN, T.R.G., HOC J.M., MURRAY, D.M. (eds) London. Academic Press

Computational models for visual communication in multi-media discourse.

Alistair Sutcliffe,

Department of Business Systems Analysis,
City University,
Northampton Square,
London EC1V 0HB,
U.K.

Phone +44-71-253-4399 ext 3420

Fax +44-71-250-0837

E Mail sf328@uk.ac.city

Topic area: Frames of reference for HCI design.

1. Introduction

In multi-media interfaces it is becoming increasing apparent that visual interaction is poorly understood compared with the linguistic medium. Although the cognition of image comprehension must await further theoretical developments, more pragmatic theories are required to explain how we interact with images. This paper reports progress towards development of computational models of visual discourse in Esprit Basic Research Action 3160, Interactive Dialogues for Explanation and Learning. The motivation for this work was to create computational processes which could manage human computer discourse containing images for explanatory and tutorial systems.

Images have been used extensively for explanation and in tutoring systems either as concrete representations of the domain (e.g. STEAMER Hollans et al 1984), or to present abstract views of knowledge (GUIDON-WATCH Richer and Clancey 1985). Some systems have incorporated simple rules for using images as in Cawsey's (1988,1990) system for teaching circuit design. However, previous systems have not embodied models for selecting between a variety of images and no computational models for managing interaction with images have been constructed. This study intends to create computational models to enable a more flexible and sophisticated use of visual media.

It is not the purpose of the research to investigate how images should be designed, as this concerns principles of graphical design (see Cleveland 1985, Marcus 1987); instead the research concentrated on how to select preformed images to explain specific question types. The syntactics of image processing are assumed, since the focus of investigation is on visual semantics. Explanation also entails visual interaction with the image by highlighting, pointing, etc., to draw attention to specific features.

The problem investigated is control of properties of an image during a dialogue rather than trying to understand diectic reference in questions about images, as this has been extensively researched in other projects (e.g. Klein 1987, Lee et al 1990). The use of images for explanation, however, is unlikely to occur in isolation, hence the problem of when to select visual as opposed to verbal media had to be addressed. Finally interaction with images has to be integrated with verbal discourse, so the user system dialogue can interleave image and verbal explanation.

2. Approach.

Although the form of images was assumed their usage had to be studied so that rules and heuristics could be devised for linking types of illustration to different explanatory functions. The approach taken was to investigate how designers had used visual illustration in a wide variety of sources (e.g. illustration in

books, instruction manuals, graphic and icon instructions, and messages in everyday situations, and use of images by domain experts during explanations). A taxonomy was created to categorise the purpose of visual illustrations. Each visual illustration category was related to one or more intended explanatory purposes. The taxonomic approach created data for formulating rules for selection of image types according to the type of question being asked.

Matching image types to explanatory tasks necessitates understanding the question. Classifications of queries have been reported by Sarantinos and Johnson (1990), and verbal mediated explanation based on Rhetorical Structure Theory has been categorised by Maybury (1990). However, these investigations had not investigated questions with explicit visio-spatial referents. Consequently a second theme adopted query classification frameworks and used them to analyse questions containing explicit visio-spatial reference.

A third theme investigated a spatial model of image components so that a dialogue could be carried out with respect to it. This was motivated by the need to understand what feature in an image to highlight/hide/enlarge/animate etc. A formal model of the visio spatial semantics of images was defined so that computational processes could manipulate image components given linguistically based spatial references. Once a question had been classified, its content had to be extracted and submitted as an information request to the domain knowledge base. Consequently the spatial image model had to be mapped to the domain knowledge model, in this case Task Knowledge Structures (Johnson et al 1988) so that image components could be related to the content of questions.

The main theme of the investigation was use of visual media for delivering explanations. Interpreting linguistic reference to locations was not studied as this introduces further complexity of diectic reference. Our main concern was appropriate choice of visual media and flexible, responsive interaction with images during explanation.

3. Results

Brief results are reported for each of the three themes. For further details the reader is referred to IDEAL (1989).

3.1 Taxonomy of Visual Explanation

Images and visual illustrations from a variety of sources (e.g. encyclopaedias, instruction manuals, visual instructions and signs) were analysed in several domains. The taxonomy is based on two dimensions: a categorisation of image types and the illustration paradigms used within an image for a specific explanatory purpose.

Images were divided into two classes. Conventioned images (diagrams, maps, and other symbolic forms) in which the rules for

interpretation had to be learned and non- conventioned images (pictures and natural images) for which interpretation rules could be assumed as part of general knowledge. Within conventioned images, sub categories are recognised for:

(i) Diagrams - agreed convention; in which the semantics are assumed to be learned and generally understood by a prospective user population, e.g. Line graphs, Pie Charts, Histograms.

(ii) Diagrams - inferred convention; which do not have such generally well known semantics. Meaning can be generated by most users from general knowledge and reasoning about the context of the image, possibly augmented by a small amount of explanatory text. Examples are Hierarchy, Network and Venn diagrams.

(iii) Diagrams - specialised convention; in which meaning is available only to a restricted user population who have learned the link between diagram syntax and semantics in a restricted domain, e.g. VLSI diagrams, Optical lens diagrams, Dance notations

Within any diagram the primitive component is referred to as a symbol. Symbols may also be used independently in visual explanation to mark diagram features for the reader's attention or to illustrate directionality and sequence. Obviously the number of symbols is potentially very large so an exhaustive classification was not undertaken.

Text is classified as a primitive component of conventioned images. It seemed reasonable to classify text directly annotated onto images as labels, and other explanatory text associated with the image as captions, headings and instructions. The content of the text was not of prime interest.

Images may be composed of sub images which are recognized either by a distinct boundary of space or by virtue of recognition of objects which group together to form a coherent whole. The taxonomy therefore attempts to describe how these sub images have been organised to convey knowledge and inter-relationships between images. How sub images have been (apparently intentionally) organised, is categorised as follows:

* Linear Sequence- sub-images are placed in a sequence. The most common example of this is a series of pictures as snap shots of states to explain a task sequence.

* Hierarchy- images placed in a vertical relationship to express composition, or some other hierarchical type of relationship.

* Network- images placed in a spatial relationship more complex than simple linearity.

Images are then classified into the paradigms adopted by the image designer for specific visual explanation purposes, e.g.

* Area/Volume... Value: The size of an image component is used to express a value in the knowledge. This can be used with simple shapes, e.g. different sized boxes or circles to illustrate populations of cities.

* Symbol-Arrow... Directionality: Arrows are the most commonly used means of coding directionality or sequentiality. They may also be used to mark image components and code connectivity.

* Symbol-!mark... Warning: Exclamation marks are used to draw attention to part of an image for warning purposes. A large number of similar symbolic encodings of attention can be found in illustrations.

* Contour/Boundary...Inclusion, Set Membership: inclusion within a boundary is used to express composition and set membership.

* Exploded image.... Composition: parts of a structure are depicted in an exploded form to show how a structure is composed of its constituent parts.

* Cut away..... Composition: parts of a picture are cut away to reveal underlying structures, often used to illustrate structural composition of objects.

* 3D perspective... Value; objects placed in a 3D perspective are usually interpreted in terms of distance; however, this visual metaphor can be used to code time and other properties.

Figure 1 show an image annotated to illustrate the taxonomic analysis. Some illustration paradigms may be used for several different purposes as their visual form may code a variety of meanings (e.g. an arrow can code directionality, precedence, connectivity, etc), while other paradigms are more restricted. The classification makes no claim to being exhaustive. Its main purpose was to generate sufficient heuristics for associating paradigms of explanatory needs. These heuristics were devised by study of the context in which a particular paradigm was used in a manual, book, or verbal explanation.

3.2 Analysis of questions with visio-spatial reference.

Very few questions involving spatial reference were found in the corpus of explanations collected in the chosen domain of study (Electronic Mail). Spatial reference is also rare in questions analysed in other explanation studies, apart from reference to directions in "How do I get to ?" type questions. The investigation, therefore, had to proceed by intuitive analysis of possible spatial references in questions. This was helped by using a framework of theoretical semantics employing spatial reference (Jackendoff 1983).

