EACE
European Association of Cognitive Ergonomics

HUMAN-COMPUTER INTERACTION: TASKS AND ORGANISATION

Proceedings
of the Sixth European Conference on Cognitive Ergonomics (ECCE 6)
Balatonfüred, Hungary, September 6-11, 1992.

Edited by
Gerrit C. van der Veer, Michael J. Tauber, Sebastiano Bagnara,
Miklòs Antalovits



Sixth European Conference on Cognitive Ergonomics (ECCE 6)
Balatonfured, Hungary, September 6-11, 1992.

Organized by:
CNR (National Research Council of Italy): PF FATMA
CUD (Italian Consortium for Distance University)
Italsiel
National Committe for Technological Development of Hungary
Technical University of Budapest
University of Siena

CUD
Corso Vittorio Emanuele II, 229
00186 Roma, Italy

Library of Congress Cataloging-in-Publication Data
Human-computer interaction: Tasks and Organisation
Proceedings of the Sixth European Conference on Cognitive Ergonomics
Editors. Gerrit C. van der Veer, Michael J. Tauber, Sebastiano Bagnara, Miklòs Antalovits

ISBN 88-7721-232-2

European Association of Cognitive Ergonomics

ECCE 6

SIXTH EUROPEAN CONFERENCE
ON COGNITIVE ERGONOMICS

HUMAN-COMPUTER INTERACTION: TASKS AND ORGANIZATION

PROCEEDINGS

Balatonfüred, Hungary, September 6-11, 1992

Edited by
Gerrit C. van der Veer, Michael J. Tauber, Sebastiano Bagnara, Miklós Antalovits

ECCE 6, 1992
Bad structured

Contents

Preface (G.C. van der Veer, M.J. Tauber, S. Bagnara, M. Antalovits) vii

Part 1:
User-computer interfaces: Theory and Analysis

E. F. Churchill The formation of mental models: are "device instructions" the source? 3

S. Bagnara, D. Ferrante, P. Legrenzi, M. Malfatti, A. Rizzo Active and latent failures: How counterfactual thinking shifts the focus in causal assessment 17

S. P. Davies Strategies, errors and display-based skills in a complex task 29

L. Linde Effect of fatigue in man-machine interaction 45

Part 2:
Cooperation and communication: Analysis and evaluation

F. Vanderhaegen, I. Crevits, S. Debernard, P. Millot A dynamic task allocation in air traffic control: Effects on controllers' behaviour 59

G. De Michelis, P. Donzelli, T. Schael, B. Zeller Computer support for cooperative work in space science activities 71

C. C. Wood A cultural-cognitive approach to collaborative writing 87

C. Grusenmeyer The interest in the notion of shared functional representation between the operators in change of shift phase 103

Part 3:
Cooperation and communication: Theoretical approaches

J. May, I. Denley, U.-B. Voigt, S. Hermann, P. F. Byerley Designing IBC services which enable users to reach their goals 117

Y. Waern, K.-G. Waern Cooperative problem solving, as reflected in an exploration of some mechanical design engineering projects 135

B. Adelson, T. Jordan The act of negotiating during design 145

C. Castelfranchi, R. Conte, A. Cesta The organization as a structure of negotiated and non negotiated binds 155

Part 4:
User interface evaluation

T. R. G. Green, M. Petre When visual programs are harder to read than textual programs 167

C. M. M. Hurts User- vs. system-initiated data entry to expert systems: A behavioral perspective 181

C. M. Allwood, T. Kalén Lack of usability in a patient administrative system as a function of events in the system implementation process 193

L. Iszó, M. Antalovits, A. P. O. S. Vermeeren The role of pilot studies in user-interface research 205

Part 5:
User interface design

I. Burmistrov Flexible script interface: An intelligent spatial interaction style 219

S. M. O'Brien, E. A. Edmonds, L. Candy, N.P. Rousseau Visualization and graphical interaction: Contrapuntal support for knowledge-workers 231

A. Dijkstra, C. Huls Searching for linguistic arguments: A preliminary assessment of the usefulness of linguistic representations in an Editorial Supported Environment 243

R. Oppermann Adaptively supported adaptability 255

Part 6:
User interface theory

G. de Haan, G. C. van der Veer ETAG as the basis for intelligent help systems 271

V. Kaptelinin Integration of computer tools into the structure of human activity: Implications for cognitive ergonomics 285

M. Rauterberg Cognitive Complexity: An empirical method of a quantitative measurement 295

List of contributors 309

Preface

The European Association of Cognitive Ergonomics (EACE) organises biannual conferences on topics related to Human-Computer Interaction. This volume contains the papers accepted for the sixth European Conference on Cognitive Ergonomics (ECCE 6), Balatonfüred, Hungary, September 6-11, 1992. The focus of this conference is on tasks and organisation in human-computer interaction with special reference to the concepts of cooperation and communication, user-computer interfaces, and organisational structures. For each of these concepts, contributions were invited related to theory, analysis, design, learning, and evaluation. All proposed papers were reviewed by at least 3 referees.

The order of the chapters in this volume follows the order of presentation at the conference. The first part hits some aspects of the psychological basis of human-computer interaction, dealing with theories and analyses of user-computer interfaces. The next parts focus on cooperation and communication, respectively from the point of view of analysis and evaluation (part 2) and theoretical approaches (part 3). The user interface is the core concept of the other half of the volume, analysed first of all by evaluation approaches (part 4), followed by an overview of research on design (part 5) and concluded by some theoretical contributions.

The editors like to mention the following colleagues, who provided reviews and comments on the proposed contributions: David Benyon, Paul Booth, Carlo Cacciabue, Elisabeth Dienes, Giorgio de Michelis, Pierre Falzon, David Gilmore, Thomas Green, Jean-Michel Hoc, Lajos Iszó, Krisztina Lakatos, Lena Norros, Guiseppe Mancini, Reinhard Oppermann, Rob Roe, Janine Rogalski, Boris Velichkovsky, Hans van Vliet, Yvonne Waern, Hartmut Wandke, Ted White. The collection of the reviews, the contacts with the authors about the final papers and the organisation of the manuscript of the proceedings for publication has been greatly facilitated by the help of Elly Lammers. We thank her for her patience and persistence.

Amsterdam, June 3, 1992

Gerrit C. van der Veer,
Michael J. Tauber,
Sebastiano Bagnara,
Miklós Antalovits

part 1

USER-COMPUTER INTERFACES: THEORY AND ANALYSIS

"The Formation of Mental Models: are 'Device Instructions' the source?"

Elizabeth F. Churchill

Department of Psychology
The University of Nottingham
Nottingham
U K

Abstract

When considering users' understanding of devices, two types of device knowledge have been proposed: procedural or "how-to-do-it" knowledge, and model-based or "how-the-device-works" knowledge. This paper compares these knowledge types in terms of their informational and computational properties with respect to achieving tasks, and considers the effects of practice on these knowledge structures. The paper reports: (i) computational models (in Soar) of learning to use a simple device from procedural instructions and from "how-it-works", device model instructions, and (ii) empirical investigations testing the hypotheses derived from these models. From the computational models it was predicted that (1) expert performance on problem solutions would develop through practice, and (2) although expert performance would not be observably different for subjects given procedural instructions compared to those given device-model instructions, the competence underlying expert performance would differ. This underlying difference would differentially affect the development of new solution methods in later problem solving. The experimental results showed clear performance speed-up with practice. However, contrary to the predictions of the models (and models of routine skills (Card, Moran and Newell, 1983)), it appeared that subjects did not learn methods (i.e. solutions to *problem types*), but learnt local, display-driven, action selection rules. In addition, it was shown that subjects presented with procedural instructions develop model-based knowledge.

1. Introduction: the received wisdom on Mental Models

When considering users' understanding of devices, two types of device knowledge have been proposed: procedural or "how-to-do-it" knowledge, and model-based or "how-the-device-works" knowledge. Procedural knowledge consists of task-related strings of actions which offer the user a "recipe" for getting a task done. By contrast, model-based or device model knowledge captures the causal relationships between task-relevant, internal device components. Although there is much debate about what exactly constitutes a device model (c.f. Wilson and Rutherford, 1989), typically the term is used to refer to a schematic representation of the internal components of the device and the causal relationships between those components. This model can be "run", enabling users to mentally simulate the effects of their actions before executing them. This mental simulation provides device-related command semantics; knowing the effects of an action on both visible and invisible device states means that the consequences of commands are more fully understood, offering more scope for using those commands in novel situations, and less likelihood of incurring unexpected and unfathomable results.

For these reasons, it is generally thought that *teaching* a device model of this kind facilitates the development of smooth performance. This intuition has been supported to some extent in

experimental work (see Rouse and Morris, 1986; Mayer, 1976; Halasz and Moran, 1983; Halasz 1984; Kieras and Bovair, 1984; Bibby and Payne, 1990). These studies suggest that these representations are indeed used in action selection, remain fairly stable over time and are re-consulted on presentation of unfamiliar problem types (c.f. Halasz, 1984).

There are a number of problems with these studies: (1) a central *assumption* in these studies is that the user's mental representation corresponds to that given in instruction. Given that user knowledge is generally affected by interface features, by features of the task and by any instructions that are presented (Barnard, 1987; Norman, 1983), and that different representations of a system may give rise to the same functional outcome (Norman, 1983), it is not clear that this is a reasonable assumption. In addition, it is not clear that pure observation of behaviour is an adequate technique to establish the nature of mental models. (2) Analysis of the instructions and the tasks presented in these studies has shown that the experimental design is biassed in favour of subjects presented with device model instructions. Most instruction booklets contain examples. The effect of this is that model-group subjects receive *both* solution procedures and device model instructions whilst procedural group subjects receive *only* solution procedures. Experimental tasks tend to be divided into those requiring the retrieval and application of example solution procedures (which is supported by both sets of instructions) and those requiring knowledge of the internal components for inferring correct actions (which is only supported by the device-model instructions). The instructions for the two groups are therefore not 'informationally equivalent' (c.f. Larkin and Simon, 1987; Bibby, 1989); device-model subjects have the information required to derive and apply solutions and to infer device-related command semantics whilst the procedural subjects have only the example solutions. It hardly seems reasonable to conclude that device-related command semantics are superior to procedural instructions for action selection when the device related semantics are *necessary* for the action selection. All one can conclude is that if you don't give people enough of the right information, they find solving problems hard. This concurs with results reported by Duff and Barnard (1990) and Duff (1990). (3) there is no account of learning. Through practice, actions selected for tasks become compiled into automatised solution methods (Anderson, 1983; Laird, Rosenbloom and Newell, 1986), but within these studies, there is no account of the way in which device knowledge affects the development of automatised task-action mappings. In addition, observation of users shows that local models of device functioning are developed all the time. Few studies have allowed for or explicitly charted changes in user representations through device interaction (for an exception see Brazier, 1991) although it seems likely that users will consider *all* the resources available to them to get tasks achieved, and this implies that the underlying representations will consist of more than just the instructions presented.

In the light of these factors, it is not surprising that the results of studies into mental models and performance are by no means clear-cut (c.f. Wilson and Rutherford, 1989; Foss, Smith and Rosson, 1982 (cited in Duff and Barnard, 1990); Duff, 1990). The intuition that device-model instructions are *better* may not be true. For example, when specifying the *computational* properties of alternative representations, whilst device model knowledge proves more informative in the studies cited above, it is also *more costly* for deriving solution methods (c.f. Halasz and Moran, 1983). To mentally simulate the effects of actions requires a number of mental transformations. This computational perspective suggests there may be tasks for which procedural representations are as effective but computationally less costly than device model representations. In these situations, device model knowledge, far from offering more understanding and easier problem solving may prove a handicap. These issues are addressed in the simulation and experimental work.

2. Computational modelling of alternative instructions

A number of models of expert performance have been reported, but these typically address and describe the knowledge that exists *once expertise has developed*. Computational models must

account for the way in which knowledge is used and for changes in that knowledge as a result of learning, offering models of expert performance *as it develops*.

In this vein, I built two models of device knowledge in Soar, a cognitive architecture (Laird, Rosenbloom and Newell, 1986; Newell, 1990; for more on these models see Churchill and Young, 1991). The device selected for analysis was a 'postfix notation' or 'reverse polish notation' calculator. Postfix notation calculators have an internal memory stack; to solve calculations numbers must be stored in this stack. For example, to solve the binary calculation '5 + 8', users must press the key marked '5' (this number appears in the display), then press the key marked 'ENTER' to place the number in the display into the memory stack, press the key marked '8' and then press the key marked with the arithmetic operator '+'. Pressing the arithmetic operator combines the number last entered into the memory stack with the number in the display to produce the result, 13, which is placed in the display. The operation of this type of calculator given alternative instructions was studied by Halasz and Moran (1983) in an attempt to demonstrate the superiority of model-based knowledge in device use. This was therefore considered an ideal candidate for investigation.

In Soar, search takes place in a hierarchy of nested problem spaces; in the simulation models of calculator knowledge, external world actions (e.g. keying in numbers, pressing 'ENTER', etc) were operators in the "top-space". Sub-spaces represented the calculator knowledge other than external world actions; this knowledge was either procedural or model(stack)-based.

In simulation model 1, procedural knowledge was represented as a"recipe" datastructure located in a sub-space. This recipe represented a list of actions that need to be done to achieve a binary calculation on a postfix notation calculator; this datastructure therefore represents the knowledge embodied in instructions that specify to do task T, the relevant actions are 'A1 then A2 then A3 then A4'. Figure 1 shows a diagram of part of the datastructure; here, the first two actions in the recipe are shown. Each step in the "recipe" is marked as having "been done" or not; the value associated with the attribute "been-done" specifies whether or not an item has been inputted.

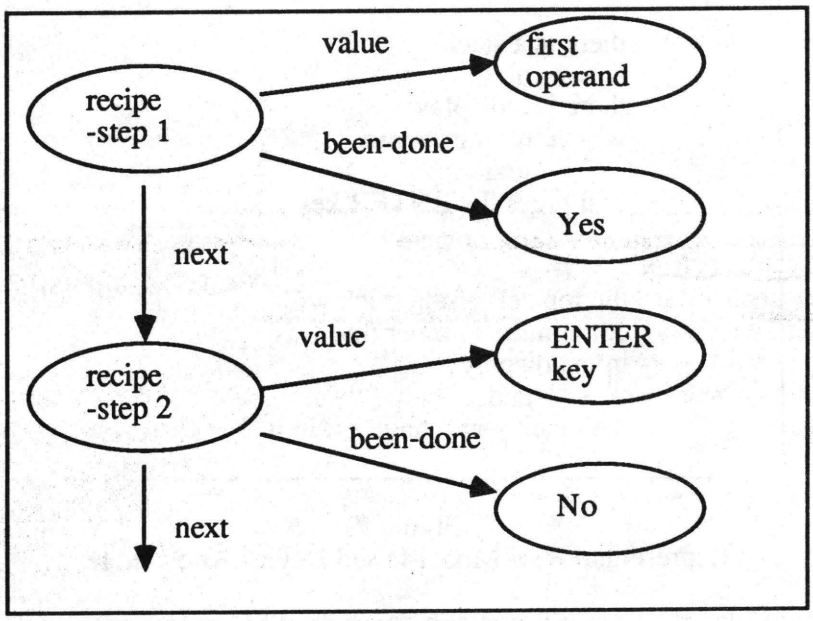


Figure 1
Recipe datastructure

A set of operators scan over the datastructure to determine how much of the recipe has been executed (e.g.whether the first operand has been inputted, the ENTER key has been pressed yet, etc). If the attribute "been-done" has the value "yes" the scanning goes on to the "next" action in the list. The first action that has the value "no" to the "been-done" attribute is selected as the next thing to do.

The datastructure and the scanning process correspond to remembering a set of actions and then mentally running through them one by one, "ticking them off" as having been completed when each one is done. When the next action needs to be selected the recipe structure must be inspected again.

In simulation model two, model-based knowledge of the calculator display and the internal memory stack was represented in a sub-space. Look-ahead search was used to simulate a forward running of the device model in the user's mind. The knowledge is represented as productions that match the current state of the device. These device states are imagined ones and do not necessarily correspond to those of the device in the real world. Figure 2 shows part of one production. Here, the conditions specify that when there is assumed to be an internal stack memory and the display cell of the stack has a number in it, then the result of pressing the ENTER key will be to place the number from the display into the top cell of the internal stack and to leave the display with a zero in it.

To determine the next appropriate action, the current state of the device is matched to the preconditions for the known actions; the current state description comprises what the user can see in the interface and the imagined state of the internal memory stack. The appropriate action is applied, leading to a new device state. This cycle continues, as if the device state were being altered from the start state to the desired goal state. This corresponds to the user's behaviour as imagining the effects of one's action on the device before actually carrying them out in the "real" world.