The work of Sarantinos and Johnson (1990) was extended to handle questions with spatial reference. This taxonomy classifies

questions hierarchically according to their general purpose (e.g. Feature Analysis, Justification, Definitions) followed by more detailed analysis of the question's semantics, for instance Comparison - Equivalence. Questions are analysed by building up a tree decomposing the semantic elements and showing their inter-relationships. The question analysis taxonomy of Sarantinos and Johnson already contained categories such as Place, Contents and Order which were useful for spatial reference. Place, for instance is used for questions related to location "where is X ?". Four new categories augmented the classification:

(i) Path- questions requiring the spatial reference of one or more objects with respect to a known location. The objects are within immediate visual range, e.g. "How do I get to the station ?".... "Go over there"- pointing to the location.

(ii) Route- a path type question which requires more detailed verbal explanation of how to arrive at the location because the desired object is not in immediate visual range, e.g. "How do I get to Kings Cross ?"... "Go along this road, under the bridge and then take the first left".

(iii) Distance- questions requiring assessment either accurately or approximately of the spatial separation of an object from a known location, e.g. "How far is York from Kings Cross?" ...200 miles.

(iv) Direction- question requiring reference to NSEW coordinates, for a path, e.g. "Which way is King's Cross ?"... "Go left for 50 yards, then straight".

Questions could not be classified by the surface content alone, as information about the discourse focus and context of the questioner is needed to disambiguate many queries. A further part of the research will integrate focus information into refining query classification. Space precludes further consideration in this paper. A sample classification of a question with spatial reference is illustrated in figure 2. Most questions involved procedure, feature analysis, or concept completion in Sarantinos and Johnson's scheme, which is shown in fig 2 with the augmented categories.

So far the questions analysis could plausibly be linked to explanation types described in the taxonomy. However, an explanation machine will not be able to make any sense of a question unless it has a model of the knowledge represented in an image. A computational model needs to link the location of objects in an image to the underlying domain knowledge so that questions such "What is on the left hand side of menu ?" can be answered. This is addressed in the next section.

3.3 Formal Models of Images

Task Knowledge Structures (Johnson et al 1988) were selected as

the knowledge representation model. All images were modelled as a hierarchy of components composed of primitives, i.e. shapes which could not be decomposed without losing their meaning.

Describing the spatial relationships of image primitives could be approached in terms of coordinate geometry; however, this reinvents graphical description algorithms used in display interface management systems. The study was concerned not with the absolute physical morphology of images, but instead with the spatial knowledge which can be expressed in linguistic terms. Spatial information could have been stored in the TKS representation, but this would encounter problems with spatial descriptions from different viewpoints. It is possible to have different spatial descriptions of the same knowledge structures, for instance objects in a scene viewed from the side, above or below. Consequently spatial knowledge is separated from domain knowledge. One or more spatial image models are defined for each domain knowledge model, each image model holding a description from a specific viewpoint.

As linguistic expression had been assumed it was appropriate to look for a linguistically based formalism. Sowa's Conceptual Structures (Sowa 1984) is appealing because it provides a modelling framework with a typed semantic lexicon to describe semantic primitives, and inter alia image primitives; a set of conceptual relationships with which to associate primitives; and a syntax for constructing conceptual graphs to describe spatial structures within images.

Sowa's formalism does not have any specific spatial relationships although the some relationships can be used, e.g.

CONT links Entity:*x to Entity:*y where *x has content *y
PART links Entity:*x to Entity:*y where *x is part of *y
LOC links a set of Entities and Concepts to a Place

where *x denotes an uninstantiated instance variable.

New spatial relationships were added to express linguistic concepts such as Above, Below, Beside (left hand/right hand); Adjacent to (in contact with); Inside (contained within); Outside (without a boundary), etc. In total 6 new conceptual relationships were found to be necessary. Each relationship was formally defined following Sowa's practice, e.g.

ABOVE links Entity:*x to Entity:*y where *x is vertically displaced upwards from *y in the perspective of a set viewpoint, in an example:-

[WINDOW]<----(ABOVE)----[DOOR]

BESIDE links Entity:*x to Entity:*y where *x is horizontally displaced to the side from *y in the perspective of a set viewpoint

[MAT]<----(BESIDE)----[CAT]

The mat is on the left hand side of the cat

The directionality in this case is visually coded by the arrow's direction. The spatial image model is composed of image primitives declared as entities in a typed lexicon, combined with spatial relationships to form a conceptual graph of the image. Image primitives are related by one or more spatial relationships. In conventioned images another relationship (JOIN) is added to encode the connection of two node-objects by an arc. Pixel level detail of the primitives is not stored as this information is assumed to be held in graphical display components of user interface management systems. The layout of the conceptual graph is designed to follow the image topology as far as possible, although this is not essential. The conceptual description of a spatial image model and its relation to absolute topology of a conventioned image is illustrated in figures 3 and 4.

Concepts of closeness and separation are difficult to formalise because these terms are used flexibly according to the scope and granularity of detail within discourse focus. Hence in an image 10 cms wide, close may be taken to be objects <1cms apart, but in a larger image "close" may take on another spatial meaning. A pragmatic approach was adopted by quantify "Close" as a ratio the inter-object distance to the whole image diameter. It is recognised that this is limited because judgement of closeness also depends on knowledge of the semantic properties of objects, i.e. two objects may be judged to be close because they belong to the same class, even though spatially they are separated.

Judgement of proximity also necessitates representation of an object's absolute location. To be accurate the area and volume of objects has to be described at the physical level. However, as this approach had been rejected, a compromise is to store area occupied by an object as grid coordinates. This approximate quantification of overall size of the object will be inadequate for objects with irregular boundaries but the compromise is expected to be adequate for conventioned images in which shapes tend to be regular.

3.4 Mapping to TKS

To generate explanations spatial knowledge has to be used in conjunction with semantic knowledge of the image. A means of mapping image components to knowledge structures is necessary. A preliminary set of mapping relations are being investigated between the objects declared in the image primitive lexicon and TKS objects. TKS objects have a list of properties, relationships to other objects and associated actions. For a message in an E-mail system, a possible mapping is as follows:

Image model

MESSAGE < ENTITY

```
     (PART)-->HEADERLINE
     (PART)-->TEXTLINE
```
A message is composed of a headerline followed by one or more textlines

```
TEXTLINE < ENTITY
```
A textline is a line containing alphanumeric characters.

```
HEADERLINE < TEXTLINE
```
A textline is the top line used to title a window

```
[HEADERLINE]<---(ABOVE)---[TEXTLINE]
```

Maps to Task Knowledge Structure in Domain Model

```
Object, Message
     informationSource: AddressList, Reply,
PreviousCorrespondence.

     header: from <self> address <self address>
             to <Address>
     body: <text>
     footer: copy to<Address>
```

A decision had to be taken about what level of description should be stored in the image model. If more detail is stored then more redundancy is built into the knowledge representation. On the other hand if the visual description is minimal there may be spatial references which can not be processed. For instance reference to the location of "copy to" field in the message is not explicitly known in the image model so the system can only reply that it is "in the body of the message". There is no satisfactory answer to this problem, as the level of spatial detail in questions can not be totally anticipated. The representation of spatial detail has to be specified according to the domain. In E-mail only high level spatial knowledge is necessary to answer questions about a screen layouts or a network diagram. However, in a geographic information system a more refined level of detail would be stored.

4. Specification of the Computational Models.

First a prospective conceptual architecture of the computational models and processes is outlined and then the rule sets comprising the models are described.

The conceptual architecture for the computational models and processes is illustrated in figure 5. In the context of the whole project, the components controlling visual explanation will be integrated within a larger architecture. The most important components which the visual components will interact with have been shown, e.g. the Planner and Discourse focus/query analyser. The Planner submits typed queries to the Domain knowledge retrieval process which extracts the appropriate knowledge structures to answer the query. The discourse analyser interacts with the Planner

which controls the assembly and delivery of the explanation. The modules illustrated in fig 5 are all subordinate to the Planner which integrates visual and verbal explanation. Hence in the scenario of explaining E-mail networks, the planner may call the visual explanation modules to select an image for an explanation strategy that calls for illustration.

The Media Selection Controller advises the Planner about the suitability of visual illustration of specific knowledge structures. A simple rule set is employed which tests the meta-properties of the TKS knowledge structure and its contents. For instance, structural knowledge is a good candidate for visual illustration. However structural TKS may also represent abstract objects and their relationships, e.g. subject classes of books in a library system. The rules therefore give a higher weighting to structural TKS containing concrete objects. Other knowledge structures are less suitable for visual illustration are given lower weightings, e.g. procedural and planning knowledge. The outcome of the rules is a suitability rating for selecting visual media for a particular knowledge structure. Some examples of the rules are:

```
TKS Object Gateway
          Property <abstract>
          Location <place>
          ConnectedTo<network1...n>
          Relationship < network1>

If Object-type = Structural Then Visual-Suitability = 0.5
If Object-type = Concrete Then VS = 0.25
If Object-type = Abstract Then VS = 0.15
If Object-type = Has-relationships Then VS = 0.5
etc...
```

The rules yield a cumulative score, so in the above example the object Gateway scores 0.70. This value is then used by the planner to decide if visual illustration is suitable.

If the planner chooses visual media then the Visual Image Selector is activated. Two approaches are possible for generating images; composition or use of templates. Composing images from a library of primitives and a generative grammar was investigated, i.e. defining a rule set for the possible spatial relationships between primitive components. While this approach was possible for simple conventioned images, e.g. a network diagram depicting connectivity between nodes on an E-mail network; in more complex images the combinatorial complexity would make the grammar difficult to process. Also some knowledge has to be encoded about the structure of an image with regard to its purpose. For instance if sending an E-mail message was to be visually illustrated by an animated sequence of letter icons being placed in post boxes and arriving in an in-tray, some knowledge of the metaphor and layout of letters, postboxes and in-tray would be required.

In view of this, image templates are used to represent specific knowledge structures in the E-mail domain, such as network topology diagrams, screen layouts, and diagrams illustrating components of messages, address lists, etc. The visual image selector matches the image templates against the properties of the TKS to be explained and retrieves the appropriate template. Each template has an associated spatial image model which is then used by the Visual Manipulation Planner.

The Visual Manipulation Planner is called by the main planner module. It has two tasks. First it selects the appropriate visual explanatory paradigm to match the question. Secondly it has to control interaction so that the image is used in conjunction with the verbal explanation. In the E mail domain the number of visual paradigms was limited to highlighting objects and use of arrows to indicate directionality. A rule set matches the question types to the paradigms and subsequent manipulations of the image, e.g.

If Q-type = Direction Then use Arrow
If Q-Type = Direction and Path Then use Arrow And Highlight-Path
If Q-type = Location Then use Highlight

The Visual Manipulation Planner makes reference to the TKS content of the explanation to decide which Object, relationship, etc to highlight. If the question makes spatial reference the Manipulation planner has to interrogate the image semantics model to locate the knowledge structure being referred to. It then selects the identity of the appropriate components to be manipulated. Further explanatory paradigms are suggested by the taxonomy and may be added so that images may play a more active role in explanation.

The Visual Explanation control sub system will interact with the object management system of a target hardware environment. The output will be in the form of instructions to select, highlight, and move objects on the screen display. This interface is not within the remit of the project hence output from the computation models will be command strings and clauses specifying which manipulations should take place on objects.

5. Future work

More detailed specification of the models and process is currently in progress and will be followed by implementation. Further research will be undertaken to link the control of visual explanation to the topic focus mechanism so that it may be possible to alter a visual display in light of shifts in discourse focus. For instance, if the verbal discourse changes from discussing networks to gateways then the gateway part of a diagram could be expanded.

The initial study of explanation by E-mail experts showed little use of diagrammatic visual media. More study on how experts use visual media during explanation is required, accordingly a corpus

of explanations will be recorded using a scenario encouraging use of diagrams and visual illustration. This will be analysed for give a better basis for strategies of interaction with images.

6. Discussion and Conclusions

This study should advance the use of visual media by making selection and response to images more flexible in dialogues with advanced advisory and tutoring systems. Use of visual media in intelligent tutoring systems has not been extensively researched in spite of the extensive use of images in many systems (e.g. STEAMER, GUIDON WATCH). Interacting with video images has been studied by Parkes (1989) who proposes a model for controlling animated sequences of film according to an instructional script. The approach taken in IDEAL is to place control at the discourse level so that visual media can be integrated with utterances in a verbal discourse.

However, integration requires the context of a plan for generating an explanation. One way forward may be to integrate interaction with images with more general discourse theories (e.g. Grosz and Snider 1986). This approach may enable flexible interleaving of media by employing a discourse theory incorporating both visual and verbal communication. Interaction with images in visual explanation is relatively simple in the current model, however, more sophisticated constraints could be incorporated. If this direction were followed visual explanation could approach the concept of virtual worlds (Borning 1986). Further data is required about the trade off between the sophistication of image based interaction and its efficacy.

Contextually sensitive use of images will require a theory of visual attention integrated with discourse focus. Although this problem can be partially answered using partitioned focus spaces (e.g. immediate, past attention, see Grosz and Snider 1986), we are a long way from understanding the relationship between what we can see and what we are attending to. Further empirical and theoretical research is required to link language and image processing.

Images have been described in a variety of formalisms for visual languages and other representations (Chang 1987, Krishnamurti 1987). However, these studies have concentrated on generation and editing graphical displays. While this is sufficient for graphical design tools, intelligent interaction with images requires understanding of the image semantics. Mapping of physical image components returns to the problem of attention. Without knowledge of the granularity of the observed image and the focus of the observer's attention, it is difficult to map the semantic properties to image components. To solve these problems a theory will have to embrace discourse, attention and visual comprehension. The research presented in this paper is still in its preliminary stages and can not propose any solutions, although it may claim to have elucidated some of the problems inherent in mixed media

interaction.

Acknowledgements

The author in indebted to Jenny Darzentas, University of the Aegean for help on the visual explanation taxonomy. Interactive Dialogues for Explanation and Learning is a basic research action supported by the Commission of the European Communities. Partners are City University, Queen Mary Westfield College, London; University of Pisa, Italy and University of the Aegean, Greece. Thanks are also due to Maggie Cooper for her helpful comments on drafts of this paper.

References

Cawsey ,A. (1988) Explaining the behaviour of simple electronic circuits DAI research paper no. 376. Edinburgh Scotland.

Cawsey A.; (1990); Understandable explanations. In Proceedings 5th Alvey Explanation Workshop, Ed N. Filer; University of Manchester, April 1990.

Chang S.K. (1987); Visual languages: A tutorial and survey. IEEE Software 4(1), pp 29-37.

Cleveland W.S. (1985), The elements of graphing data, Wadsworth.

Grosz B.J. and Sidner C.L., (1986), Attention, Intentions, and the Structure of discourse. Journal of the ACL, Vol 2(3), pp 175-204.

Hollan J, D, Hutchins, E.L, & Weitzman, L. (1984) Steamer: an interactive inspectable simulation-based training system. AI Magazine vol5 No 2 pp 15-27.

IDEAL (1989); Interaction Dialogues for Explanation and Learning, Technical Annex, Esprit Basic Research Action 3160, Dept of Business Computing, City University.

Jackendoff R.; (1983); Semantics and Cognition. MIT Press, Cambridge, Mass.

Johnson P, Johnson H., Waddington R. and Shouls A.; (1988); Task related knowledge structures: Analysis, modelling and application. In People and Computers IV (HCI-88); Eds Jones D.M. and Winder R.,pp 35-62, Cambridge Univ Press.

Klein E. (1987) Dialogues with language, graphics and logic. In Esprit 87: Achievements and Impact Part 2, pp 867-873

Krishnamurti R. (1987). Issues in shape editing and shape representation. Internal report, EdCAAD, Edinburgh, Scotland.

Lee J., Kemp B and Manz T. (1989), Knowledge based graphical

dialogue: A strategy and architecture. In Esprit 89 Conference Proceedings, pp 321-333, Kluwer Academic Publishers, Dordretch.

Parkes A.P. (1989); The Prototype Cloris System: Describing, Retrieving and Discussing Videodisc stills and Sequences. Information Processing and Management, Vol 25(2), pp 171-186.