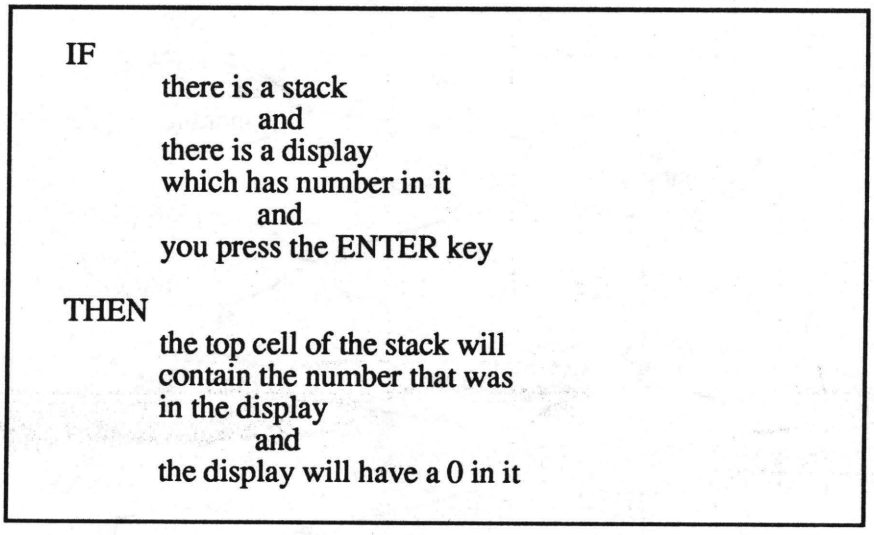


Figure 2
Representation of Model-based Device Knowledge

For both simulation models, search in the sub-space resulted in top-space action selection. As Soar has a learning mechanism, practice results in reduced search through the development of 'chunks'; these cache the method of action-selection which took place in sub-spaces. These

chunks fire automatically on re-presentation of problems of the same type, circumventing the need for sub-space search and accounting for speed-up practice effects.

Running the calculator models led to five observations: (1) initial problem solving to select actions was more extensive for the device model knowledge than searching the procedural recipe data-structure. More search is seen in performance as more time. This is in accord with experimental results (Halasz and Moran, 1983) which showed the model group taking longer to solve problems initially. (2) after a number of practice trials, the chunks fired to produce smooth performance. In terms of user performance, this corresponds to performance speed-up as a result of practice. (3) after learning, the performance of the models on the task was identical, although the underlying knowledge in the chunks from the two simulations was very different. Halasz and Moran reported comparable results between the two groups after practice on problem types for which the instructions were informationally equivalent; this expert performance was described as resulting from the execution of automatised solution procedures and 'method' execution (c.f. Card, Moran and Newell, 1983). However, there was no explicit consideration of the content of these methods, nor of how they developed. (4) on presentation of new problem types for which the compiled chunks were not appropriate, the sub-spaces were again searched for the derivation of new solution methods. This would be seen in user performance as increased time to solve problems on the presentation of new problem types. This is in accord with results shown by Bibby (1989) where initial performance differences between groups vanish with practice but return when presented with novel tasks. (5) the chunks from the device-model space search proved to be more flexible than the chunks from the procedural space search, implying that these would transfer more easily to new problems.

These results led to the following empirical hypotheses:

1. given problems for which device model and procedural instructions are informationally equivalent, users presented with device model knowledge will take longer in initial problem solving than users presented with procedural instructions, because device models are computationally more costly than procedures
2. that there will be performance speed-up as a result of practice for both groups.
3. that after practice, group differences will disappear and performance will be comparable.
4. that on presentation of a new problem type, users will search their device knowledge to derive a solution. Thus there will be an increase in time to solution on presentation of the first example of every new problem type.
5. that when the instruction knowledge is *not* informationally adequate, users will look for a solution. This is modelled in Soar as a search process over the users knowledge. The Soar models say nothing about what happens when that knowledge search fails.

3. Experimental evaluation of the models

The experimental design for both experiments is based on that of Halasz and Moran (1983). Experiment 1 compares representations in terms of the computational cost of solution development and application, and the effects of practice. Experiment 2 considers the informational properties of representations and considers how users annotate their existing representations to solve new problem types.

3.1 Experiment 1: using presented knowledge: practice and skill development

The aim of Experiment 1 is to study the effects of computationally different instructions on the development of expert performance. The first four of the empirical hypotheses discussed above are concerned with this issue.

Experimental Procedure

Ten subjects who had not previously used a postfix notation calculator were randomly assigned to two groups. One group was given "how-to-do-the-task-step-by-step", operational instructions for completing calculations on a post-fix notation calculator plus example solutions (the 'No-model' group). The second group was presented with the same instructions expanded with references to the internal states of the device, i.e. with reference to storage of numbers (both task numbers and intermediate results) in the internal memory stack (the 'Model' group). The instructional bias of earlier experiments was minimised by presenting fewer example problem solutions to the Model group. Groups were matched for age and educational achievement.

The experiment involved reading the instruction booklets, practice using the mouse, completing 50 arithmetic problems and completing a questionnaire which included giving a description of the device in terms of "how-it-works". The arithmetic problems varied in the level of embedding and bracketing from one level of embedding, e.g. "4 - (2 + 3)", to 5 levels of embedding, e.g. "(((((3892 - 1043) * 3) - 2945) * 2) - 1932) * 2 " . Problems were matched for number of keypresses, level of embedding and direction of embedding (left, right or centrally embedded).

Results

All subjects completed all the problems, implying that both sets of instructions were informationally adequate for producing solutions to the problems. The time to solution for each question was log transformed and the two groups compared. This measure showed the Model group taking longer on significantly more questions than the No-model group ($df = 49$; paired t-value 2.16; $p < 0.05$). This result supports the hypothesis that the Model group spent more time deriving solutions.

A number of performance measures were analysed: attempts data, timing data at the question level and inter-keypress latencies. These results showed that both groups found certain question more difficult than others, but on all measures the two groups performed comparably.

To determine which questions were more difficult and to observe the effects of practice, figures 3 and 4 show the amount of time "saved" through practice. Figure 3 shows the mean time to solution on the first and the last binary calculations in the problem set; there were 4 binary calculations which were evenly spaced through the problem set. Figure 4 shows practice effects on left and right embedded problems; here the time to solve the last example is subtracted from the time to solve the first example, to determine how much time is "saved" as a result of practice.

Figure 3 illustrates clearly that both groups reduced the amount shows clear effects of practice; both groups reduce the time taken to produce a solution to under 5 seconds. In addition, this graph shows that the Model group take less time to produce a solution to the first calculation than the No-model group. Although this difference is not statistically significant, it counters the hypothesis that the model group would take longer to produce solution as a result of the more computationally costly device representation. There are two possible explanations for this: *either* the device-model representation is *not* more costly computationally for solving problems of this type than the procedural representation (contrary to the predictions derived from the simulation models), *or* the Model group subjects were not using the device-model representation.

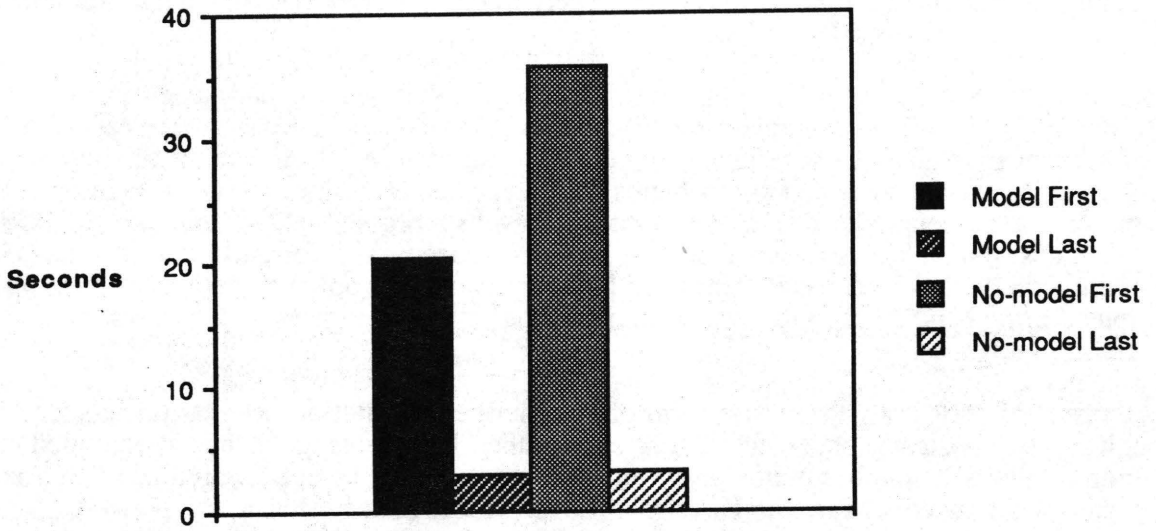


Figure 3
Measure of practice effects on binary calculations: first example and last example by group

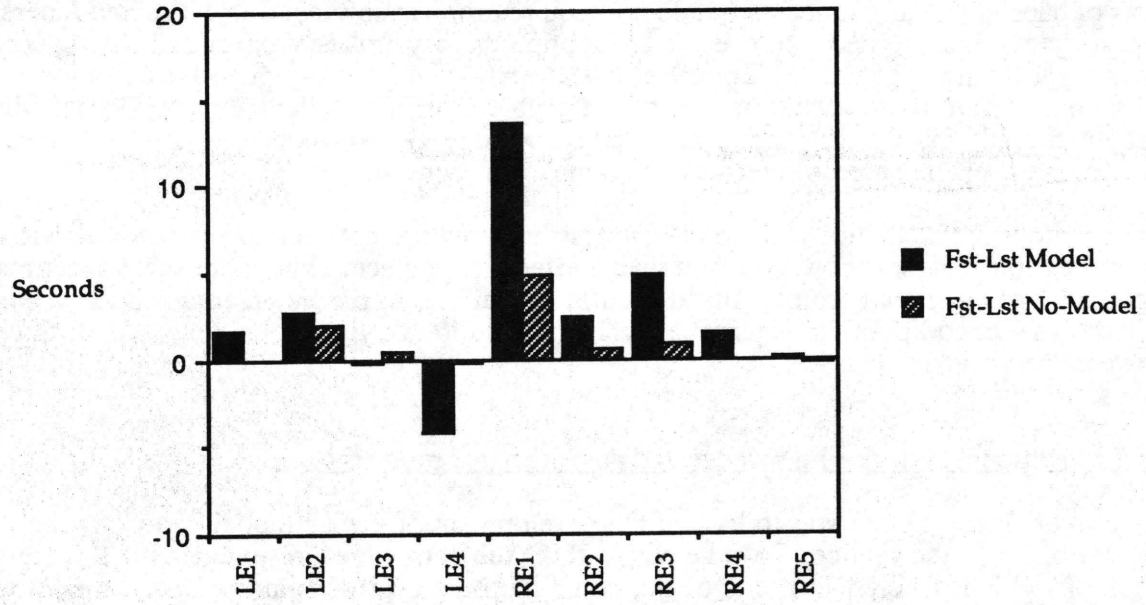


Figure 4
Practice effects: mean time to solution on first of problem type minus mean time to solution on last of problem type for both groups

Figure 4 shows the left embedded (LE1 - LE5) and right embedded (RE1 - RE5) problems. For both groups, there is a clear effect of practice reducing time taken to solve problems. More

importantly however, on these problems the reduction in problem solving time is clearly more pronounced for the Model group, particularly for the Right-embedded questions, RE1 - RE5. It would appear that these results confirm that on certain problem types, model-based knowledge is computationally more costly.

One explanation for the difference between this result and that on the binary calculations is that both instruction booklets illustrated the solution to a binary calculation; in this case, Model group subjects may not have been using the device model representation to solve the first binary calculation (which was also the first problems in the set) , but may have been able to retrieve the instruction booklet example. This is in accord with results reported by Duff and Barnard (1990) which suggest that subjects retrieve and apply learnt procedural information in preference to reasoning with device-model knowledge. In the more complex problems, there were no explicit examples for the Model group to retrieve so model-based knowledge was all that was available for solution derivation.

These results demonstrate that some form of knowledge compilation occurs, as the speed-up through practice is clearly observed. To address whether the knowledge is indeed compiled into 'solution methods' a number of analyses were carried out at the level of individual keystrokes using parameters specified by the Keystroke Level Model (Card, Moran and Newell, 1983). Points at which the keystroke level model would predict 'mental operations' were extracted from the keystroke data. These are at points where decisions may be taken and are seen as "planning boundaries", i.e. the points at which subjects retrieve solutions to the next sub-part of a decomposed task. If these planning boundaries reflect points at which subjects are determining the next action(s), then two hypotheses are of relevance: (1) As a result of practice, repeated action sequences are compiled into automatised solution 'methods'; methods are selected on the basis of cues in the task structure and do not require problem solving. Subjects should therefore be taking less time to select and execute the appropriate keystrokes at the end of the experiment than they were at the beginning. To reflect this, the recorded inter-key-press latencies for mental operations should be lower at the end of the problem set. (2) If device-model knowledge is computationally more costly than procedural knowledge, the No-model group should take less time for keystroke latencies which include mental operations than the Model group.

Contrary to these predictions, the inter-keypress latencies did not decrease through the problem set; regression analyses showed no significant effects of practice. This implies that performance speed-up did not result from reduced mental operations at predicted task-related planning boundaries. On addition, the keystroke level analysis did not yield consistent time differences between the groups.

3.2 Experiment 2: device knowledge and device related problems

The aim of Experiment 2 was to look at the development of device model knowledge through interaction with the device. To investigate this, subjects were presented with a variety of problems which differed in terms of the computational cost of solution development as in Experiment 1, but were also presented with a number of questions which were informationally not equivalent with respect to the instructions presented: some of the problems required explicit stack manipulations which were not supported by the procedural instructions. Subjects in this group therefore needed to develop model-based knowledge in order to solve these problems. Although Halasz and Moran (1983) do not report any such knowledge development, it is hypothesised here that subjects will indeed use the resources available to them to make up the deficit in the instructions presented. In terms of the Soar simulation models, this would mean that the external environment is used as a resource for search; this model of cognitive behaviour is not currently supported in the models.

Experimental Procedure

Twelve subjects who had not previously used a postfix notation calculator were randomly assigned to two groups: as before, one group was given "how-to-do-the-task-step-by-step", operational instructions for completing calculations on a post-fix notation calculator (the No-model group); the other was presented with the same instructions expanded with references to the internal states of the device (the Model group). Groups were matched for age and educational achievement.

The experiment involved reading the instruction booklets, practice using the mouse, completing 17 problems and completing a questionnaire which included giving a description of the device in terms of "how-it-works". The problems were of three sorts[1]: (1) "routine" problems required simple solutions, examples of which had been presented in the instruction booklet. These included binary calculations (e.g '5 + 8'), simple running problems (e.g. '6 + 5 + 9 + 1 + 4') and simple bracketed problems (e.g. '4 - (9 - 5)' & '(5 + 7) - 4)'), (2) "combination" problems, which are made up of combined routine problems (e.g. '(9 - 3) * (5 - 2)'), and (3) "invention" problems which involve extra-arithmetic constraints and require knowledge of the stack for solution derivation (e.g. '15 - (3 - 15) Type the 15 only once'). In addition a number of on-paper questions were presented to see if subjects could reason about solutions away from the calculator interface.

Results

Results from the on-paper problems showed no significant group differences suggesting subjects were equally able to reason with the knowledge away from the calculator.

The mean time for completion of all the problems in minutes for the Model group was 67 minutes (sd= 41.57) and 63 minutes (sd = 16.26) for the No-model group. This time difference between the groups is not statistically significant. More detailed time analyses are excluded because of the high variance due to subjects verbalizing throughout the experiment.

Statistical analyses showed consistently that both groups found the 'invention' problems the most difficult; there were no significant different differences between the groups on routine and on combination problems. The No-model group performed significantly worse on 'invention' questions on all measures. This is in accord with the results presented by Halasz and Morán (1983), and with the hypothesis that the instructions presented to the No-model groups were informationally impoverished and did not support the development of solutions to stack related problems.

Analysis of the verbal protocols support this proposition. It is also clear that subjects in the No-model group actively problem solve to derive a model of the internal functioning of the device. Analysis of the post-experimental reports showed that subjects in the No-model group had learned a mental model of the internal memory stack of the calculator. Some of the subjects responses in answer to the the questions "How does the calculator work?" are shown below in Figure 5.

Subject 7, No-model group

The description below illustrates that the subject understands (1) that numbers are stored, (2) that these numbers are stored in a particular order, (3) that keying in an arithmetic operator acts upon the stored numbers, and (4) that once used in a calculation, the numbers are no longer stored. The scope of the arithmetic operator (i.e. which numbers in the stored 'line' are acted on) is not specified.

---

1 The naming convention for the problem types is taken from Halasz and Moran (1983)

The numbers are held in a line until a function button is pressed when they are replaced by the result of the operation

Subject 8, No-model group

The description given by Subject 8 mixes task decomposition and task solution strategy with the calculator's functionality. For example, the first sentence specifies how the subject solved the problems in the context of this calculator, i.e. one sub-expression at a time. The description demonstrates that the subject understands (1) that the calculator stores numbers and results (2) that these stored numbers and results can be combined ('cross-referenced') by pressing arithmetic keys, and (3) that there is a specific order in which the stored items can be accessed ('the machine knows....').

The calculator first of all deals with one function at a time, i.e. 1 + 2. Then the answer is stored. The next function is then carried out independently i.e. 1 + 2 again but the two functions can be cross referenced if commands such as '+' and '+', or '+' and '-' are placed together.

Because the commands + and + are together the machine knows to first deal with the present package and then refers the answer to the previous package. The system of individual packages and then the reference of one package to another can be built up to solve complete calculations consisting of many calculations.

Subject 9, No-model group

This subject shows clear understanding of (1) the storage of numbers in a memory and the role of the ENTER command, (2) the ordered storage of numbers that are entered, and (3) the effect and scope of inputting arithmetic operators.

When you put a number in and press ENTER, it stores that entry for the present. If you then place another number in, followed by ENTER, this will store the numbers side-by-side.

If you have a series of numbers entered, and press a function, this will calculate the function of the last two numbers entered. Pressing another function will then use the next function with the number before the last one used.

Subject 10, No-model group

This description shows the subject understands (1) that numbers are 'stored' by using ENTER, (2) that commands and operations use those numbers to produce 'answers', (3) a clear solution procedure and its consequences ('commands are entered after all the numbers required....') and (4) that using this representation of the internal storage of numbers was of use in developing solutions to the invention problems.

Every number entered into the display unit is stored in its original format until being "acted upon" by a command when it stored as an answer. Commands are entered after all the numbers required for the sequence of calculations have been stored.

Visualizing the sequence of stored numbers helped to solve the more complicated problems when a number is required more than once, but could only be typed into the display unit once.

Subject 11, No model group

This subject's description of the calculator shows (1) that using ENTER stores numbers in a memory, (2) that arithmetic operators act upon stored numbers ('doing

various things to them retrospectively'), (3) that numbers are stored in an ordered list, (4) that this list can be manipulated 'to a certain extent' by using SWITCH.

> By listing input digits (using ENTER) then doing various things to them retrospectively by function keys. SWITCH allows you to hop backwards or forwards in the list (to a certain extent).

Figure 5
Subjects descriptions of calculator functioning

These descriptions clearly show that 5 of the 6 subjects in the No-model group developed a clear model of the calculator memory. This model was surprisingly consistent across all the subjects; the calculator memory seems consistently to be described in terms of a 'line' of stored numbers. This memory 'line' is functionally the same as the memory stack described in the instructions presented to the Model group subjects. Thus we can conclude that if subjects are presented with inadequate knowledge, they will induce the knowledge required to provide the relevant command semantics.

This knowledge of the internal memory stack suggests that with further practice the no-model subjects would be performing as well as the Model group subjects on all problem types. This result is not reported by Halasz and Moran who presented subjects with more than 50 questions. One explanation for this is that Halasz and Moran imposed a time-limit of problem solution development. Given that the more difficult problems required extensive problem solving, one possibility is that by setting a time limit on solution time for each question, Halasz and Moran prevented successful problem solving, and thus prevented procedural group subjects from developing a device model. This explanation is supported by Soar's learning mechanism; learning is dependent on successful completion of problem solving goals. By enabling subjects to problem solve until *they* wished to give up, the experiment presented here ensured that subjects could reason extensively.

There are a number of consequences of this clear model-knowledge development: (1) the results presented by Halasz and Moran (1983) do not describe the development of device-model knowledge. As a result, their theoretical account of the cognitive processing that underlies user performance cannot account for these dynamic changes in the user's representation. The information sources available to the user need to be considered; it is clear from these results that the knowledge No-model subjects used at the end of the experiment for deriving problem solutions was *not* in the instruction booklet presented to them. (2) with respect to the simulation models, it is clear that accounting for user learning requires more than knowledge compilation. Any realistic simulation model will also need to account for these changes in the user's knowledge representation. Again, this requires a wider consideration of the resources available to the user; clearly focussing on the instructions presented is not adequate. In addition, this development of new knowledge has an immediate impact; the learning is within-trial. It was not the case that subjects did not think about the calculator functioning until prompted at the end of the experiment. Verbal protocols revealed that this knowledge developed *and was used* throughout the experiment; subjects problem solved, induced model-related command semantics to explain calculator functioning, and then used this knowledge to construct a coherent model of the internal memory and its relationship to other available actions. Although the simulation models presented in Section 2 dealt with only simple problems, they did clearly demonstrate that Soar is capable of modelling this within-trial learning; chunks are used as soon as they are built.

Discussion and Conclusions

To summarise, a number of hypotheses were derived from the simulation models and were outlined above. The experiments demonstrated: (1) subjects given device-model instructions took longer to solve significantly more problems than subjects presented procedural instructions. This supported the hypothesis that device-model instructions are more computationally costly than procedural instructions. This effect seemed to be particularly strong on certain problem types, possibly due to the extra processing required in these more complex problems. (2) practice effects were clearly observed for both groups. There were considerable reductions in problem solution time after practice. (3) group differences disappeared after practice. (4) keystroke level analyses did not reveal learning at predicted "planning boundaries". It seems, therefore, that the problem solving methods that users create must be local and display-driven. Considering the tasks analysed here, the selection of these methods must be at the level of mapping one or two actions to a bracket or a number in the arithmetic expression. The existence of such local methods explains the flexibility shown by my subjects in solving novel problem types. Of course, using purely local methods leads to difficulties with some problem types, which require more global problem solving, such as arithmetic problems that are centre embedded or complex stack manipulation problems. This was precisely what was observed; subjects in both groups found these problems difficult. (5) No-model subjects developed model-based knowledge when presented with stack-related problems.

I conclude therefore that computational models of device users now need to address the following two issues:

1. local, display-driven methods, rather than AI-like generalised 'parse, plan and execute' methods. These 'parse, plan and execute' methods are typical of problem solving models of interaction, where the problem solution is selected more globally on the basis of problem structure, like my Soar models described above, where the problem was to 'solve a binary calculation'.
2 describing the development of users' mental models showing how both internal and external knowledge is utilised.

A number of Soar models to address the development of proceduralised methods and the development of device model knowledge are currently being implemented. The design of these simulation models incorporates wider resources than the instructions presented, including aspects of the visible interface.

Acknowledgments

I would like to thank Thomas Green for commenting on earlier drafts of this paper.

References

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

Barnard, P. J. (1987). Cognitive resources and the learning of human-computer dialogs. In J. M. Carroll (Ed.), *Interfacing thought.* Cambridge, MA: MIT Press.

Bibby, P. A. (1989) *Knowledge of Devices: The Role and Representation of Mental Models of Devices.* PhD Thesis, University of Lancaster

Bibby, P. A., & Payne, S. J. (1990). *Learning about devices by internalizing instructional descriptions* (Research Report RC 15522). Yorktown Heights, NY: IBM T. J. Watson Research Center.

Brazier, F.M.T (1991) *Design and Evaluation of a User Interface for Information Retrieval*, PhD Dissertation, Free University, Amsterdam.

Brewer, W.F (1987) Schemas versus mental models in human memory. In P. Morris (Ed.), *Modelling Cognition*, John Wiley and Sons Ltd.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Churchill, E.F. and Young, R.M. (1991) Modelling Representations of Device Knowledge in Soar, in *Proceedings of AISB '91*, Luc Steels and Barbara Smith (ed.), Springer-Verlag

Duff, S.C (1990) *Task and Device Representation in the Use of Interactive Systems* Phd Thesis, University of Cambridge.

Duff, S.C, and Barnard, P.J. (1990) Influencing Behaviour via Device Representation; Decreasing Performance by Increasing Instruction, in D.Diaper et al (eds) *Human-Computer Interaction - INTERACT '90*, Elsevier Science Publishers.

Foss, D., Smith, P. and Rosson, M. (1982) *The Novice at the Terminal: Variables affecting understanding and performance*. Paper presented at the Psychonomic Society meeting, Minneapolis.

Halasz, F.G (1984) *Mental Models and Problem Solving in Using a Calculator* Phd Thesis, Stanford University.

Halasz, F. G., & Moran, T. P. (1983). Mental models and problem solving using a calculator. In *Proceedings of CHI '83: Human Factors in Computing Systems*. New York: ACM.

Howes, A. and Payne, S. J. (1990) Display-based competence: towards user models for menu-driven interfaces. *International J. Man-Machine Studies*, 33, 637-655.

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

Laird, J. E., Newell, A., & Rosenbloom, P. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Larkin, J. H. and Simon, H. A. (1987) Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science*, 11, 65-100.

Mayer, R. E. (1976) Comprehension as affected by structure of problem representation. *Memory and Cognition*, 44, 249-255.

Newell, A (1990) *Unified Theories of Cognition*, MA Harvard.

Newell, A., & Rosenbloom, P. (1981). Mechanisms of skill acquisition and the law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

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

Payne, S. J. (1987). Complex problem spaces: Modelling the knowledge needed to use interactive systems. In H. Bullinger & B. Shackel (Ed.), *Human-Computer Interaction: Interact* '87. Amsterdam: Elsevier Science Publishers.

Rouse, W.B. and Morris, N.M. (1986) On looking into the black box: prospects and limits in the search for mental models. *Psychological Bulletin*, 100, 349-363

Wilson, J.R and Rutherford, A (1989) Mental Models:Theory and Application in Human Factors. *Human Factors*, 31(6), pp 617-634

Young, R. M., Green, T. R. G., & Simon, T. (1989). Programmable user models for predictive evaluation of interface designs. In K. Bice & C. Lewis (Eds.), *Proceedings of CHI '89. Human Factors in Computing Systems*. New York: ACM.

Active and latent failures: How counterfactual thinking shifts the focus in causal assessment

Sebastiano Bagnara, Donatella Ferrante°, Paolo Legrenzi°, Mario Malfatti*, Antonio Rizzo*[1]*

*Universita' di Siena & Istituto Psicologia del CNR, Siena, Italy.
°Dipartimento di Psicologia, Universita' di Trieste, Trieste, Italy.

ABSTRACT

Socio-cultural and compensation norms put the responsibility of accidents on front-line people, rather than on those who established the conditions for the accidents. Such norms are the societal and institutional consequences of a very general cognitive bias. In assessing the causes of deleterious outcome, people are biased toward active failures. The present paper reports experimental evidence that counterfactual thinking, i.e., the retrieval or construction of alternatives to experience, can counter this bias, and facilitate people to move from active to latent failures in assessing the causes of an accident.

1. Introduction

In the last years, the research trend on accident analysis has confirmed the prediction that "future studies of human error will need to encompass organizational as well as individual fallibility" (Reason, 1990, pg. 250). Indeed, nowadays, the studies are still paying attention to the technical components and the actions of the individual human operators involved in a accident (i.e., the front-line parts of systems involved in accidents), but also consider organizational and technical design and management principles, practices, and decisions.
The role of socio-technical and organizational factors has been analyzed in many domains: a) Serious and exceptional accidents (e.g., Fennel, 1987; Wilson, 1986; Sheen, 1987). b) "Routinary" accidents in transportation (e.g., car accidents) (Malaterre, 1990) and industrial plants (Kletz, 1990). c) Development of conceptual tools (e.g., Waagenar, et al., 1990), and systems for accident analysis and reporting (e.g., Bagnara, et al., 1989, van der Schaaf, 1991). e) Errors in human-computer interaction (Frese, et al., 1990).

---

[1] This research was supported by grants from CNR Target Project 'Prevention and Control Disease Factors'; sub-project 'Stress'; grant No. 9103615. Address the correspondence to: Antonio Rizzo, Istituto di Psicologia del CNR, viale Marx 15, 00137 Rome, Italy; or Università di Siena, via Roma 47, 53100 Siena, Italy.

On the other hand, cognitive science and its applied counterpart, cognitive ergonomics, have gained a pretty large corpus of knowledge on the basic cognitive processes underlying human error and the interaction with socio-cultural cognitive artefacts (Norman, 1990; Reason, 1990).

Within the new research trend, a comprehensive framework for the analysis of accident causation in complex systems has been developed (Reason, 1990; Wageenar, Hudson, and Reason, 1990), where socio-technical, organizational factors, and cognitive processes are acknowledged as both triggering and root causes of accidents. This framework moves from the distinction between active and latent failures. Active failures are errors and violations that have immediate and visible negative effects, and occur right before an accident. Usually, the people involved in them operate near to the outcomes of the human-machine systems (e.g., pilots, control rooms operators, car drivers, etc.).
Latent failures are organizational and design actions or decisions that yield conditions that may remain silent for a long time, until they combine with triggering conditions and then produce an accident. They are present in the system well before the onset of an accident sequence, and, consequently, may facilitate the adoption of unsafe behavior. Socio-organizational factors play a major role in establishing the conditions for latent failures.

The shift of interest toward organizational factors and latent failures is however very recent. Previously, especially in safety management, the focus was on active failure, exception made for complex systems and accidents with dramatic outcomes (see, e.g., Perrow, 1984). The reasons for this general attitude are to be found in socio-cultural norms and stereotypes, and compensation laws, which tended (and tend) to put the responsibility on front-line people, rather than on those who established the conditions for the accidents, i.e. for the latent failures. Such norms are the social and institutional consequences of a very general cognitive bias: In assessing the causes yielding to a deleterious outcome, people are biased toward active failures (e.g., operator behaviors, technical faults) and under-evaluate the role played by latent failures (e.g., work organization, plant conditions), unless a deep analysis of the causal scenario that yielded to the accident is carried out (see, e. g., Wagenaar, et al. 1990).

A possible candidate to counter this bias, and then facilitate people to move from active to latent failures in assessing the causal role of events in a scenario, may be the use of the counterfactual thinking, that is the retrieval or construction of alternatives to experience (Kahneman and Miller, 1986). It is we ll-known that the concept itself of causation is intimately tied to counterfactual questions, and that counterfactual thinking affects causal assessment (Makie, 1974).

In general, the study of counterfactual thinking tries "to identify the rules that determine which attributes of experience are immutable and which are allowed to vary in the construction of counterfactual alternatives to reality" (Kahneman and Miller, 1986, p. 150). Wells and Gavansky (1989) provided experimental evidence that an event is judged "as causal of an outcome to the extent that mutations to that event would undo the outcome". Recently, it has been claimed (Girotto, Legrenzi, Rizzo, 1991; Ferrante et al. 1992) that the mutability of an event is inversely related to the perceived constraints to that event in a scenario, and directly related to the availability of alternative states to that event compatible with the same perceived constraints. For instance, Girotto, et al. (1991) shown that the human action is characterized by very high mutability, since it is usually: a) Perceived as an expression of free will, independent from external causes, and then characterized by a low degree of constraints. b) Considered as an outcome of a choice between two or more competing behaviors. Whichever the human action, it is in general thought as if constraint-compatible alternatives to it were always available.

However, so far, the role of counterfactual thinking in causal assessment has dealt with the evaluation of a single causal candidate, whereas it should concern the scenario as a whole. The whole web of the perceived relationships among the events in a scenario has to be properly rearranged when an event is mutated. Data (Ferrante, et al., 1992) showed that, when an event is constrained by, for example, social rules or physical states, it is mutated together with its more salient constraints in order to reach a newly-gained stable state of the scenario.

The present paper is aimed at: a) Confirming the idea that the mutability of each event depends on its perceived relationships with the other events in the scenario; and b) Showing that the counterfactual evaluation of accident scenarios should allow to explain accidents in terms of the structural properties of the scenarios they are embedded in, rather than in terms of isolated links or specific enabling conditions in a linear causal chain of events.

2. Experiment

The general hypothesis that counterfactual thinking influences the causal attribution process in direction of the whole structure of the causal scenario has been tested by using a story (adapted by a real event, reported by Gopher, et al. (1990)) about an accident occurred in a chemical plant. The story presented two active failures, a human error and a fault in the equipment, the general characteristics of the scenario in which the accident took place, and the usual work practices. Based on this story, three three different scenarios (Error-Only, Fault-Only, and Error and

Fault) were built. Scenarios were evaluated by the subjects for both mutation and cause, having the order of assessment manipulated.
It was reasoned that the hypotheses would be corroborated if the causal assessment would be affected by a previous counterfactual (mutation-based) assessment of the same scenario. Moreover, it was expected a higher mutability of human behavior and working setting in comparison with Error and Fault, in accordance with the general principle of mutability.

Method

Subjects. 120 male and female university students attending psychology and humanities courses took part in the experiment. They were randomly assigned to one of six groups of equal size (n = 20).

Materials. Three versions of a scenario with a dramatic outcome were constructed. Each version of the story had a different distribution of the experimental events: in version a) Only the Fault was present; in version b) Only the Error was present; and in version c) both Error and Fault were present. The scenarios presented to the subjects had two common ("Work Environment" and "Work Procedures") and two different sections ("Accident" and "What Happened"). The following story was arranged for version c (Error & Fault Scenario) (In the following text Error and Fault are in italics. Of course, there was not such cue in the text presented to the subjects).