Maybury M.T., (1990); Classifying and reacting to user feedback to guide text generation. In Proceedings 5th Alvey Explanation Workshop, Ed N. Filer; University of manchester, April 1990,

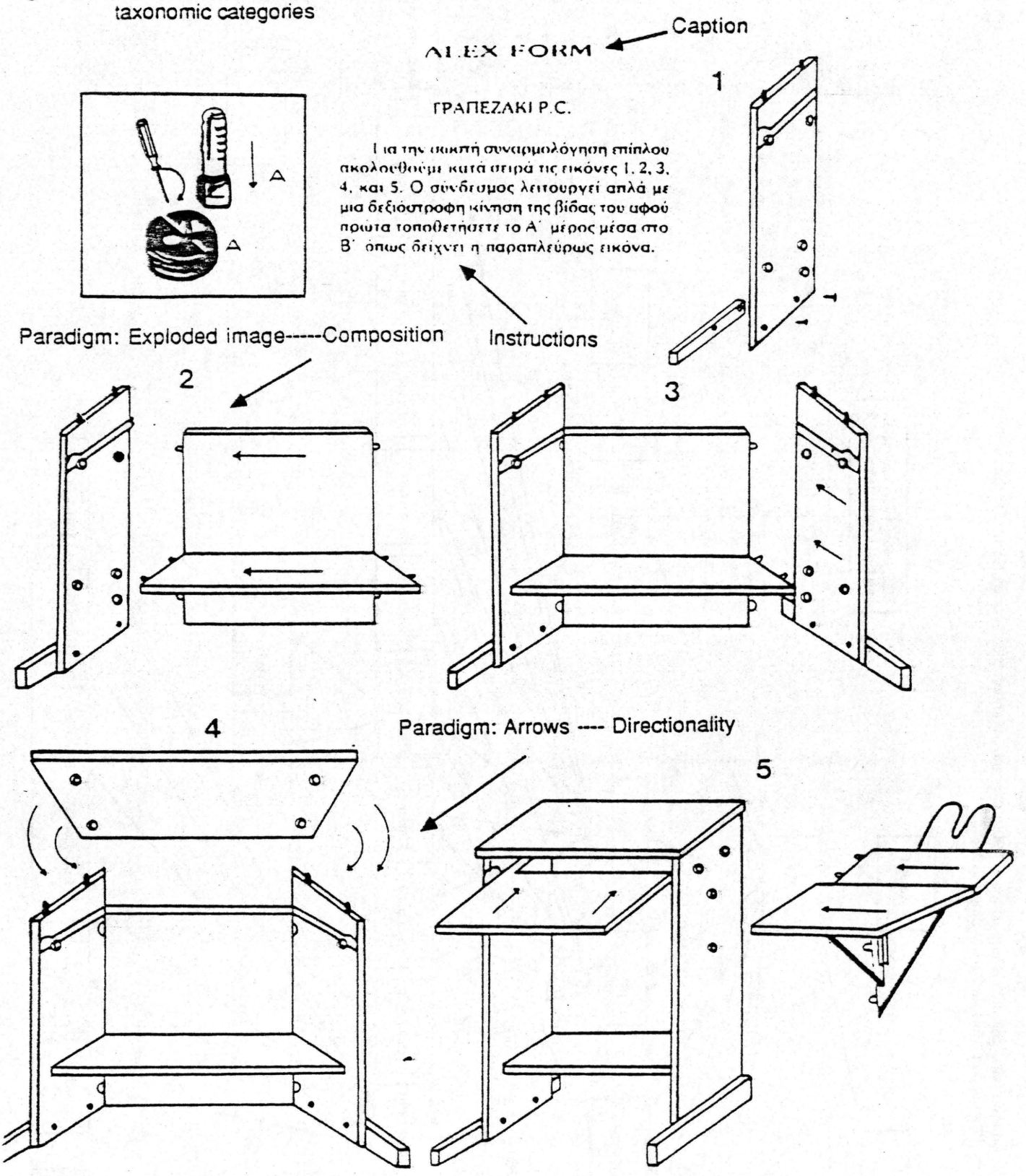
Marcus A. (1987) Graphic design for Computer Graphics. In Reading in Human Computer Interaction. Eds Baeker B. and Buxon W.A.S. Morgan Kaufman.

Richer M.H. and Clancey W.J. ;(1985); GUIDON-WATCH: A graphic interface for viewing knowledge based systems. IEEE Computer Graphics and Applications, 5(1),

Sarantinos E. and Johnson P.; (in press); Explanation dialogues: A theory of how experts provide explanations to novices and partial experts. To appear in Artificial Intelligence.

Sowa J.F.; (1984); Conceptual structures: Information processing in mind and machine. Addison Wesley, Reading, MA.

Figure 1. Visual illustration of furniture assembly task, annotated with taxonomic categories



Non-conventioned image

5 sub images in sequence----> Explanatory function: Task procedure

1 sub image -----> Explanatory function: Task action

Figure 2. Query Classification schema (after Sarantinos and Johnson 1990) showing augmented categories for spatial reference
The categories in bold classify the question "How do I get to Kings Cross ?"

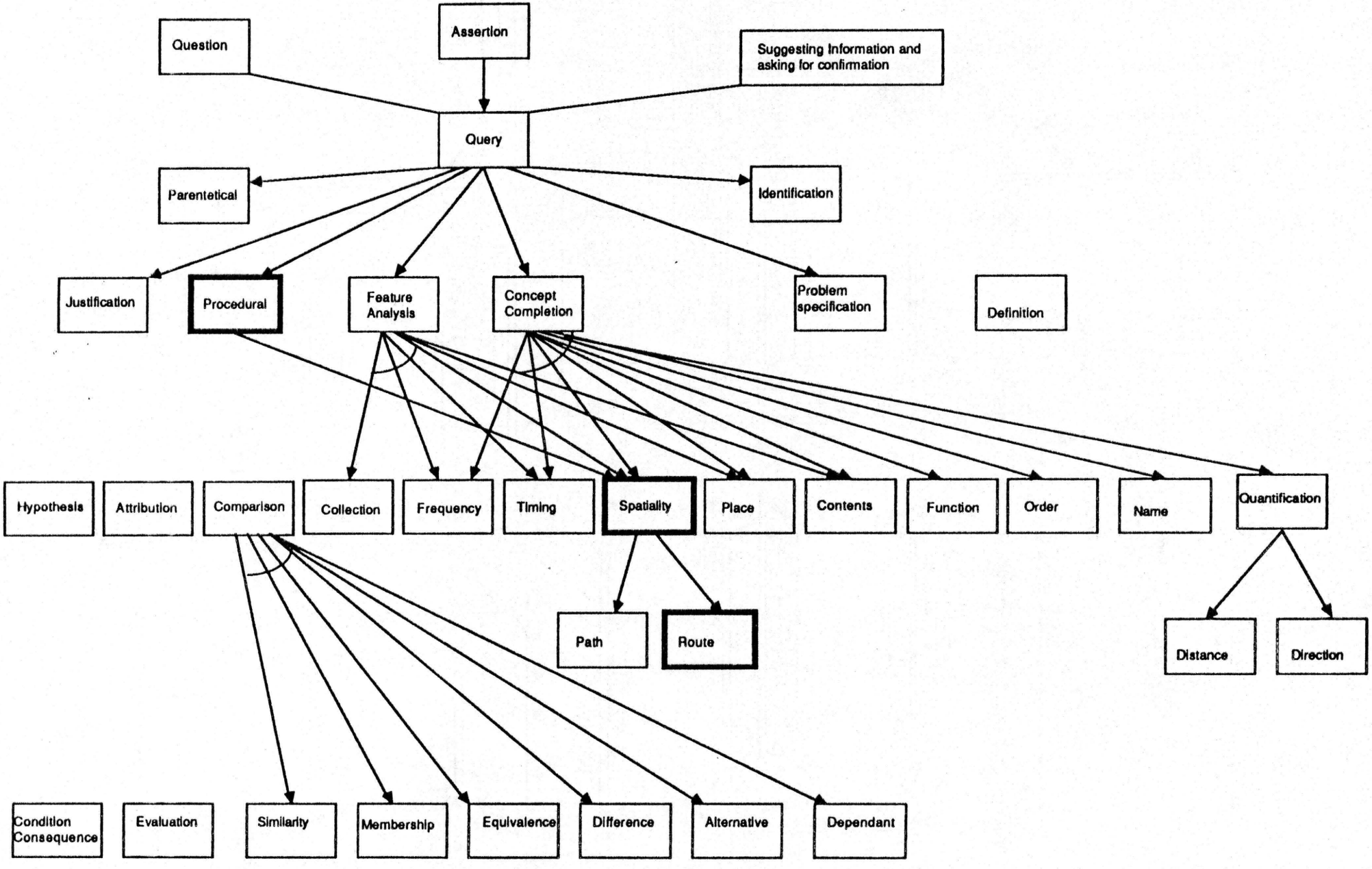


Figure 3. Illustration of E-Mail Networks.