Work Environment
The accident took place in a chemical plant (see, Figure 1). The production process requires to combine raw material with an additional chemical in a prescribed quantity into a reboiler and to heat the content of the reboiler until the chemical material in it vaporizes.

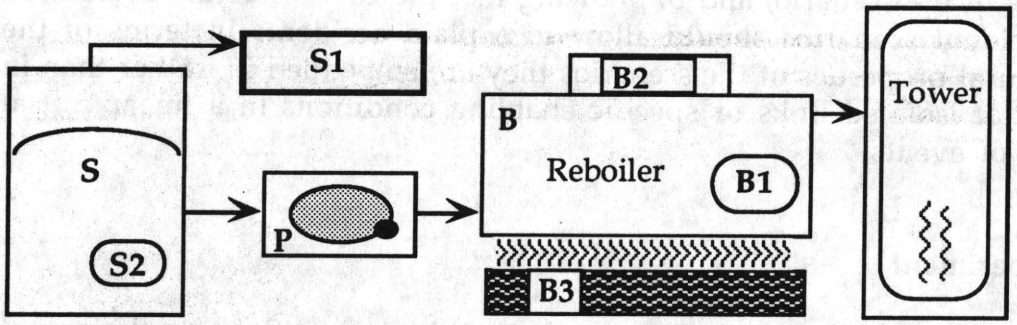


Figure 1. Schematic representation of the chemical plant where the accident took place.

Work Procedure
The operator has to feed the reboiler (B) with a set quantity of the additional chemical, close the tank hatch (B2), open the faucet in the storage tank (S), operate the pump (P), and watch over the transfer of raw material from the storage tank to the reboiler by on-going monitoring of the altimeter (B1). When the altimeter reports that feeding reaches the set quantity, the

pump stops automatically. Along the feeding process, the operator has to set the temperature in the furnace (B3) under the reboiler to reach vaporization of the compound in the reboiler.
In this process, it is important to control the quantity of material that enters the reboiler. This control is performed by the regulator of an altimeter attached to the reboiler, which also serves as an automatic controller stopping the action of the pump (P) after the desired quantity of material has been fed through. It is also important to set: a) the quantity of additional chemical to be entered in the reboiler as a function of the raw material. The additional can be fed into the reboiler either before or after it has been filled with raw material; b) the heating of the compound in the reboiler for its vaporization.
The additional chemical is entered through a hatch (B2) at the top of the reboiler. Beneath the reboiler, there is a furnace (B3). In the case of a malfunction in the automatic altimeter operation, the hatch opening also enables a manual measurement of the level of material in the reboiler. This operation is performed by sticking in the reboiler a measuring rod through the hatch opening.

The Accident
On the day of the accident the operator set the pump into motion in order to feed the raw material (45 tons worth) to the reboiler. At the same time, he started the furnace, *but mindlessly set the temperature too high.* The operator followed on the altimeter the process of filling. He noted that it was taking longer than expected. He suspected, though he could not tell so directly, *a failure in the altimeter.* He verified his suspicion by examining the gauge on the storage tank (A2), and then immediately stopped the pump so as to halt the flow to the reboiler. Then, he climbed on the reboiler in order to assess the quantity of material in the boiler through the opening hatch and possibly add more additional chemical for a proper chemical reaction. As he was opening the hatch, the boiling material overflew from the reboiler and spilled out onto his legs, causing severe burns.

What Happened
The fault at the altimeter produced the exceeding inflow into the reboiler of raw material. Furthermore, a too high temperature in the furnace produced an excessive heating of the compound in the reboiler. All this brought to an early vaporization of the additional chemical and altered the rate raw material/additional chemical imperiling the process. To avoid this the operator had to fill in more additional. But as he was opening the hatch the exceeding raw material and its high temperature produced the overflow of boiling material.

Procedure:. Each participant was a written copy of one of three versions of the story and an answer frame. The versions were randomly assigned to subjects. For each version, half of the subject had to read the story and were asked to rank, firstly, the different ways the events in the story might have been to avoid the outcome, and, secondly, the causes leading to the accident. For the other half, the order of assessment was inverted (causes first, mutation after). Subjects were informed that they could refer back to the story if they wished.

Design:. Since not all the events were present in all the scenarios, the experimental design was divided according to the distribution of the events. There were two 3 (Event: either Fault or Error, Operator behavior and Plant Inadequacy) X 2 (Order of assessment: either Cause or Mutation first), plus one 4 (Event: Fault, Error, Operator behavior and Plant inadequacy) X 2 (Order of assessment) factorial designs.

3. Results

The subjects' responses were analyzed and coded by three independent evaluators. The inter-coders' reliability was higher than 95%. Disagreements were resolved via discussion. Answers were classified according to the following categories: a) human Error, b) Fault of the equipment, c) behavior of the operator after the Error and/or Fault, d) inadequacy of the plant or procedures. Note that in the story was not explicit information about the last category.
For each experimental event, two indexes (mutability and causality) were computed. The events were ranked on the basis of the order the mutations or the causes were reported by the subjects: In the both mutation and cause lists of each subject, rank 1 was assigned to the first experimental event mentioned, the rank 2 to the second, and so on. The rank 4 was assigned to any event that was not mentioned by the subject. Three subjects were discarded from the final analysis because their answers did not allow to identify the order of the mutations (e. g., two o more experimental events were merged).

Data were submitted to two-way ANOVAs (Event X Order of Assessment). Each order of assessment of any scenario was to a one-way ANOVA (Event factor at three or four levels). Post-hoc comparisons were computed by mean of Fisher PLSD. The main results will be presented as a function of the indexes (mutability and causality) and the scenarios.

Mutability.
i) In the Fault-Only Scenario (FOS), the two-way ANOVA showed a significant Event main effect ($p < 0.0001$; $F = 22.44$).

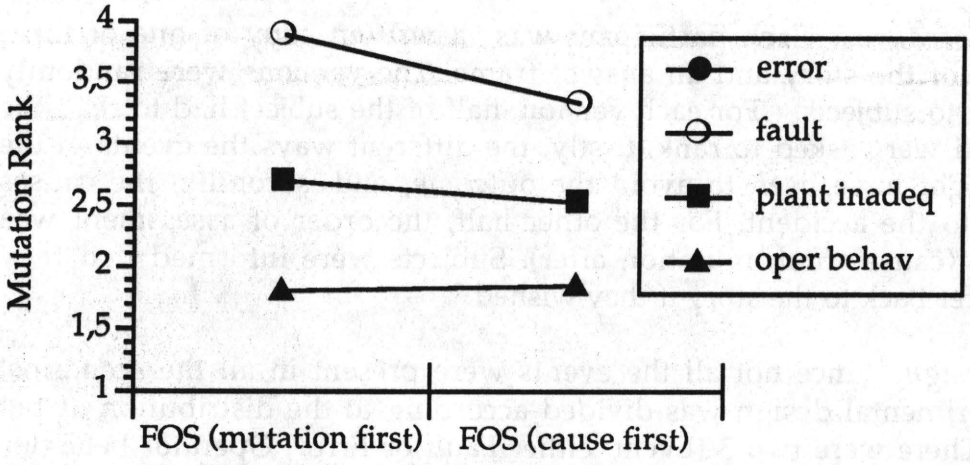


Figure 2. Fault-Only Scenario (FOS): Event x Order of assessment.
(Note that in this and all the following figures: a) A low rank means that the event has been reported first in the list of mutation or causes. b) Mutation first condition requires subjects to list the mutation immediately after the reading of the story, then to list the most relevant causes. c) Cause first condition requires subjects to firstly report the most relevant causes of the accident then to produce the mutations in the story).

The one-way ANOVAs showed a significant effect both for the Mutation-first condition ($p < 0.0001$) and for the Cause-first condition ($p < 0.01$) (see, Figure 2). Post-hoc analyses showed a significant ($p < 0.001$) difference between each event in the Mutation-first condition (Operator behavior ranked > Plant inadequacy > Fault), and only between Operator behavior and Fault in the Cause-first condition ($p < 0.001$).

ii) In the Error-Only Scenario (EOS), the Event main effect ($p < 0.001$; $F = 9.49$) and the interaction ($p < 0.05$; $F = 3.19$) were significant (see, Figure 3). The Mutation-first condition turned out significant ($p < 0.001$). Post-hoc analyses showed that the differences between Plant inadequacy and both Operator behavior and Fault were significant ($p < 0.01$). The comparison for the three levels of the Event factor in the Cause-first condition did not reach significance.

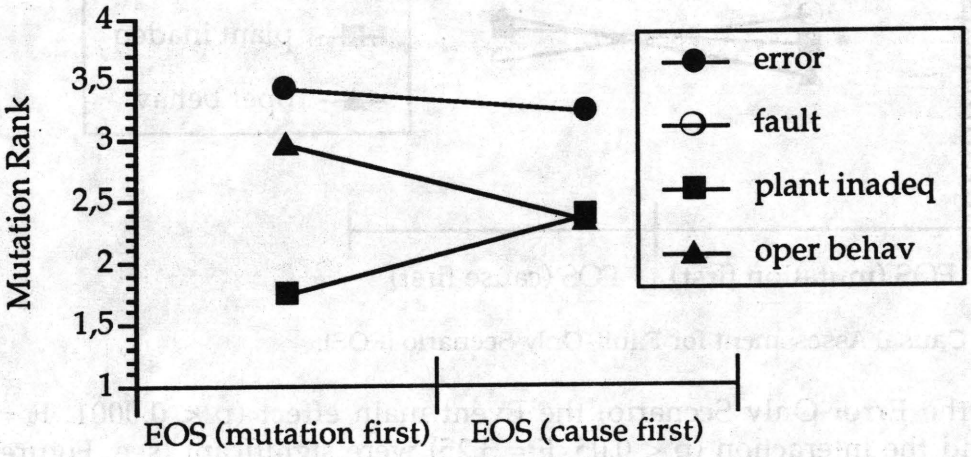


Figure 3. Error-Only Scenario (EOS).

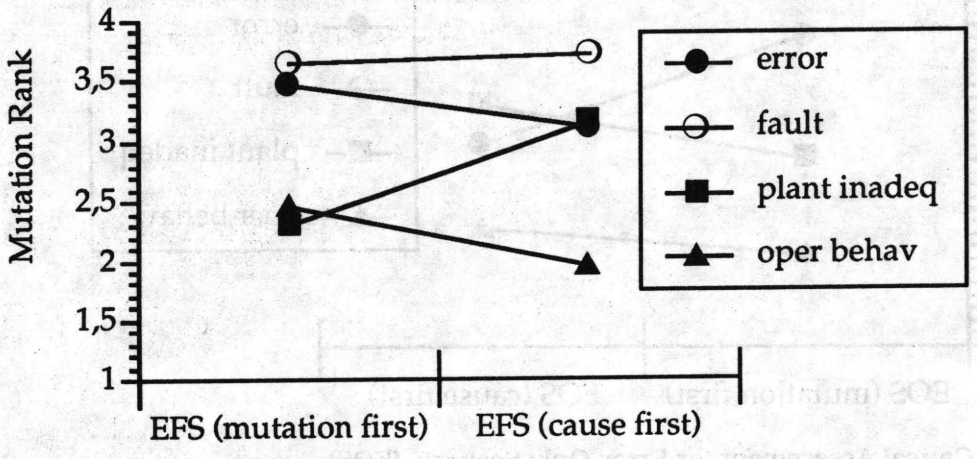


Figure 4. Error&Fault Scenario (EFS).

iii) In the Error and Fault Scenario (EFS), the Event main effect ($p < 0.0001$; $F = 11.99$) and the interaction ($p < 0.05$; $F = 2.74$) were significant

(see, Figure 4). Both the Mutation-first and the Cause-first conditions showed significant differences among the events ($p < 0.001$). Post-hoc analyses revealed: a) a difference between Plant inadequacy and Operator behavior versus Fault and Error for the mutation first condition; b) a difference between operator behavior versus the remaining events for the Cause-first condition.

Causality.

i) With the Fault-only Scenario, nor the main effect neither the interaction were significant (see, Figure 5).

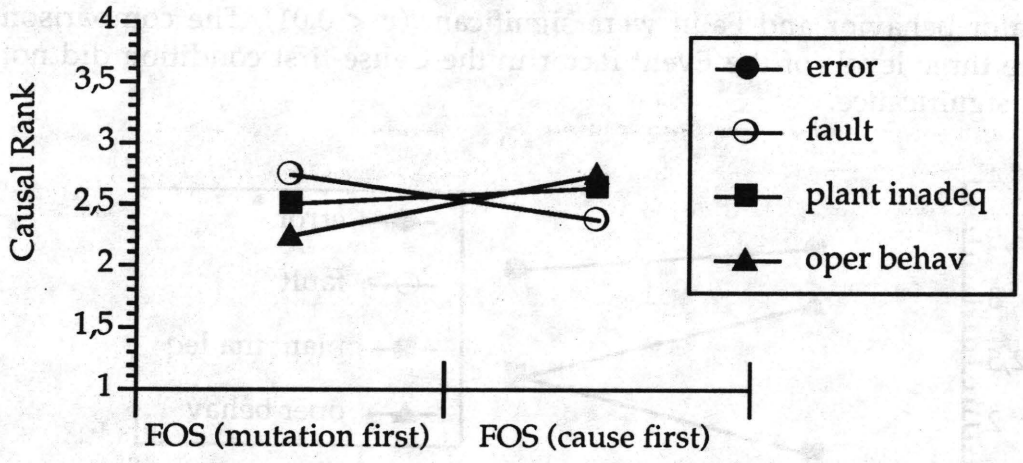


Figure 5. Causal Assessment for Fault-Only Scenario (FOS).

ii) With the Error-Only Scenario, the Event main effect ($p < 0.0001$, $F = 10.63$), and the interaction ($p < 0.05$, $F = 3.25$) were significant (see, Figure 6).

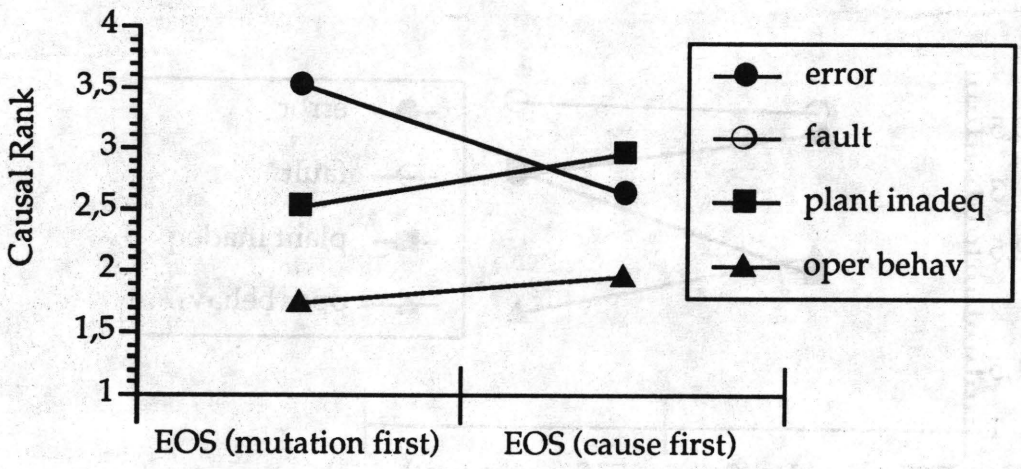


Figure 6. Causal Assessment for Error-Only Scenario (EOS).

The Mutation-first condition and the differences within it turned out significant ($p < 0.0001$, and $p < 0.01$, respectively). The comparison for the three levels of the Event factor in the Cause-first condition did not reach

significance ($p < 0.06$), though the post-hoc analyses showed a significant difference between and Plant inadequacy ($p < 0.05$).

iii) With the Fault & Error Scenario, only the interaction ($p < 0.001$, $F = 5.29$) was significant (see Figure 7). The Cause-first condition turned out significant ($p < 0.001$) with the Error different from both the Operator behavior and the Plant inadequacy ($p < 0.01$), and the Fault different from the Plant inadequacy ($p < 0.01$). The comparison for the four levels of the Event factor in the Mutation-first condition showed no significant difference.

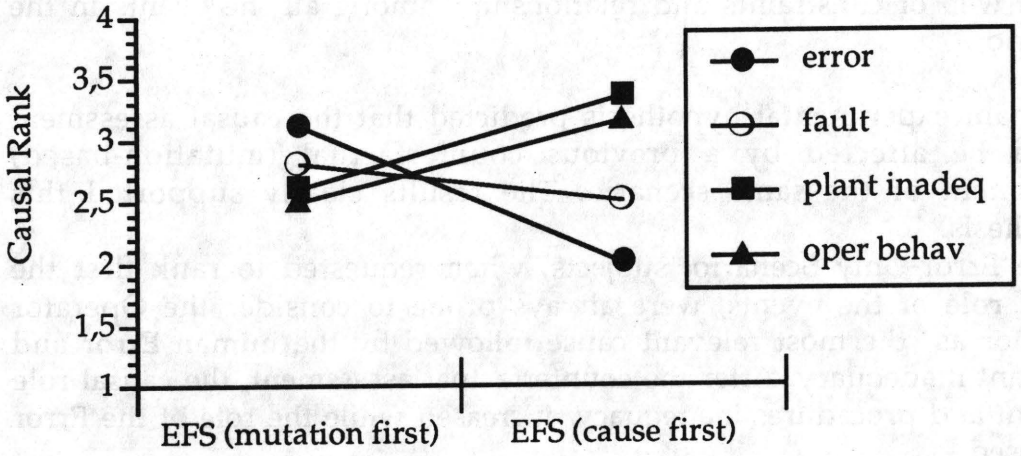


Figure 7. Causal Assessment for Error&Fault Scenario (EFS).

4. Discussion

Overall, the results confirm the principle of mutability and its corollary, which maintains that events under deliberate conscious control (i.e., human actions, when not constrained by social rules or physical states) are more mutable than surrounding events (Girotto, et al., 1991).
Whichever the scenario, Plant inadequacy (considered as expression of management decision and practice), and Operator behavior were more mutated than Error (note, due to an unwilled action, i.e., a slip, which cannot be considered as the outcome of a choice among equally available alternatives) and Fault. In detail, with the Fault-Only Scenario, subjects preferred to modify Operator behavior more than Plant inadequacy (i.e., working conditions) and technical Fault. Operator behavior and working conditions were perceived as less constrained than the technical Fault because related to voluntary decision. A similar pattern of mutations was observed for the scenario where both Error and Fault occurred (the Operator behavior and the Plant inadequacy were more mutated than the Fault and the Error). Whereas, in the scenario where only the Error occurred, the Plant inadequacy were significantly more mutated than both the Operator behavior and the Error. This unexpected pattern can be explained by analyzing the kinds of mutations produced by the subjects.

Besides the mutations concerning generic improvements of the safety conditions of the plant, there were many specific mutations about the introduction of devices signalling the error. By considering such an outcome and the inverse trend present in the Only-Fault Scenario, it may be speculated that subjects assumed that, when one of the two sides in a human-machine system fails, the other should enable a recovery. This speculation indicates that the relationships among error, recovery and context should be further investigated.
In general, the present evidence corroborates the idea that mutability of an event does not depend on the causal chain only, but is affected by the whole web of constraints and relationships among all the events in the scenario.

The main experimental hypothesis predicted that the causal assessment would be affected by a previous counterfactual (mutation-based) assessment of the same scenario. The results clearly supported this hypothesis.
In the Error-Only Scenario, subjects, when requested to rank first the causal role of the events, were always prone to consider the Operator behavior as the most relevant cause followed by the human Error and the Plant inadequacy. After the counterfactual assessment, the causal role of Plant and procedures inadequacy increased while the role of the Error decreased.
In the scenario where both the Error and the Fault were present, the effect of counterfactual thinking was even stronger. When the causal assessment was performed before the counterfactual thinking, the subjects produced a ranking of causes where the Error was by large the first, followed by the Fault, Operator behavior and Plant inadequacy, respectively. After the counterfactual assessment, there was no significant difference in the ranking among the events, and the order presented an inverted trend. In the Fault-Only Scenario, the trend was consistent with the other two scenarios, but there was no significant effect.

It is important to note that the characteristics of the plant, even if not explicitly reported in the story, played a major causal role after the counterfactual evaluation. It allowed to identify events and states not explicitly presented in the scenario, which however might actually play a role in the production of the outcome. These states are thought as binding, though hidden and implicit, attributes and properties of scenario, and strongly influence the propensity of a scenario to assume alternative configurations (see, Kahneman and Varey, 1991). Indeed, the counterfactual thinking deals with all (either explicit or implicit) alternative states of the world that might have occurred. The mutability of the events in a scenario ant its propensity to swing in an alternative state are strongly interwoven. It can be concluded that the counterfactual thinking is instrumental in making apparent both the hidden and the global features of a scenario.

6. Conclusion

The above presented and discussed results show that naive people can move from active to latent failures when they make use of counterfactual thinking in assessing the causal role of the events in a scenario yielding to a deleterious outcome. This effect can be exploited in accident data collection and analysis. An increasing number of systems for reporting accidents rests on self-reports of accidents and on individuals' causal assessment and suggestions for improving the safety conditions. The data produced by these reports are usually assumed as very relevant: yet they can be strongly biased toward active failures. The adoption of simple questions concerning the undoing of the negative outcome can facilitate the analysts to move from the active to latent failures and to "discover" the hidden events and structural properties which may have played a substantial role in producing the accident.

References:

Fennel, D. (1988). Investigation into the King's Cross underground fire. London, Department of Transport: HMSO).

Ferrante, D., Rizzo, A., Legrenzi, P., Girotto, V. (1992) Social and physical constraints in counterfactual thinking. Technical Report 92/2. University of Trieste: Dept. of Psychology.

Frese, M., Brodbeck, F. C., Zapf, D., Prünper, J. (1990) The effects of task structure and social support on users' errors and Error handling. In: Diaper, D., Gilmore, D., Cockton, G., Shackel, B. (eds.) Proceedings Interact'90. Elsevier, Amsterdam.

Girotto, V., Legrenzi, P., Rizzo, A. (1991) Event controllability in counterfactual thinking. Acta Psychologica. 78, 111-133

Kahneman, D. , Varey, C. A. (1991) Propensity and counterfactual: The looser that almost won. Journal of Personality and Social Psychology, 59, 1101-1110.

Kletz, T. A. (1990). Critical Aspects of safety and loss prevention. Guildford, Butterworth-Heinemann.

Mackie, J.L. (1974). The cement of the universe: A study of causation. Oxford, Clarendon Press.

Malaterre, G. (1990) Error analysis and in-depth accident studies. Ergonomics, 33, 1403-1421.

Perrow C. (1984) Normal Accidents. Basic Books, New York.

Reason, J. (1990). Human Error. Cambridge, MA, Cambridge University Press.

Sheen, J. MV (1987). Herald of free enterprise. Report of Court N° 8074 Formal Investigation. London, Department of Transport.

van der Schaaf, T.W. (1990). An integral approach to safety management. Tech. Report EUT/BDK/42. Eindhoven, University of Technology.

Wells, G. L., Gavanski, I. (1989). Mental simulation of causality. Journal of Personality and Social Psychology, 56 ,161-169.

Wilson, R. (1986). Chernobyl Report: Coping with the human factor. Nature, 234, 25-30.

Strategies, errors and display-based skills in a complex task

Simon P. Davies

Department of Psychology,
University of Nottingham,
University Park, Nottingham, NG7 2RD, UK

ABSTRACT

The paper considers the role of working memory in the execution of a complex skill. More specifically, it is concerned with the relationship between working memory and the development of expertise and the role played by external memory sources and display-based problem solving in computer programming tasks. Two experiments are reported which demonstrate that the development of expertise in programming does not appear to depend upon an increase in working memory capacity or availability. This finding is unexpected given the importance placed upon working memory capacity in other theoretical accounts of the development of complex skills. These experiments provide evidence for an alternative view which suggests that expertise in programming may be dependent upon the development of strategies for efficiently utilising external displays for the purpose of recording intermediate states and partially formed solution steps. In this context, it appears that novices rely extensively upon working memory to generate as much of a solution as possible before transferring it to an external source. In contrast, experts engage in problem solving behaviour which is characterised by the extensive use of an external display as an information repository. This gives rise to a pattern of generation behaviour which manifests itself in terms of a closely linked cycle of code generation and evaluation activities. One of the most striking results of this work is that when experts are unable to use an external display to support facets of this activity then their performance deteriorates to the level typically exhibited by novices. These results are discussed in terms of a framework which emphasises the role of display based-problem solving and its contribution to strategy development. Finally, the implications of this work for our understanding of errors and for the design of programming environments are briefly discussed.

1. Introduction

A common finding in recent research into the cognitive aspects of programming is that code is not generated in a linear fashion - i.e., in a strict first-to-last order (Davies, 1991; Rist, 1989). Rather, many deviations are made from linear development, where programmers leave gaps in the emerging program to be filled in later. Green et al (1987) have proposed a model to account for this finding. Their Parsing/Gnisrap model introduces a working memory component into the analysis of coding behaviour which forces the model to use an external medium (eg the VDU screen) when program fragments are completed or when working memory is overloaded. This means that programmers will frequently need to refer back to generated fragments in order to recreate the original plan structure of the program which may have only been partially

implemented in code. The parsing element of the model describes this process, while gnisrap (the reverse of parsing) describes the generative process.

Davies looked at the nature of the nonlinearities in program generation for programmers of different skill levels. One finding to emerge from this work was that experts perform a greater number of between-plan jumps than novices and that novices tend to perform more within-plan jumps - that is, adopt a linear generation strategy. This might seem anomalous, since if we assume that the re-parsing of a generated output involves some cognitive cost, then one might expect the development of programming skill to be partly dependent upon a programmer's ability to generate as much of the program internally before writing it to an external source, thus reducing the need to re-parse. However, the opposite appears to be the case. The results of Davies (1991) suggest that skilled programmers make much use of external memory sources (i.e., a VDU screen) while novices tend to rely upon the use of internal memory to develop as much of the solution as possible before transferring it to external memory.

2. External memory and display-based performance in problem solving

The parsing/gnisrap model displays several important commonalities with emerging models of planning and problem solving in other complex domains. In particular, this model stresses the role played by external memory sources as information repositories which can be used to record intermediate problem solving states. In a similar vein, work connected with the development of planning models in Artificial Intelligence (Agre and Chapman, 1987; Ambros-Ingerson,1987) has suggested that actions are often enacted before plans or problem solution sequences are complete. In formulating this alternative account of planning, these models stress the inexorable link between the planning process and the execution of plans.

More recently, the role of external memory sources as repositories for search control knowledge and intermediate state information has gained prominence in a number of accounts of problem solving (Greeno, 1989; Larkin, 1989). For instance, Larkin (1989) has proposed a production system model of human 'display-based' problem solving (DiBS) where the system's working memory is divided into two kinds of elements: those reflecting the problem solver's internal goals and those representing features of external real world objects. As in other production system models, specific productions are executed when their preconditions are satisfied in working memory (Anderson, 1983). The associated actions of these productions then act in turn to modify the content of working memory. In terms of Larkin's model, these changes can arise in two ways - either as changes to the physical world or as changes to the problem solver's internal representation of goals and subgoals.These approaches suggest that problem solvers may develop strategies which place reliance upon the use of an external display rather than upon internal sources. Indeed, Howes and Payne (1990), suggest that "becoming skilled with a system does not necessitate learning a set of packaged methods. Rather, skilled users become good at getting the right information from the display when they need it, in the service of performance" (p 653).

These accounts of problem solving and planning are clearly very similar in their general form to emerging accounts of problem solving and planning in the programming domain. For instance, a central feature of the parsing/gnisrap model is the idea that code generation involves two fundamental psychological processes; one in which the external structure (program code) is created from the internal (cognitive) structure that represents the problem requirements and an inverse process by which this internal structure is recreated when necessary from the external structure.

Similarly, Gray and Anderson (1987) stress the importance of the evaluation episodes that are frequently seen to occur during code generation. The existence of these evaluative activities

presumably implies that programmers are able to extract relevant information from the code that they have already generated in order to inform their subsequent problem solving activity. This will necessitate re-parsing the code and then matching it with an internal representation of plans and goals. A similar distinction between evaluative and generative activities in statistical problem solving has been proposed by Allwood (1984). Hence, in terms of these models of problem solving, the external display become a central repository for intermediate state information when working memory is loaded.

One question that arises in the present context relates to the extent to which expertise in programming and in other complex skills can be explained by recourse to an extended working memory model as opposed to a model which places emphasis upon the role of externalised memory structures and display-based comprehension? The following experiments attempt to address this issue directly. The first experiment considers the role of working memory in the determination of strategy for novice and expert programmers. The second experiment looks at the effects upon certain error forms of restricting the kinds of manipulations programmers can make within an environment.

3. Experimental Studies

In the first experiment, subjects carried out a simple articulatory suppression task while engaged in a program generation activity. If working memory limitations cause programmers to make use of an external medium, as suggested by Green et al, then the act of loading working memory through a concurrent task should give rise to an increase in nonlinearities, since subjects would have to engage more fully in the parsing/gnisrap cycle in order to make use of the external display.

The second experiment looks at the way in which restricting the use of an external medium affects performance. Here, if programmers are not able to correct already generated code at later stages in the coding process, then this should have some effect upon their performance. In this experiment, subjects created a program using a full-screen editor that provided no opportunity for the revision of existing text. The use of such an editor clearly places a significant load upon a subjects working memory capacity since they will be required to internally generate as much of the program as possible before externalising it. By placing emphasis upon the use of working memory it should be possible to induce error prone behaviour which parallels that evident when working memory is loaded in other ways, for instance via articulatory suppression.

We might expect a detrement in expert performance when the device used to create the program is restricted in such a way as to make retrospective changes impossible. This is based upon the assumption that experts make greater use of external sources to record partial code fragments which are then later elaborated. Conversely, it has been suggested that novices will tend to rely more upon generating as much of the program internally before writing it to an external source. It is clear that these strategic differences will be supported to a greater or a lesser extent by the device used to create the program. Hence, for expert programmers, it might be suggested that restricting the device will cause them to revert to a novice strategy, since they will then be unable to use the external display in the normal way.

Establishing support for this hypothesis would have a number of implications. Firstly, it would suggest that the development of expertise may not be based simply upon the acquisition of knowledge about a given domain. If this were the case, we would expect experts to perform better than novices regardless of the constraints imposed by the task environment. Secondly, it would indicate that increased working memory availability does not necessarily lead to better performance. Moreover, if working memory availability is correlated with expertise, then

experts should perform better that novices in situations where they must rely upon internal sources. If this is not the case, then we might question the status of working memory in theories dealing with the development of complex skills. An alternative explanation is to argue that experts have developed particular strategies for dealing with task complexity that involve close interaction with external information repositories in order to record partial solution fragments as they are generated. If novices have failed to develop such strategies, then it is unlikely that their performance would be affected significantly by restricting the task environment.

This analysis can be extended by classifying the errors in the programs generated by subjects. A scheme devised by Gilmore and Green (1988) suggests four main categories of error:

1 - Surface level errors caused mainly by typing and syntactic slips: (e.g. confusion between < and >, missing or misplaced quotes etc).
2 - Control-Flow errors: (e.g. missing or spurious else statements, split loops etc).
3 - Plan-Structure errors: (e.g., guard test on wrong variable, update wrong variable etc.)
4 - Interaction errors: Errors occurring at the point where structures of different types interact: (e.g. a missing 'Read' in the main loop, initialisations within the main loop).

It is clear that some of these errors can be categorised as knowledge-based (specifically, plan-structure errors) while others will be dependent upon working memory limitations. For example, both control-flow and interaction errors, since they depend upon establishing referential links and dependencies between code structures, are likely to be affected by working memory constraints. In terms of the first experiment, we might expect both control-flow and interaction errors to predominate in novice solutions where working memory availability is reduced. In the case of experts, it is argued that the interactions between code structures will be evaluated in the context of an external memory source. That is, by re-parsing existing code fragments in order to reconcile them with the code the programmer is currently working on. Thus, that the act of loading working memory should not affect the occurrence of these types of error.

In the case of the second experiment, we would expect the converse. If experts are not able to use the external display in the manner predicted, then it might be hypothesized that interaction and control-flow errors will predominate in the condition where use of the device is restricted. It might also be predicted that this experimental manipulation will not affect the occurrence of plan-structure errors since these are hypothesized to be knowledge-based rather than strategy-based.

3.1 Experiment 1. Effects of articulatory suppression on strategy and errors

3.1.1 Method

Subjects: Twenty subjects participated in this experiment. One group of ten subjects consisted of professional programmers. All the subjects in this group used Pascal on a daily basis and all had substantial training in the use of this language. Members of this group were classified as experts. A second group consisted of second year undergraduate students all of whom had been formally instructed in Pascal syntax and language use during the first year of their course. Members of this group were classified as novices.

Procedure and Design: Subjects were asked to carry out a simple articulatory suppression task which involved repeating a string of five random digits. At the same time, subjects were requested to generate a simple pascal program that could read a series of input values, calculate a running total, output an average value and stop given a specific terminating condition. This

specification was derived from Johnson and Soloway (1985) and was chosen because it has formed the basis of many empirical studies. Hence, the resulting programs could easily analysed for errors and plan structures.

Error Type	Experiment 1(Suppression)				Experiment 2 (Restricted Env.)			
	No Suppression		Suppression		Non Restricted		Restricted	
	Novice	Expert	Novice	Expert	Novice	Expert	Novice	Expert
Surface	19	28	10+	25	21	29	21	11*+
Plan	36*	22	15+	25	40*	22	38*	9*+
Control	24	25	34*	23	21	26	19	33*
Interaction	21	25	41*	27	18	23	22	47*+

Table 1. Number of errors on task completion for experiments 1 and 2 showing significant differences at the 0.05% level within (*) and between (+) conditions.