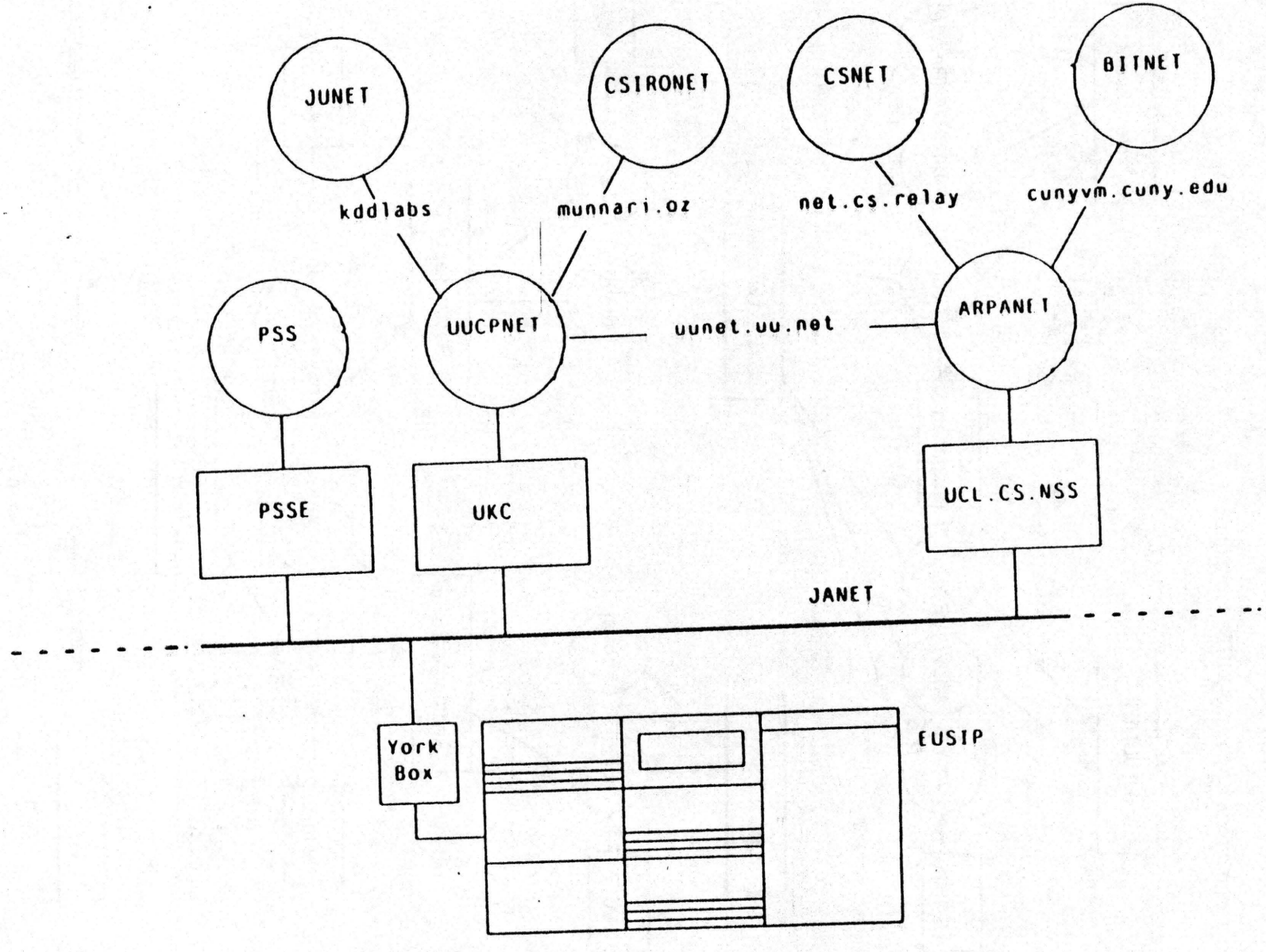


Figure 4 Spatial Image Model for Network diagram depicted in figure 3 (left hand side only)

Image Primitive Lexicon

Image Primitive	Maps to TKS structure
Circle [Diameter 2cm] <ENTITY	Network[Object]
Rectangle [Area 1.5 x 2.5 cm]<ENTITY	Gateway server[Object]
Arc <ENTITY	Connection [Relationship]
Labelled Arc < Arc	Gateway [Object]
Square [1.3 cms]<ENTITY	Network server[Object]
Composite rectangle [3.5 x 7cm]<Rectangle	Network host[Object]

Image reference point: JUNET< coordinates 1,1.