Subjects were allowed to study the specification for 5 mins and were then asked to generate a program corresponding to this specification while engaged in the concurrent suppression task. The subjects were given 15 mins. to complete this task, typing their solutions onto a familiar text editor. Subjects' keystrokes were recorded for further analysis. This analysis provided an indication of the temporal sequence in which programs were generated. Three independent raters were asked to analyse all the resulting program transcripts for the presence of common plan structures (Soloway and Ehrlich, 1984) and for errors (using the classification described above). Within and between-plan jumps were defined as follows: Within-plan jumps were classified as movements between a particular line of the program text to another line which formed part of the same plan structure. Between-plan jumps were defined as movements from the current line to lines within different plan structures (see Davies, 1991). These protocols applied only to situations where the jump was followed by an editing action. The experiment was a two-factor design, with the following independent variables: 1. Articulatory suppression/No suppression and 2. Level of expertise (Novice/Expert).There were two dependent variables: 1. The number of Between/Within-plan jumps and 2. Errors remaining in the final program

3.1.2 Results

Plan-jumps: Figure 1 shows the number of within and between-plan jumps performed by novice and expert programmers in the two experimental conditions. Analysis revealed main effects of suppression ($F_{1,72} = 8.47$, $p<0.01$) and expertise ($F_{1,72} = 12.56$, $p<0.01$) on jump-type and a more complex interaction between suppression and expertise ($F_{1,54} = 4.73$, $p<0.05$). A number of post-hoc comparisons were carried out using the Newman-Keules test with an adopted significance level of $p<0.01$. This procedure indicated that experts produced significantly more between plan jumps than novices in the non-suppression condition. Conversely, novices produced a greater number of within plan-jumps in this condition. In the case of the suppression condition, there were no significant differences.

Errors: Figure 2 shows the total mean number of errors remaining in the programs on task completion for novice and expert subjects in the two experimental conditions. Analysis revealed a main effect of expertise ($F_{1,36} = 9.37$, $p<0.01$) and suppression ($F_{1,36} = 4.54$, $p<0.05$) and an interaction between these two factors ($F_{1,36} = 15.89$, $p<0.01$). Once again a number of post-hoc comparisons were carried out using the Newman-Keules test with an adopted significance level of $p<0.01$. This indicated a significant difference in error rates in the both experimental conditions when comparing the novice and expert groups. In addition, a significant difference between error rates across conditions was evident for the novice group. In the case of the expert group the same comparison proved not to be significant.

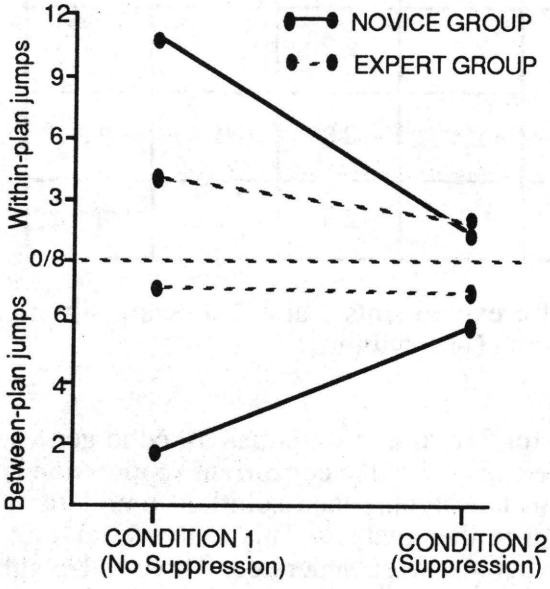


Figure 1 Within and Between-Plan jumps by novices and experts during the first experiment

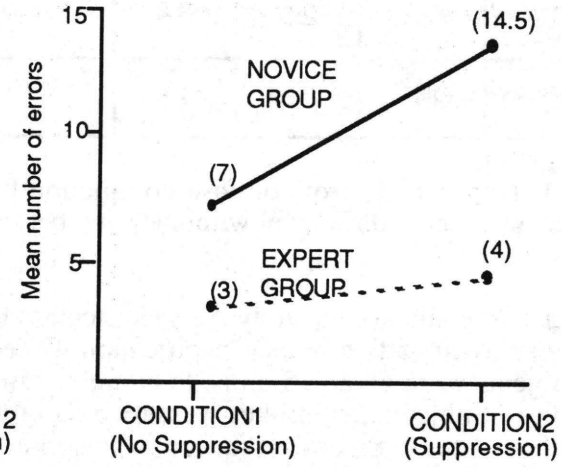


Figure 2 Mean number of errors in experiment 1 for novice and expert subjects

Error classification analysis: In the case of experts, there is a fairly even distribution of error types across the two experimental conditions (See Table 1). Indeed, further statistical analysis revealed no significant differences between error types both within and between conditions (multiple t-tests). In the case of the novice group, the distribution of error types is less straightforward. In the non-suppression condition, novices produced a significantly greater number of plan errors in comparison to the other categories (t-test). Moreover, the only significant difference between the novice and experts groups in this condition was the number of plan errors produced by the novice group (t-test). In the second condition, the distribution of errors across classification types for expert subjects was again fairly even. No significant differences between any of the error classifications were evident. For the novice group, significantly more control-flow and interaction errors were evident in comparison to the other two error classifications (t-test). Moreover, for the novice group,the number of plan errors occurring in the second condition was significantly less than in the first condition (t-test).

Discussion: This experiment shows that expert performance in programming tasks is not significantly affected by articulatory suppression. Hence, for experts the number of errors produced is not significantly different comparing the suppression condition to the non-

suppression condition. Moreover, it appears that strategy is similarly unaffected. Hence, the prevalence of between-plan jumps in the non-suppression condition for the expert group is not diminished in the suppression condition. Similarly, the occurrence of within-plan jumps does not differ significantly in the two experimental conditions.

Conversely, the novice group produced significantly more errors in the suppression condition when compared to the non-suppression condition. In addition, the nature of the coding strategy that they adopt is also affected. In particular, it appears that novice programmers revert from a linear generation strategy characterised by the prevalence of within-plan jumps, to a strategy more characteristic of experts. That is, to a strategy which reflects a greater number of between-plan jumps.

Earlier it was stated that expert programmers appear to rely much more extensively than novices upon the use of external sources to record partial code fragments and that the act of loading working memory or of otherwise reducing its availability would not affect this process. It was suggested that experts will tend engage in very closely linked cycles of planning, subsequent code generation and evaluation. Since it is posited that this process relies little upon the programmer's working memory capacity it is reasonable to expect that articulatory suppression would not affect the nature of performance in the context of this task. The results of this experiment provide support for this view. Further support for this view is evident in the error data. In the non-suppression condition, novice subjects are clearly more error prone than experts. This finding is not unexpected. However, in the suppression condition, the error rate for the expert group changes little from this base line whereas the novice error rate more than doubles. This may indicate that when working memory is loaded novices must externalise information and that this constitutes a strategy which they find unnatural, thus leading to an increased error rate.

A more detailed analysis of these errors reveals a change in the nature of errors for novice subjects between the two experimental conditions. In the non-suppression condition, the novice group make a greater number of plan errors, suggesting knowledge-based difficulties. Conversely, in the suppression condition a greater proportion of control-flow and interaction errors are evident. In terms of the present analysis, the preponderance of control-flow and interaction errors may reflect problems keeping track of the interdependences between elements in the emerging program. When working memory availability is reduced it appears that novices experience some difficulty with these interdependences. Unlike experts, it appears that novices cannot use the external display as an aid to memory to its full extent.

An alternative explanation for these findings is that experts simply have an extended working memory capacity (Chase and Ericsson, 1982; Staszewski, 1988). Such an account would presumably have no difficulty predicting the results of the experiment reported above. In order to assess the cogency of this alternative explanation, the second experiment reported in this paper adopts a different approach for exploring the relationship between working memory and the development of programming skill. In particular, if experts, for whatever reason, are able to extend their effective working memory capacity or increase its availability in other ways then restricting the task environment should not significantly affect their performance.

3.2 Experiment 2. Effects of restricting the task environment

The second experiment is complementary to the first. Whereas the first experiment attempted to reduce the subjects' available working memory capacity, this experiment has been designed to encourage subjects to rely upon working memory. Hence, if experts have an extended working memory capacity they should demonstrate performance equitable to that displayed in the first experiment. Moreover, if the extended capacity notion is correct, then experts should out

perform novices even in the situation where the task environment is severely restricted as in this second experiment.

3.2.1 Method

Subjects, Procedure and Design: The same subjects took part in this experiment, with the order of participation randomised. Subjects were asked to produce a program corresponding to a brief specification which involved processing simple bank transactions. Here, the nature of the task environment formed the basis for the two experimental conditions. In one condition, subjects used a familiar full-screen text editor. In the second condition subjects used a modified version of the same editor,which allowed only restricted cursor movement. That is, from the top of the screen to the bottom. and only between adjacent lines. Once a subject had generated a line and pressed the return key, they were unable to then return to that line to perform other editing operations. The editor did however allow edits to the current line being generated. Subjects first participated in a 5 min. familiarisation session, where the basic modifications to the editor were described. Subjects were then asked to attempt to generate a program from the specification and were asked to check each line of their program before pressing the return key, in order to determine whether they were satisfied with their response. 15 mins were allowed for this task. This experiment was a two-factor design with the following independent variables: Environment (restricted/unrestricted) and Level of expertise (Novice/Expert). In this case the dependent variable was the number of errors remaining in the final program.

3.2.2 Results

Errors: The results of this experiment are shown in figure 3. These data were analysed using a two-way analysis of variance with the following factors; Environment (restricted or unrestricted) and Level of expertise. This analysis revealed a main effect of Environment ($F_{1,36} = 5.74$, $p<0.05$), a main effect of Level of expertise ($F_{1,36} = 4.21$, $p<0.05$) and an interaction between these two factors ($F_{1,36} = 9.76$, $p<0.01$). Post-hoc comparisons were carried out using the Newman-Keules test with a significance level of $p<0.01$. This analysis revealed a significant difference between the number of errors produced by novices and experts in condition 1.

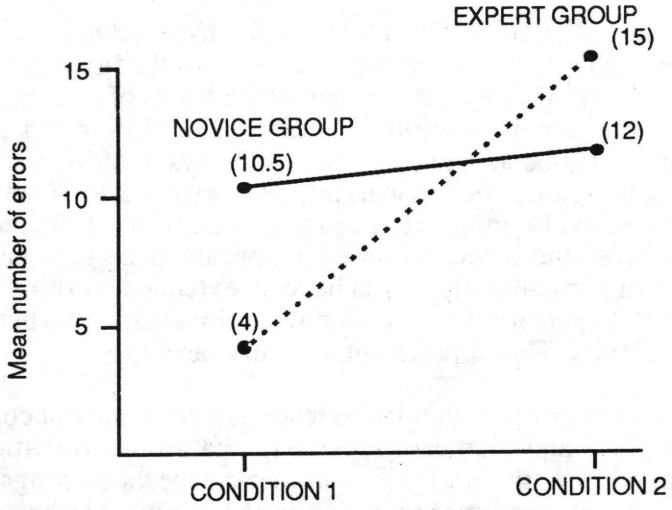


Figure 3 Mean number of errors in experiment 2 for novice and expert subjects

Error classification: The resulting program transcripts were analysed according to the classification scheme described previously. In the case of experts, there were no significant differences between error types within this condition (t-tests). For the novice group, the distribution of error types in the first condition suggests a greater proportion of plan errors in comparison to the other categories (t-test). In the second condition, the distribution of errors across classification types for expert subjects was more complicated. This showed a greater proportion of control-flow and interaction errors compared to the other classifications (t-tests). In addition, experts produced significantly more control-flow and interaction errors in comparison to the first condition (See Table 1).

Discussion: These results provide a striking demonstration of the effects of restricting a task environment. We have argued above that experts rely to a great extent upon using the external display to record fragments of code that are then further elaborated at subsequent points during the generation process. This led to the hypothesis that if programmers were unable to return to previously generated fragments then they would be forced into a situation where they would have to rely extensively upon working memory. However, it appears that while novices are seemingly unaffected by changes to the task environment, experts not only perform worse than novices but also produce the kinds of errors that are indicative of an inability to internally construct links and dependencies between code structures. These results reveal that experts produce more errors than novices in the restricted task environment. Moreover, experts produce a significantly greater number of control-flow and interaction errors in this second condition.

It was suggested previously that the first experiment that the results might be interpreted as indicating that experts have an extended working memory capacity. However, if this is the case then the results of this second experiment would appear to be rather anomalous. If we assume that experts have an extended working memory capacity in comparison to novices, then we might expect that situations which cause experts to rely upon working memory would not give rise to such an extensive decrement in performance. Moreover, in terms of this view there appears to be no reasonable explanation as to why experts produce many more control-flow and interaction errors in comparison to novices.

A more cogent explanation for these findings might simply involve suggesting that experts rely upon external sources and are not able to efficiently revert to a strategy that demands extensive reliance upon working memory. This would account for both sets of experimental findings. In the first experiment a reduction in working memory availability did not affect expert performance. This could clearly be accounted for in two ways. On the one hand, it could be argued that experts simply have an extended working memory capacity. Conversely, we might claim that experts rely extensively upon external sources and find it difficult to adopt other alternative strategies. However, the second experiment appears to suggest that the first of these explanations is incorrect. In particular, if experts have an extended working memory capacity then we would expect them to perform better than novices in situations where a reliance upon working memory is necessitated. This appears not to be the case.

Another finding relating to this data was that in the restricted environment condition the expert group produced fewer surface and plan errors. An explanation for this may be that, in the restricted environment condition, the normally automatic aspects of programming skill are disrupted. This may lead the programmer to attend to the knowledge-based components of programming skill leading to a reduction in surface and plan-based errors. There is evidence in the literature which suggests that so called 'skill' and 'knowledge-based' errors are to some extent disassociable (Reason, 1979).

3.3 Conclusions

These experiments have a number of general implications. Firstly, it appears that experts rely upon external sources to record code fragments as these are generated and then return later, in terms of the temporal sequence of program generation, to further elaborate these fragments. It has been suggested that a major determinant of expertise in programming may be related to the adoption or the development of strategies that facilitate the efficient use of external sources. The externalisation of information clearly has a high cost in terms of the reparsing or recomprehension of generated code that is implied. Hence, it might seem counter intuitive to suggest that problem solvers will tend to rely upon this kind of strategy rather than upon a strategy which involves the more extensive use of working memory. However, this explanation is consonant with existing work which has implicated display-based recognition skills in theoretical analyses of complex problem solving (Larkin, 1989). The contribution of these analyses has been important, but they have neglected to consider the relationship between display use and expertise and the consequent effect that this may have upon the nature of problem solving strategies.

3.3.1 Display-Based Problem solving and errors

The work reported here also poses implications for the way in which we might attempt to explain the occurrence and distribution of error types. In particular, it is clear that a certain classes of error can be attributed to working memory limitations and that such errors are not distributed at random. In terms of the error classification employed here, it appears that interaction and control flow errors predominate in situations where working memory availability is reduced. Previous work (Anderson, 1989) suggests that errors arising from working memory failures will occur at random. However, the results of the studies presented here suggest that working memory related errors may have a more systematic distribution, and that the type of errors one might expect to occur may to some extent be predictable.

Another finding relating to the error data was that in the restricted environment condition the expert group produced fewer surface and plan errors than in the unrestricted condition. One possible explanation for this is that, in the restricted environment condition, the normally

automatic procedural aspects of programming skill are disrupted. It is possible that this may cause the programmer to focus attention upon the knowledge-based components of programming skill leading to a reduction in surface and plan-based errors. There is evidence in the literature which suggests that so called 'skill' and 'knowledge-based' errors are to some extent disassociable (Rasmussen, 1983; Reason, 1979;1990) and it is possible that the restricted environment led to an increased awareness of knowledge-based errors. Moreover, in common with the findings of the present study, one would expect this effect to occur only in the context of expert performance where the procedural aspects of programming skill are likely to be subject to automatization.

3.3.2 Language features and task environments

Gilmore (1986) has criticised Anderson and Jeffries' analysis of working memory errors from a rather different perspective. Gilmore suggests that their analysis "is very weak and its main impact lies in the new approach, rather than the detailed analysis. The main weakness is that language features do not seem to be considered relevant to the analysis. Anderson and Jeffries make no attempt to analyse the causes of processing overload ..." (p 528-529).

This criticism is pertinent to the present analysis since the nature of display-based problem solving in programming will be highly dependent upon features of the particular programming language considered. For example, Green (1990) suggests that some programming languages are "viscous" in that they are highly resistant to local modification. For instance, adding a line to a Basic program may involve renumbering other lines such that the correct control flow is maintained. In terms of the analysis presented here, less viscous languages will provide better support for the kind of incremental problem-solving processes that are proposed. Hence, we might predict that programmers using different languages will make different kinds of error. In addition, since experts appear to employ an incremental strategy and novices a characteristically linear strategy, then it could be argued that some languages may be more suited to experts and others to novices.

The language features described above will affect the strategies employed in the generation of code. However, there are other language features which will affect its comprehension. Gilmore and Green (1988) suggest that some languages are "role-expressive" (for example, Pascal) in that they may contain a rich source of lexical cues which enable a programmer to distinguish more easily the constituent structures contained in a program written in that language. They contest that less role-expressive languages (for instance, Prolog) are lexically more amorphous and that such languages will not facilitate certain forms of comprehension. However, it also appears that this effect may be dependent upon the prior experience of the programmer, since only those trained in program design appear to be able to benefit from cues to plan structure (Davies, 1990).

More recently, claims have been made about object-oriented languages which may provide support for the present analysis. For example, Rosson and Alpert (1990) have claimed that such languages facilitate decomposition of the problem space by enabling programmers to develop encapsulated chunks of code whose internal operations are effectively isolated from other chunks. They go on to claim that this form of decomposition will increase the amount of information that can be held in working memory since objects in the problem space can be held as separable chunks whose lower order implementation has no implications for other objects of interest. This arises because object-oriented languages enable a programmer to establish an abstract interface to a data structure which effectively hides the implementation details from the procedure using that data. All access to a data structure is effected via operations provided by the data structure's public interface. In object-oriented languages, the data contained within an object are private to that object and are accessible only via messages to the owning object

(Micallef, 1988). Hence, the message interface can make useful information about an object available while hiding its implementation details (Goldberg and Robson, 1983).

Rosson and Alpert claim that this kind of encapsulation will reduce working memory load since the programmer need not worry about the interactions between the object they are constructing and other objects. This claim has not been subject to empirical evaluation but the analysis presented in this paper would suggest that such a claim may well be valid. One should note, however, that at other levels, the use of object-oriented languages may place an extra burden upon working memory. In particular, in most object-oriented systems one is forced to specify the relationships among objects (or more accurately, among classes of objects) before operations upon those objects can be defined. Détienne (1990) has shown that this requirement causes considerable difficulties since changing the structure of an evolving program can be very difficult. To avoid this problem, one might expect programmers to rely upon working memory in order to establish relationships between objects and classes before committing this structure to an external representation.

One issue that is important in the context of the display-based analysis proposed in this paper is how one might begin to devise a scheme for externalising working memory during program generation. In the realm of object-oriented languages it has been proposed that one way of facilitating this is to provide a description level similar to that found in bibliographic databases (Green, Gilmore, Blumenthal, Davies and Winder, 1992). Here, the problem is one of retrieving a target from a partial description. Typically, bibliographic databases not only represent attributes of a text itself but also additional key words which can be used for searching and browsing. In programming terms, descriptors might be based upon persistent (eg. functional subsystems) or transient (eg. a set of items to be documented) relationships between code structures; chronological relationships (eg. code developed or tested at a particular time) etc.

Providing this additional representation of a program may facilitate the recomprehension process that is central to the analysis presented in the present paper. In particular, the provision of a description level could make certain relationships salient within a given task context, thus facilitating display-based recomprehension. Indeed, even such things as simple colour cues or tags (Lansdale, Simpson and Stroud, 1988) which identify commonalities between important code structures may facilitate the representation of salient relationships. Clearly there is significant scope for further research into mechanisms which can support both the externalisation of working memory and the recomprehension processes that appear to be central to display-based competence.

The language features described in this section are important in terms of the present analysis, since the incremental nature of code generation and comprehension/ recomprehension will clearly be affected by the nature of the language. The present analysis extends existing work by suggesting ways in which language features and strategy may interact in concert with features of the task environment to give rise to particular forms of behaviour. It should be noted that these effects would not be taken into account by display-based views, since the salience of particular features of the display remains undifferentiated. Hence, one important extension to the display-based models of problem solving would be the incorporation of a mechanism which allows one to specify the salience of particular display-based features. Moreover, existing accounts of display-based problem solving give no consideration to the kinds of information manipulation that are possible in different display spaces.

The framework presented here also suggests that problem solving success in programming will in part be determined by the nature of the task environment. Of particular importance will be the extent to which an environment supports nonlinear generation strategies and the ease with which changes to existing structures can be effected. These considerations may shed light upon

the finding that the use of certain forms of programming environment can be frustrating for experienced programmers. For instance, Neal (1987 a and b) has conducted a number of studies exploring the efficacy of syntax-directed editors. Such editors provide syntactic templates for particular structures. Hence, in Pascal, if the programmer inserts a 'begin' statement, a corresponding 'end' statement will be generated automatically. Neal found that experienced programmers frequently expressed dissatisfaction with such editors. Neal (1987a) comments that expert programmers "felt that to enter a program they had to do much more, both conceptually and physically because of the methods allowed for inserting and changing text ..." (p 100). Neal's findings together with those reported in this paper suggest that environments intended to support the coding process should provide the flexibility to support both incremental development and change.

3.4 Summary

While this paper has indicated the importance of display-based performance in programming, it has also suggested two primary limitations of this general approach. Firstly, existing accounts of display-based problem solving ignore the apparent relationship between expertise and the development of strategies for utilising display-based information. Secondly, such accounts fail to consider the possibility that different forms of display-based information will be differentially salient in the context of a given task. Further developments of display-based accounts of problem solving will need to address these issues if they are to provide a coherent description of human performance in the context of complex tasks.

There are a number of possible extensions to this work which will hopefully provide additional support for the analysis presented here. One question relates to the extent to which the present findings might generalise to tasks other than program generation. One way in which these experiments might be extended is to conceive of a similar method of constraining comprehension strategies. Here, one might present subjects with a limited access window through which a program can be inspected, allowing subjects to view a few lines of code at a time. If the interpretation presented here is correct, then one might once again expect experts to be affected rather more than novices by this manipulation, since our claim is that experts are relying to a greater degree upon the information provided by the display.

Another implication of a display-based view is that by relying upon a display, the effects of task interruption should be minimised, since less information is being manipulated in working memory. Hence, an experimental scenario might developed where programmers of different skill level are interrupted with or without concurrent maintenance of the display. If experts have developed strategies which lead to greater reliance upon the display, as suggested here, then they should be disrupted to a greater extent than novices, and especially in the condition where the display is not maintained.

Finally, while the view we have presented here may have considerable potential for our understanding of complex problem solving, it is clear that there is significant scope for research which might provide evidence for the generalisability of this approach. Theoretical accounts of display-based problem solving are becoming popular. However, the cogency of this stance will be dependent upon further articulation of those psychological mechanisms which govern the relationship between 'knowledge in the head' and 'knowledge in the world'.

References

Agre, P and Chapman, D., (1987). Pengi: An implementation of a theory of activity. In Proceedings of the Sixth International Conference on Artificial Intelligence, 268 - 272. Menlo Park, Calif: American Association for Artificial Intelligence.

Allwood, C. M., (1984). Error detection processes in statistical problem solving. *Cognitive Science*, **8**, 413 - 427.

Ambros-Ingerson, J. A., (1987). Relationships between planning and execution. AISB quarterly newsletter, Number 57.

Anderson, J. R., (1983). *The architecture of cognition.* Harvard University Press, Harvard, MA.

Anderson, J. R., (1987). Skill acquisition: Compilation of weak-method problem solutions, *Psychological Review*, **94**, 192 - 210.

Anderson, J. R., (1989). The analogical origins of errors in problem solving. In D. Klahr and K. Kotovsky (Eds.), *Complex information processing: The impact of Herbert A. Simon.* LEA, Hillsdale, NJ.

Anderson, J. R. and Jeffries, R., (1985). Novice LISP errors: Undetected losses of information from working memory. *Human-Computer Interaction*, **1**, 107-131.

Chase, W. G. and Ericsson, K. A., (1982), Skill and working memory. In G. H. Bower (ed.), *The psychology of learning and motivation.* Academic Press, New York.

Davies, S. P., (1990). The nature and development of programming plans. *International Journal of Man-Machine Studies*, **32**, 461 - 481.

Davies, S. P., (1991). The role of notation and knowledge representation in the determination of programming strategy: A framework for integrating models of programming behaviour. *Cognitive Science*, **15**, 547 - 572

Détienne, F. (1990). Difficulties in designing with an object-oriented language: an empirical study. In D. Diaper, D. Gilmore G. Cockton and B. Shackel (Eds.), Human-Computer Interaction, Interact'90. North-Holland, Amsterdam.

Gilmore, D. J., (1986). Structural visibility and program comprehension. In M. D. Harrison and A. F. Monk (Eds.), People and Computers: Designing for Usability, Cambridge University Press, Cambridge.

Gilmore, D. J. and Green, T. R. G., (1988). Programming plans and programming expertise. *Quarterly Journal of Experimental Psychology*, **40A** (3), 423 - 442.

Goldberg, A. and Robson, D., (1983). *Smalltalk - 80: The language and its implementation*, Addison-Wesley, Reading, MA.

Gray, W. D. and Anderson, J. R., (1987). Change-episodes in coding: when and how do programmers change their code? In G. M. Olson, S. Sheppard and E. Soloway (Eds.), Empirical studies of programmers: second workshop, Ablex, Norwood, NJ.

Green, T. R. G., (1990). The cognitive dimension of viscosity: a sticky problem for HCI. In D. Diaper, D. Gilmore G. Cockton and B. Shackel (Eds.), Human-Computer Interaction, Interact'90. North-Holland, Amsterdam.

Green, T. R. G., Bellamy, R. K. E. and Parker, J. M., (1987). Parsing and gnisrap: a model of device use, Proc. INTERACT'87, H. J. Bullinger and B. Shackel (Eds.), Elsevier Science Publishers B. V., North-Holland.

Green, T. R. G., Gilmore, D. J., Blumenthal, B. B., Davies, S. P. and Winder, R. (1992). Towards a cognitive browser for OOPS. *International Journal of Human-Computer Interaction,* **4**, 1, 1-34.

Greeno, J. G., (1989). Situations, mental models and generative knowledge. In D. Klahr and K. Kotovsky, (Eds.), *Complex Information Processing; The impact of Herbert A. Simon.* Erlbuam, Hillsdale, NJ.

Howes, A. and Payne, S. J., (1990). Display-based competence: towards user models for menu-driven interfaces. *International Journal of Man-Machine Studies*, **33**, 637 - 655.

Lansdale, M. W., Simpson, M. and Stroud, T. R. M., (1988). A comparison of words and icons as external memory aids in an information retrieval task. *Behaviour and Information Technology*, **9**, 2, 111 - 131.

Larkin, J. H., (1989). Display-based problem solving. In D. Klahr and K. Kotovsky, (Eds.), *Complex Information Processing; The impact of Herbert A. Simon.* Erlbuam, Hillsdale, NJ.

Micallef, J., (1988). Encapsulation, reusability and extensibility in object-oriented programming languages. *Journal of Object-Oriented Programming*, **1**, 1, 12 - 38.

Neal, L. R., (1987a). Cognition-sensitive design and user modeling for syntax-directed editors. In J. M. Carroll and P. Tanner (Eds.), Human Factors in Computing Systems IV, North-Holland, Amsterdam.

Neal, L. R., (1987b). User modeling for syntax-directed editors. In H.-J. Bullinger and B. Shackel (Eds.), Proc. of INTERACT'87, North-Holland, Amsterdam.

Rasmussen, J., (1983). Skills, rules knowledge: signs and symbols and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, 13, 257 - 267.

Reason, J. T., (1979). Actions as not planned: The price of automatization. In G. Underwood and R. Stevens (Eds.), *Aspects of Consciousness, Volume 1: Psychological issues*, Wiley, London.

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

Rist, R. S., (1989). Schema creation in programming. *Cognitive Science*, **13**, 389 - 414.

Rosson , M. B. and Alpert, S. R., (1990). The cognitive consequences of object-oriented design. *Human-Computer Interaction*, **5**, 345 - 379.

Soloway, E. and Ehrlich, K., (1984). Empirical studies of programming knowledge. *IEEE Trans. SE*, **SE - 10**(5), 595 -609.

Staszewski, J. J., (1988). Skilled memory and expert mental calculation. In Chi, M. T. H., Glaser, R. and Farr, M. J., (Eds.), *The nature of expertise*, Lawrence Erlbaum, Hillsdale, NJ.

EFFECT OF FATIGUE IN MAN-MACHINE INTERACTION

L. Linde Nat. Defense Res. Establishment, Sweden

ABSTRACT

Tasks involving interaction between a human and a computer system range from strictly routine to complex tasks, involving problem-solving activity. The relationship between fatigue - due to stressors such as a long, sustained work period, inconvenient work hour and lack of sleep - and performance in complex, interactive computer-aided tasks is discussed. It is argued that level of performance in such tasks may decline substantially due to fatigue. To predict a performance decline one has to consider both the type of stressor and the nature of the task. A framework for studying relationships between stressors and performance under different forms of informational load is suggested. Principle methods for making man-machine tasks less vulnerable to fatigue-related action failures are discussed.

1. Introduction

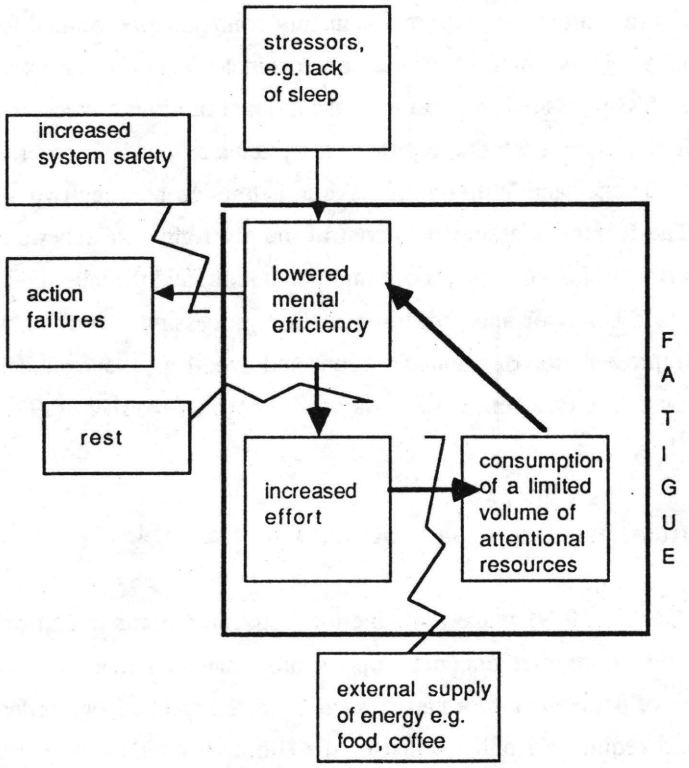


Fig. 1 Suggested relationship between stress, fatigue and action failures

Figure 1 describes a hypothetical relationship between stressors, such as the ones mentioned above and action failures. It is assumed that a person, with orders to perform a certain task, tends to increase effort to maintain a constant level of performance when exposed to stress. Ultimately increased effort might decrease the volume of attentional resources and give rize to subjective and objective (i.e. impaired performance) fatigue. The bold arrows in the figure illustrates a possible vicious circle. The figure also suggests three principles ways of breaking such a vicious circle.

2. Energetical systems and changes in level of performance

The concept of "arousal" has been used to denote a general energetical variable, ranging from sleep to extreme alertness. Many authors, however, have presented arguments against a unidimensional view of arousal (e.g. Broadbent, 1971; Kahneman, 1973; Pribram and McGuiness, 1975; Hamilton, Hockey and Rejman, 1977; Thayer, 1978; Eysenck, 1982; Sanders, 1983; Tucker and Williamson, 1984). Broadbent argued for a lower and higher order arousal system and suggested that a decrease in lower order arousal might be counteracted by an increase in higher order arousal. Lacey (1967) and Kahneman distinguish two attentional states differing in their autonomic manifestations; a state of acceptance of sensory information and inhibition of response and a state of active manipulation of information. Pribram and McGuiness and Sanders propose three system; arousal assumedly related to perception (encoding of information), activation assumedly controlling perceptual focusing and motoric behavior and effort assumed to have a superordinate function in relation to the other two. Pribram and McGuiness suggest that increase in level of arousal precedes increase in level of activation in classification tasks, but that the systems responds in reverse order in problem-solving tasks. Tucker and Williamson make a distinction between two systems; arousal and activation. The former is assumed to control mainly right hemispheric activity, global and parallell, distributed information processing (perception) and the latter mainly left hemispheric activity, motoric behavior and serial information processing ("thinking"). Figure 2 gives a hierarchical representation of mental functions and hypothetical energetical systems regulating these functions. The energetical systems are based on Broadbent (1971) and Pribram and McGuiness (1975).

3. Information processing and mental effort

Hasher and Zacks (1979) make a distinction between automatic and effortful information processing and claim that encoding operations vary in attentional requirements. Some characteristics of perceptual information, such as spatial and temporal order, are assumed to be automatic and require a small amount of the (limited) attentional resources. Other mental operations- effortful- are assumed to require a large amount of the attentional resources. Effortful processes, according to H & Z, are e.g. encoding of semantic characteristics in information (logical content) and rehearsal in STM. H & Z also assumed that effortful

processes can develop into automatic by learning. Finally, they assumed that the amount of attentional resources available for "effortful" information processing can be reduced by several factors in the individual (e.g. age, depression, stress).