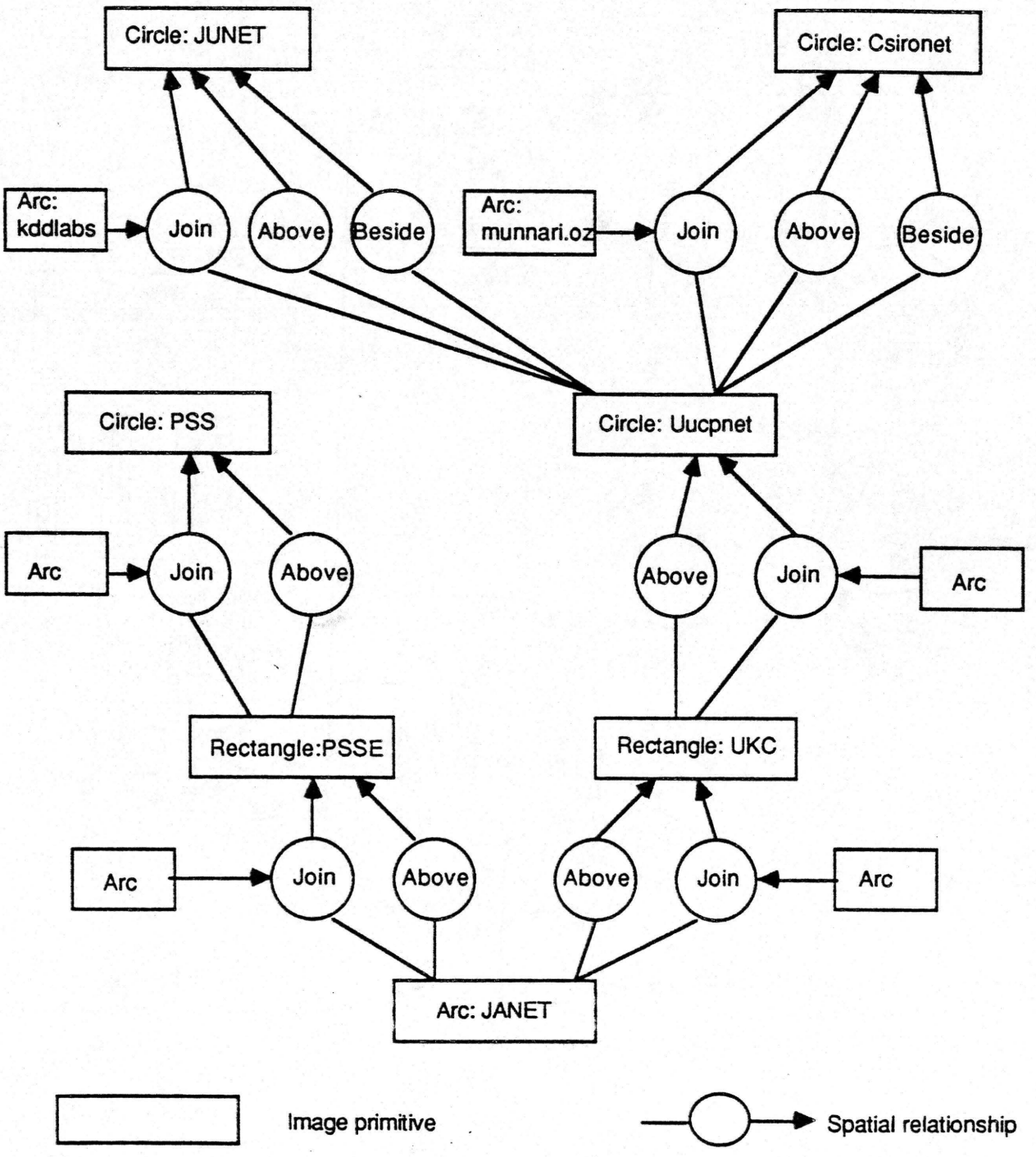
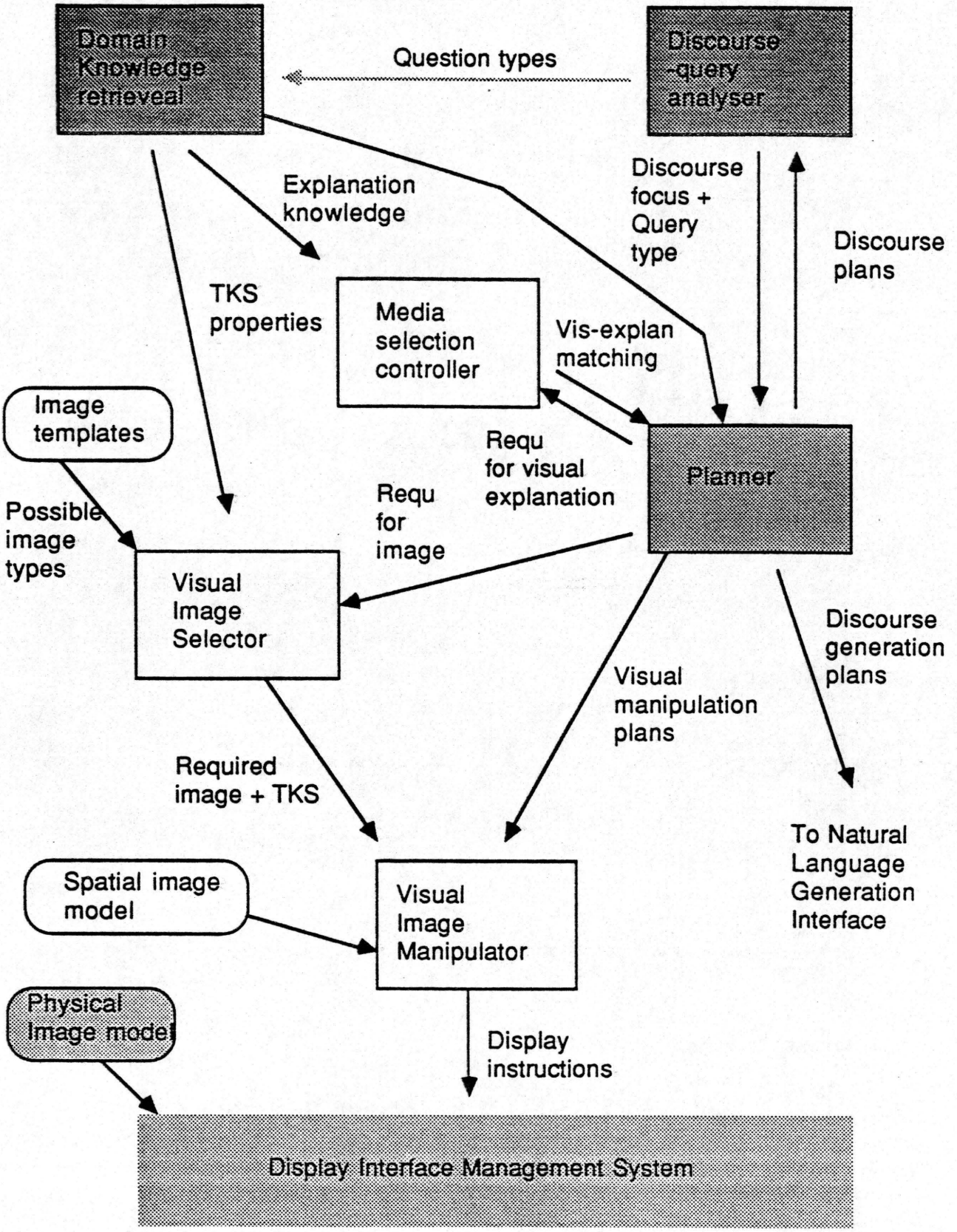


Figure 5. Conceptual architecture for Computational models and processes
Processes in shaded ▭ boxes are part of the 3160 project
but are not described in the paper.



How adaptive should a user interface be?

F.M.T. Brazier & G.C. van der Veer

Department of Cognitive Psychology
Vrije Universiteit
de Boelelaan 1111
1081 HV Amsterdam
the Netherlands

ABSTRACT

The basic assumption behind the design of any user interface is that it not only provides the user with the desired functionality but that it provides the user with a transparant and consistent way of interacting with the system. The degree of adaption and flexibility that should be included in a user interface needs to be addressed. To this purpose an experimental study was conducted. An information retrieval system with a modular architecture in which the dialogue module and the metacommunication module were distinguished, was designed and implemented. Subjects differed clearly in their preference for dialogue strategies. These differences were systematic and consistent at unit task level. A relation between spatial ability and consistent preference for a specific dialogue mode was found, in compliance with literature in this field. The nature of this relation is, however, difficult to define. Further research is required.

1. The design of a user interface

The basic assumption behind the design of any user interface is that it not only provides the user with the desired functionality but that it provides the user with a transparant and consistent way of interacting with the system. The degree of adaption and flexibility required is one of the topics which should be addressed in the design process.

Before addressing the issue of flexibility and adaption it is important to define the context in which the design of the user interface should be placed.

1.1. Context

The general architecture of the user interface is of importance to the design process of the user interface. To allow for the necessary distinction in functionality a modular approach to the design of an interface is advocated. A distinction should be made between the user interface itself and the application interface. The application interface is responsible for the interaction between the user interface and the system, whilst the user interface is responsible for the interaction between the user and the application interface. The application interface translates input obtained by the user interface into input to the system.

The user interface contains facilities which are not directly related to the execution of tasks by the system, such as help facilities. In principle a distinction can be made within the user interface between the dialogue module and the metacommunication module. The dialogue module is responsible for the input to the system: the lexicon, screen design and style of interaction. The metacommunication module is responsible for all facilities which

assist the user in interacting with the system whether this concerns the meaning of icons or the request for help.

The next assumption which is made within this context is that a distinction may and should be made between unit tasks and basic tasks (van der Veer, 1990). Unit tasks are the smallest tasks a user considers when interacting with a system. A basic task is the smallest task the system itself is designed to consider at its highest level of abstraction. This distinction is relevant to the discussion as designers can aim to define basic tasks at unit task level.

Not only are differences between system and user notable. Users themselves differ greatly. Research on individual differences in relation to user interface design suggests the need to provide users with user interfaces which match their individual needs and preferences (van der Veer, 1989). Spatial ability, a characteristic by which an individual can be considered, for example, has been found to influence users' performance with certain computer systems (Rothkopf, 1986). The degree to which the interface should adapt to the user is, however, unknown. What is known, however, is that users employ different strategies when working with a system and that these strategies should preferably be supported by the system (see Beishuizen & Brazier, 1986).

1.2. Adaption

During the design phase a number of aspects requiring adaption to the user should be considered: (1) the user groups, (2) learnability, (3) flexibility in relation to users' preferences and (4) flexibility in relation to help facilities.

Adaption to the user group for which the system is designed seems logical. The knowledge of the user (the level of expertise), the language, the unit tasks which can be distinguished in the domain and other characteristics of the group which may influence the use of the system, should all be taken into account.

Consistency in the interface is necessary for learnability as is transparancy (see Waern, 1986). The way in which the user interacts with the system should be consistent. The mode of interaction should be clearly defined and obvious to the user. The user should be confronted with the desired functionality in a way which is directly related to the user's own requirements (unit tasks). Communication about communication (metacommunication) should provide support for all interaction with the system.

When users' preferences are known, the system should be flexible enough to adapt to the consequences. Users' preferences, for example in relation to dialogue mode, should be integrated into user interface design. Whether the system should automatically adapt to the user or whether the user should be the party to request adaption is unclear.

Flexibility in the types of help available is something which is often given too little attention. A distinction can be made along the lines of Moran (1981) as to the level of help required for instrumental help. A user may wish to know what (s)he can do with the system in general (task level feedback), which tasks can be done within a certain state of the system (semantic level feedback), how the tasks can be performed (syntactic level feedback and with which devices the input can be entered (key-stroke level feedback). The content of each of these types of feedback may well overlap in a given state of the system, but the content should be formulated to adapt to the user's state of mind. In addition to instrumental feedback strategical feedback should also be available, feedback aimed at

supporting the domain specific aspects of interaction with the system. Lexical feedback on the definition of the different terms used in the interaction was also implemented.

2. An information retrieval system

In order to examine the effects of flexibility in aspects of the design of the user interface an information retrieval system with the modular architecture described above, was designed and implemented.

The information to be retrieved was stored in a database as described in the next section.

Adaption to the users was acquired in the design phase by analysing teh situation in which the system was to be used. This included necessarily analysing the tasks involved in information retrieval within the context in which the sytem was to be used, Geography education. The results of this analysis will be described below as the unit tasks which have been translated into basic tasks at application interface level.

One assumption behind the design of this information retrieval system is that flexibility is required in interaction with the system, even at the level of input. The implementation of the dialogue module is described in the section on the user interface.

2.1. Database

The data stored for the purpose of retrieval included: (1) the conceptual model of the system in which geographical concepts are defined on the basis of relevant attributes and (2) numerical data on 19 European countries, then regions and the provinces of Italy and of the Netherlands: for 1964 and for 1984.

This data was stored in a relational database created with Troll (Wasserman & Kersten, 1984). Further detail on the implementation is not considered of importance in this context.

2.2. Design of the application interface for an information retrieval system

The information retrieval system employed in this experiment was designed for the purpose of experimentation. The basis for the design was formed by analysis of the domain in which it was to be used, namely Geography, and on the analysis of the tasks which were to be performed (data retrieval, manipulation and examination). The unit tasks distinguished as a result of domain specific task analysis (see Brazier & Trimp, 1990) were:

(1) *make* a set
(2) *show* a set
(3) *sort* a set
(4) *remove* a set
(5) *overview* of sets
(6) *union* of two or more sets
(7) *intersection* of two or more sets

(A set can be seen as a view of a relation in a relational database.) The intermediary language formed the basis for the design of the dialogue module. The unit tasks provided the basis for the intermediary language in which the semantics of the interaction have been defined. The intermediary language itself provides the interface between the dialogue module and the database, see Fig 1. The seven unit tasks defined in the intermediary

language were mapped directly onto seven basic tasks in the dialogue module.

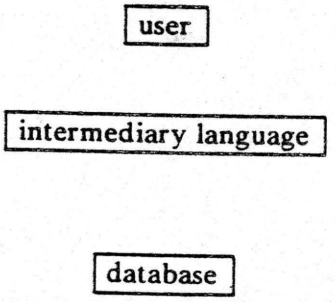


Figure 1. The intermediary language

The technical design and implementation of the interface between the intermediary language and the database, and of the system itself (including implementation of the help facilities) are not described in this context as they are not considered relevant to the design of the user interface.

2.3. The design of the user-interface for an information retrieval system

Once the intermediary language had been defined the design of the dialogue modules and the metacommunicative modules was undertaken. In previous comparative research Brazier & Beishuizen (1988a, 1988b) found a command-based interface to be more effective than a semantically equivalent menu-driven system. The mental models the users formed of the system were more complete, the strategies more effective and learning time was equivalent. However, the disadvantages of command-based systems are well understood, in particular the demands on memory. To compromise a dialogue form was chosen in which optimal flexibility was acquired: a specification could be constructed either directly as a command or through form-filling. When entering a specification the user is free to switch from command mode to form-filling mode at will. The mode does not have to be set in advance. If, when entering a command, the syntax or the attributes are no longer known to the user (s)he can easily switch to form-filling by using the function-key assigned this function. The command as entered before switching to form-filling mode is preserved as part of the specification, all new input (the rest of the one command) is entered in form-filling mode.

In addition to the function-key to switch from one mode to the other, four other function-keys were assigned roles in the acquisition of instrumental help. The four types of help incorporated are: lexical help ("what does this mean"), semantic help ("what can I do"), syntactic help ("how do I do it") and key-stroke help ("which key(s) do I use"). Task level instrumental help was not provided as the basic tasks for which the system could be used had been defined as the unit tasks of the user, and as such overlapped the semantic level instrumental feedback to a very great degree.

The way in which requests for lexical help are made known to the system depends on the mode in which the user is interacting with the system. When lexical help is requested in form-filling mode the user marks the relevant attribute on the form. If the user is in command mode (s)he is asked to enter the name of the entity for which lexical help is required.

The user's mode preference does not influence the way in which requests for the three remaining types of help are made known to the system. The state of the system does, however, influence the content of the semantic and syntactic help provided. The design of the modules incorporated in the interface has been described in Brazier & van der Veer (1990).

2.4. A few technical specifications

The information retrieval system, including the application interface and the user interface, has been written in the programming language C. Designed as a modular system written for a fairly open operating system the information retrieval system has been ported from the 4.3 BSD UNIX systems on which it was developed to Xenix System VR3 systems. Two Xenix systems each with eight graphical terminals were used in the experiment to be described below.

3. Experimental study

The information retrieval system described above provided a tool with which experimental study could be conducted. The effects of a number of the design decisions could be examined.

One of the requirements for experimentation is that the tasks with which the subjects are confronted are of interest to the subjects. To facilitate motivation the tasks may not be too difficult but also not too easy. The tasks within this experiment required domain specific reasoning skills of a type often encountered in Geography education at this level, using the system as a medium for information retrieval. The tasks were designed by the specialists in the field of Geography and Education for 15 to 17 year old students, addressing relations between geographical concepts.

One purpose of this study was to examine the consistency with which students employed two dialogue modes: command input and form-filling input. It was hypothesized that users would be consistent in their preference for one particular dialogue mode.

In addition the effect of spatial ability on preference in mode was a point of interest. Given the literature on this matter it was assumed that the preference for a mode of dialogue in which visual aspects play an important role (for example form-filling) is in some way related to spatial ability, how is unknown.

The use of the lexical help facility was another point of interest. Although the concepts included in the geographical experimental tasks were assumed to be known to the students it was interesting to know whether lexical support would be requested.

3.1. Subjects

The age of the subjects for which this experiment was devised was based on the assumption that students of 15 to 17 years of age possess problem solving skills comparable to those possessed by adults in the field of Geography. The design of the system was based on tasks which could be considered relevant to this age group within this context. As a result only those students with Geography in their curriculum were eligible. All 58 fourth form students of one secundary school in Amsterdam (two classes) participated. Complete protocols were collected for 52 students (20 male, 32 female).

3.2. Procedure

The week before the experiment began the purpose of the experiment and the consequences it would have on the students' schedule were explained to each class during a normal Geography lesson. One of the tests in the DAT intelligence test battery, spatial ability, was administered during this same lesson. During the following 4 weeks, four hours per week were alotted to each class for this experiment, mandating considerable changes in the school schedule. As the number of students participating in the experiment during one hour varied this does not imply that each student spent 16 hours working with the system. A session is one school hour, equivalent to 45 minutes. Sessions were not necessarily disjunct. Two sessions could be planned for two consecutive school hours. The implications of the schedule on the number of hours an individual student invested in the experiment is shown in Table 1. Each student spent approximately 10 hours working with the system.

Session	Phase
1	Introduction and Intelligence Test
2	Instruction
3	Instruction
4	Instruction
5	Instruction/Training
6	Training/Experimental Tasks
7	Experimental Tasks
8	Experimental Tasks
9	Experimental Tasks
10	Experimental Tasks
11	Mental Model Test

Table 1. An individual student's schedule

Instruction consisted of a short introduction by the experimenter followed by three to four sessions (Sessions 2 to 5) in which pairs of students worked through worksheets in interaction with the information retrieval system. Sixteen graphical terminals were available for this purpose. Training in Session 5 or 6 consisted of approx an hour's interaction with the system without any assistance either written or verbal (except for the hand-outs in which summaries of the specification requirements and of the domain-specific keywords, were provided).

The experimental tasks took four to five sessions (Sessions 6 - 11, depending on the student's progress) to complete. The last session (Session 12) of the experiment was used to administer the mental model test to the groups as a whole.

3.3. Materials

The worksheets consist of 19 pages of written instruction in which the use of the information retrieval system is placed in the context of 12 geographical problems. The student is taken by the hand and shown how to use the system: first as a form-filling system in which instruction at key-stroke level is required (the first 4 problems), then as a specification based system in which two modes of input are supported: forms and commands. The specifications which result from the dialogues are syntactically identical. The transition between modes is introduced in the fifth problem. In the following seven problems

the specification is shown: it is up to the students to decide which mode to employ. In addition to questions referring to the geographical problems *extra questions* are included in the worksheets concerning the use of system facilities and of command sequences. These *extra questions* are placed in between the questions referring to geographical (sub-)-problems, when applicable.

The training materials consist of eight questions not embedded in a geographical context. The sole purpose of these questions is to provide basic training in the use of the system for the execution of the five basic tasks described above.

The experimental tasks require geographical reasoning on the relation between concepts. The concepts can be operationalized using attributes known to the system. There were six experimental tasks in total. The first four were required for training: as these tasks were much more realistic than the tasks with which the students had been previously confronted, reasoning became an important factor. The last two tasks have been used to examine the use of the two dialogue modes.

The mental model test consists of two parts: one in which the knowledge the user has acquired of the system is measured by means of an objective test, the knowledge test, and one in which the level and style of the user's representation of the system is explored, the representation test. For more detailed description of the mental model test employed see van der Veer & Brazier (in preparation).

3.4. Results

On average students needed five or six *show* and *make* specifications for each task. (Although the *sort* specification was often used in interaction with the system the problems could be solved without its use. As not all students used the *sort* specification it is not considered in this context.)

Examination of the dialogue mode employed by the students shows that 34 of the 52 students were fully consistent in their preference for either forms or commands at basic task level for all of the specifications used in interaction with the system.

		SHOW	
		Forms	**Commands**
MAKE	**Forms**	11	4
	Commands	6	13

Table 2. Preference for dialogue mode in experimental tasks in relation to basic tasks

As Table 2 shows the preference for one of the modes was not necessarily consistent between basic tasks, but for 34 students is was (without a single exception) always consistent for a particular basic task. Eleven students always worked with forms, thirteen with commands, six used command input for all *make* specifications and forms for all *show* specifications whilst four always used forms for *make* specifications and commands for *show* specifications

Table 3 shows the preference for mode when protocols are scored on the basis of consistent usage in at least 75% of the input sequences (specifications). Of the 52 students 44 were consistent at this level: fifteen had a preference for forms for both *make* and *show*

		SHOW	
		Forms	**Commands**
MAKE	**Forms**	15	4
	Commands	9	16

Table 3. Preference for dialogue mode in 75% of the usage in experimental tasks in relation to basic tasks

specifications, sixteen for command input for both specifications, nine for the use of commands for *make* specifications and forms for *show* specifications, and four for the use of forms for *make* specifications and commands for *show* specifications.

The relation between the use of a dialogue mode and reference to the dialogue mode for explication of the effect of learning in the knowledge test has also been examined. In Table 4 the number of students who were fully consistent in their preference for dialogue mode for basic tasks for both the experimental tasks and for the explication of knowledge in the knowledge test.

		SHOW	
		Forms	**Commands**
MAKE	**Forms**	8	3
	Commands	4	13

Table 4. Number of students with a consistent preference for dialogue mode in experimental task and in knowledge test

Eight of the eleven students with a consistent preference for forms for both *make* and *show* specifications (see Table 2) used only forms for knowledge explication in the knowledge test, all thirteen students who had used commands for all specifications (see Table 2) in the experimental tasks were also consistent in the use of commands for specifications in the knowledge test. Of the mixed preferences four of the six students who had used commands for *make* specifications and forms for *show* specifications were consistent in the knowledge test as were three of the four subjects with the opposite preference.

		SHOW	
		Forms	**Commands**
MAKE	**Forms**	12	3
	Commands	5	16

Table 5. Number of students with a 75% preference for dialogue mode in experimental task and in knowledge test

In Table 5 the preferences when examined for 75% of the protocols and 75% of the descriptions, are displayed. Twelve of the fifteen students with a preference for forms for both *make* and *show* specifications are consistent in the knowledge test, as are all sixteen of the sixteen students with a preference for commands for both types of specifications. Three of the four students with a preference for forms for *make* and commands for *show* and five of the nine with the opposite preference, are consistent.

Although the Geography specialists maintain that the concepts used in the experimental tasks are well known to the students, 19 of the 52 students requested lexical help during interaction with the system. The other types of instrumental help were not requested during the experimental tasks which, in itself, is not surprising. The experimental tasks require the use of reasoning skills not previously required, but use the same user-interface the students master in preceding sessions. (In addition to instrumental help a group of students was provided with a facility to acquire strategical help. This will, however, not be discussed in this context, see Brazier (in preparation).)

The eight subjects with a fully consist preference for forms in both the experimental tasks and the knowledge test have a significantly higher spatial ability than those with a fully consistent preference for commands (thirteen subjects). (Student test: $t = 2.22$, $v=19$, $t_{.975} = 2.093$)

3.5. Discussion

The variation in preference at basic task level suggests the necessity for flexible interfaces in which different dialogue modes can be accessed simultaneously. Subjects differed clearly in their preference for strategies. These differences were systematic and consistent at unit task level. Given two alternative modes of input students developed their own preferred strategies. The flexibility provided improved the ease with which the students could work with the system, their individual consistent preferences were supported.

A relation between spatial ability and preference for a specific dialogue mode was found, in compliance with literature in this field. The nature of this relation is, however, difficult to define. Further research is required.

4. References

Beishuizen, J.J. & Brazier, F.M.T. (1986). Search strategies in internal and external memories, Proceedings of the Third European Conference on Cognitive Ergonomics, Paris.

Brazier, F.M.T. (in preparation). Information Retrieval. Doctoral dissertation. Vrije Universiteit, Amsterdam.

Brazier, F.M.T. & Beishuizen, J.J. (1988). Information retrieval: the user interface, SIGCHI-Bulletin April.

Brazier, F.M.T. & Beishuizen, J.J. (1988). Informatiezoeken. Tijdschrift voor Onderwijswetenschappen. 18(3).

Brazier, F.M.T. & Trimp, H.C. (1990). Supporting information retrieval: a promising objective. In: EURIT 90 Technology and Education, electronic proceedings.

Brazier, F.M.T. & v.d. Veer, G.C. (1990). Design decisions for a user interface. In: Ackerman D. & Tauber M. (eds) Mental Models and Human Computer Interaction, North Holland, Amsterdam.

Moran, T.P. (1981). The Command Language Grammar: a representation of the user interface of interactive computer systems. International Journal of Man-Machine Studies. 15, 3-50.

Rothkopf, E.Z. (1986). Machine adaption to psychological differences among users in instructive information exchange with computer. In Klix, F. Wandke, H. (eds) Man-Computer Interaction Research MACINTER I. North Holland, Amsterdam.

van der Veer, G.C. (1989). Individual differences and the user interface. Ergonomics, vol 32, 1431-1449.

van der Veer, G.C. (1990). Human-computer Interaction: Learning, individual differences and design recommendation. Haveka: Alblasserdam

van der Veer, G.C. & Brazier, F.M.T. (in preparation). Users representations of systems: a methodological approach.

Wasserman, A.I. & Kersten, M.L. (1984) A Troll/USE Tutorial. Vrije Universiteit. Amsterdam

Waern, Y. (1986) Understanding learning problems in computer aided tasks. In Klix, F. Wandke, H. (eds) Man-Computer Interaction Research MACINTER I. North Holland, Amsterdam.

Stampato presso "Entergrafica s.r.l."
Via Savoia, 24 - 00198 Roma