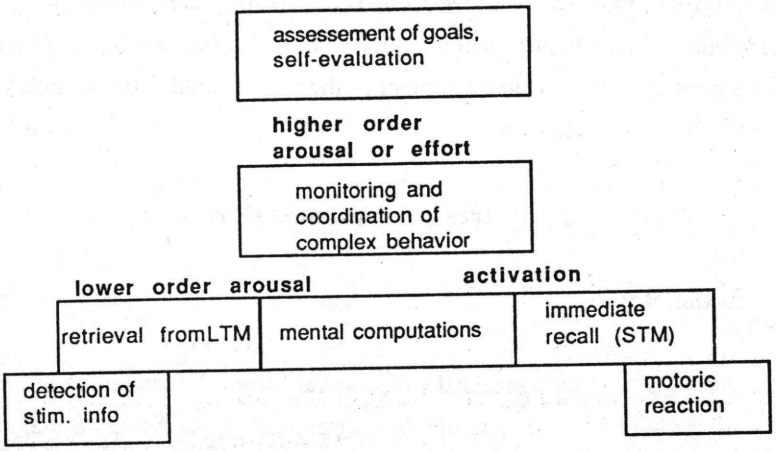


FIG. 2: A hierarchical representation of mental functions and a hypothetical energetical control system (cf. Broadbent, 1971; Pribram and McGuiness, 1975)

Table 1: Different types of informational load.

The speed, i.e. temporal density, at which information appears : Information may be presented at a speed, which exceeds the speed at which the operator is able to process it.
Example: listening to a foreign language, spoken at a high speed or air-fighting.

Length of the period, during which sustained attention is required (vigilance):
There is mostly a performance decline efter 30 to 60 min. probably due to perceptual habituation.
Example: Watching a radar screen in order to detect certain objects.

The spatial density of information presented simultaneously (selective attention and pattern recognition):
Example: Watching a digital image , e.g. from a sattelite, in order to search for certain patterns.

The required depth of information processing (automatic vs. effortful information processing):
Example: Reading textual information in a manual or helpfile or constructing a complex query in a formal query language.

Working memory load:
Example: The number of alternatives and relevant attributes that have to be kept in memory during a decision-making task.

4. Informational load

One way of characterizing man-machine pertains to characteristics of the informational influx and to how the information must be processed by an operator (cf Moray, 1982). Table 1 suggests five types of informational overload. 1. Information is presented at a too high speed. 2. The period during which sustained attention is required is too long. 3. The information is too detailed and rich. 4. The semantic content of the information is too complex. 5. Too much information has be kept in memory.

Cognitive performance and stress; experimental results

5.1 Time of the day

Body temperature (assumedly correlated with general "arousal") tend to be lowest between 2 and 6 in the morning and rize during the day in persons adjusted to a normal sleep/activity schedule, i.e. sleeping at night and working in the day (Colquhoun, 1971). However, to a certain extent an adjustment to a reversed sleep/activity schedule occurs after repeated night work (during 10 to 14 days) (Colquhoun, op.cit). There seems to be a parallell variation of level of performance in vigilance tasks, such as signal detection (e.g. Blake, 1967) and in speed of retrieval from semantic memory (Tilley and Warren, 1984; Tilley, Horne and Allison, 1985). STM, on the other tends to correlate negatively with time of day and diurnal body temperature (Blake, op.cit; Fröberg, 1979; Folkard, Knauth, Monk and Rutenfranz, 1976). Folkard (1975) found that speed of performance on the logical reasoning task devised by Baddeley (1968) increased from 8 a.m. to 2 p.m., but that accuracy declined during the same period.

5.2 Lack of sleep

A number of studies indicate a decline in vigilance due to loss of sleep for one night, or more (see e.g. Johnson, 1969). Performance in several cognitive tasks has also been found to be affected by sleep loss for one night or more, for example; decreased selectivity of attention (Hockey, 1973) lowered speed of retrieval of semantic knowledge (Tilley, Horne and Allison, 1985), increased number of errors in inductive inferences (Linde and Bergström, 1992).

5.3 Sustained period of work (task fatigue)

A classic study of task fatigue - "Cambridge cockpit studies" - among others thing demonstrated a narrowing of attention (i.e the stimulus field tended to "split up in parts") after a certain period of uninterrupted time on the task. For example, Angus and Heslegrave (1985) have studied level of performance in cognitive tasks during very long sustained work. After 24 h of continuous work and lack of sleep level of performance was 70 % of the baseline level

of performance. Table 2 summarizes some experimental results pertaining to effects of fatigue in the performance of mental tasks.

Table 2: Some observations on effects of fatigue in different types of mental tasks.

Type of stressor and observed effects

Lack of sleep
*Decreased vigilance (signal detection, and reaction time)
*Slower retrieval from LTM
*Decrease in speed of performance in syntactical reasoning
*Loss of selectivity of visual attention
*Impaired inductive reasoning (progressive matrices task)

Time of day
*Decreased vigilance during the night
*Slower retrieval from LTM during the night
*High STM capacity during the night
*High accuracy/low speed on a syntactical reasoning task in the morning
*Low accuracy/high speed on a syntactical reasoning task in the afternoon

Task fatigue
*Sustained period of work on the same task narrows the attentional field (it tends to "split up in parts")

6. Characterization of man-machine interaction tasks

Figure 3 suggests four aspects, in terms of which man-machine interaction tasks may be described. One aspect is related to the user´s level of skill on the task. Another aspect is the form of cognitive activity involved in the performance of the task. A third aspect pertains to the type of efficiency being required in the task. A fourth aspect has to do with the locus of control of the interaction (e.g. Moray, 1982). For example, the locus of control in typing, and code construction is operator-paced but in a supervision task the locus of control can be machine-paced. Tasks located on the right end of the aspects are assumed to require more effort than those located on the left end. Level of skill and type of information processing are two factors, which according to Hasher and Zacks (1979) affect amount of attentional resources

required in a task. The possibility to make voluntary short pauses while performing a task can also be assumed to affect amount of attentional resources required. There may be less opportunity for brief pauses in a machine-paced task than in an operator-paced and the opportunity to make pauses decreases with increased time-pressure. In Figure 4 the aspect "nature of cognitive activity" has been given a two-dimensional representation. The dimensions constitute hypothetical attentional states (cf. Lacey, 1967; Kahneman, 1973). Attentive acceptance of information is associated with e.g. decreased heart rate and pupil dilatation, whereas attentive manipulation of information is associated with an increased heart rate (Lacey, op.cit). For example, construction of command or program code (e.g. in formal query construction) is assumed to involve primarily serial manipulation, that is active manipulation, of symbolic information. Attentive acceptance of information is not of equal importance. On the other hand, during error search both attentive acceptance (AA) and active manipulation (AM) may be required; AA for perceptual search and AM for hypthesis testing.

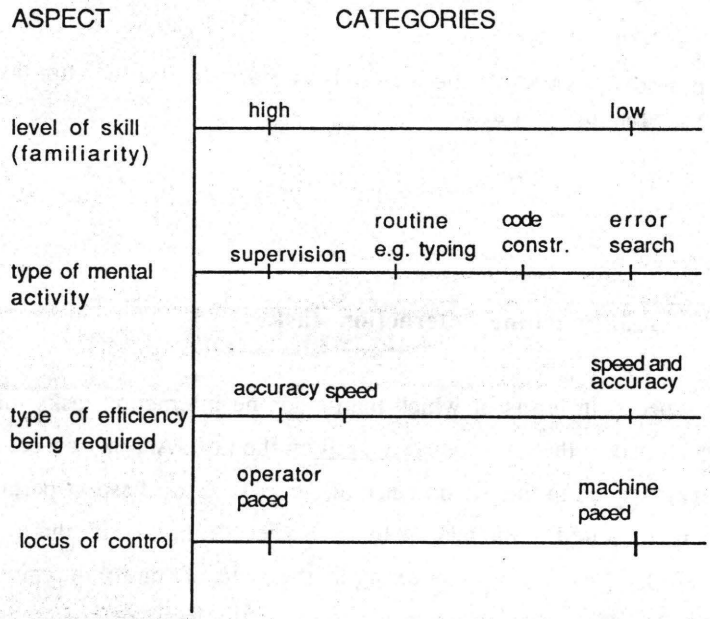


FIG. 3 Aspects and examples of categories of HCI tasks. Categories on the right of the aspects are believed to be more effortful than those on the left.

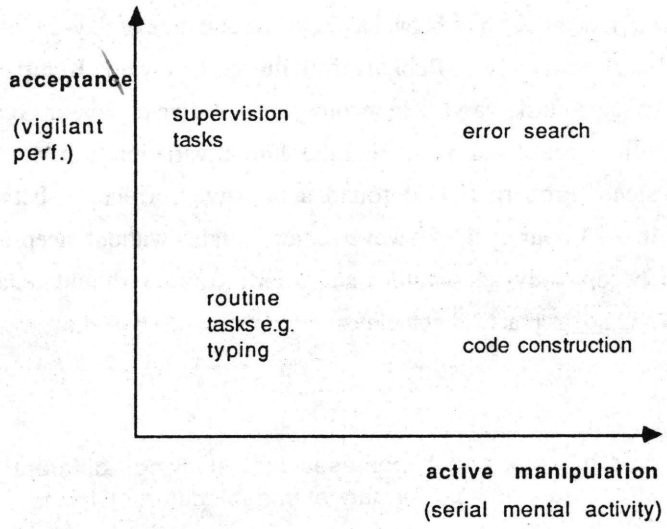


FIG. 4 Type of mental activities (attentional states) assumed to be involved in different HCI tasks.

7. Fatigue and action failures during man-machine interaction: suggested framework

Figure 3 suggested a general way of categorizing man-machine tasks. Figure 4 described one of the aspects (form of cognitive activity) in more detail; in terms of two hypothetically different attentional states. A specific type of task (e.g. supervision or error search) may also be characterized in terms of informational load, which the operator is subjected to. Table 1, on the one hand, and Figure 3 and 4, on the other, represent to perspectives on man-machine tasks. One perspective is oriented at the informational processing demands facing the operator and the other is oriented at the task (its object, how familiar it is to the operator) and at the characteristics of the interaction between the man and machine.

Table 3 gives a framework for studying the effect of different types of stressors under different forms of informational overload. Three types of stressors are considered; insufficient sleep, night work (by persons not adjusted to it) and extended period of work. For example, it is suggested that the combination of night work and high speed of informational influx might

increase the likelihood of action failures, but not the combination of night work and high semantic/logical complexity of information. Question-marks indicate that no experimental work pertaining to the factors in question has been found. Two comments should be about Table 3. 1. The diurnal variation in body temperature and test performance (e.g. in vigilance tasks) changes after a certain period of a reversed sleep/activity schedule (e.g. Colquhoun, 1971). The adjustment seems to take between 10 to 14 days. 2. There may be interactions between different external factors -such as between sleep loss and time of day- in the respect of their effects in level of performance. E.g. Babkoff, Mikulincer, Caspy and Kempinski (1988) studied performance in a search task, varying in memory load, during a 72 hour vigil. Level of performance was lowest between 2 and 6 a.m. and the diurnal variation in performance was amplified by loss of sleep. Fröberg (1979) found a negative correlation between body temperature and STM in a 24 hour cycle. However, after 2 nights without sleep there was a positive correlation between body temperature and STM. Table 3 should regarded as a framework for research and not as practical guidelines.

Table 3: A framework and hypotheses for studying different types of stressors and performance under different forms of informational load.

	lack of sleep	night work	long period of work
overload due to high speed	high vulnerability	high vulnerability	?
overload due to extended time on task	high vulnerability	high vulnerability	high vulnerability
overload due to high spatial density	high vulnerability	?	high vulnerability
overload due to sem. and logic comp-lexity of informa-tion	high vulnerability	low vulnerability	?
high load on working memory	high vulnerability	low vulnerability	?

8. Can action failures be avoided?

Table 4: Some general guidelines for choosing methods to avoid action failures in man-machine systems.

High temporal density
E.g. training of operators on the task and selection of high performing individuals

Long observation period
E.g change of work schedules

High spatial density (richness)
system design e.g.,
*structuring of information
*filtering of information
*enhancement of relevant objects

Effortful information processing
system design e.g.,
*allow the operator to search for a complex, logical set of objects in a step-by-step fashion and check the result of each step

High working memory load
system design e.g.,
*decision aids

Measures against fatigue-related action failures in a man-machine system can be taken at an organizational level, i.e. changes of work schedules, or at an individual level. E.g. operators may be given training on emergency situations or education in methods to counteract fatigue. One may also change system design. Depending on what type of informational load is most critical some type of measures may be more appropriate than others. Table 4 suggests appropriate measures for different types of informational load. For example, when there is a very high speed of informational influx, the most important way to avoid action failures may be selection of high-performing individuals and to give operators (e.g. air fighters) extensive training. On the other hand, when there is an extremely rich and detailed display of visual information (e.g. in a control panel) changes in system design may be applicable. For example, information search may facilitated by a change in the layout of objects, processes and states. The operator can also be allowed to demand filtered information (e.g. show textfiles but no other information) or enhancement of certain information (e.g. by colouring). For example in formal query writing information can have a high semantic-logical complexity. In some situations it may be easier to construct a complex query in a step-by-step way, even though such a method might be more time-consuming than writing one complex expression. That is,

in a step-by-step way of query construction the operator can check results of one logical operator at a time and thereby unload working memory. Sometimes the situation requires an excessive amount of information to be held in memory by the operator. For example, the number of alternatives and relevant attributes of the alternatives in a choice situation may be large. In such situations decision aids may be useful.

In conclusion, when designing man-machine systems the following should be considered.

1.What type of stressor(s) are critical?
2.What are the most typical effects of those stressors?
3.What type of informational load is most critical?
4.Describe the task in terms of object, type of cognitive activity etc.
5. Is a change in system design the best way to counteract action failures?

References

Angus R.G. and Heslegrave R. (1985) Effects of sleep loss on sustained cognitive performance during a command and control simulation, Behavior Research Methods, Instruments & Computers 17(1), 55-67.

Babkoff H., Mikulincer M., Caspy T., Kempinski D., and Sing H. (1988) The topology of performance curves during 72 hours of sleep loss: a memory and search task, The Quarterly Journal of Experimental Psychology, 40A(4), 737-756.

Baddeley A.D. (1968) A 3 min reasoning test based on grammatical transformations, Psychonomic Science, 10(10), 341-342.

Bartlett F.C.(1943) Fatigue following highly skilled work, Proc. of the Royal Soc., series B,131, 247-257.

Blake M.J.F.(1967) Time of day effects on performance in a range of tasks, Psychonomic Science, 9(6), 349-350.

Broadbent D.E.(1971) Decision and Stress, Academic Press, London.

Colquhoun W.P. (1971) Circadian variations in mental efficiency, in W.P. Coquhoun (ed.) Biological Rhythms and Human Performance, Academic Press, London.

Eysenck M.W.(1982) Attention and Arousal, Springer Verlag, New York.

Folkard S. (1975) Diurnal variation in logical reasoning, British Journal of Psychology, 66(1), 1-8.

Folkard S., Knauth P. , Monk T.H. and Rutenfranz J.(1976) The effect of memory load on the circadian variation in performance efficiency under a rapidly rotating shift system, Ergonomics, vol. 19(4), 479-488.

Fröberg J.(1979) Performance in tasks differing in memory load and its relationship with habitual activity phase and body temperature, FOA rapport C52002-H6.

Hamilton P., Hockey G.R.J. and Rejman M.(1977) The place of the concept of activation in human information processing theory: an integrative approach, in S. Dornic (ed.) Attention and Performance VI, LEA Hillsdale, New Jersey.

Hasher L. and Zacks R.T.(1979) Automatic and effortful processes in memory, Journal of Experimental Psychology: General, 108 (3), 356-388.

Holding D.H.(1983) Fatigue, in G.R.J. Hockey (ed.) Stress and Fatigue in Human Performance, John Wiley & sons LTD.

Johnson L.C. (1969) Psychological and physiological changes following total sleep deprivation, in A. Kales (ed.) Sleep: physiology and pathology, Philadelphia, Lipincott.

Kahneman D.(1973) Attention and Effort, Prentice Hall.

Lacey J.(1967) Somatic response patterning and stress: some revisions of activation theory, in M.H. Appley and R. Trumbell (eds.) Psychological Stress, New York, Appleton-Century Crofts.

Linde L. and Bergström M. (1992) The effect of one night without sleep on problem-solving and immediate recall, Psychological Research, 54, 127-136.

Moray N. (1982) Subjective mental worklad, Human Factors, 24 (1), 25-40.

Pribram K.H.and McGuiness D. (1975) Arousal, activation and effort in the control of attention, Psychological Review, 82(2), 116-149.

Revelle W., Humphreys M.S., Simon L. and Gilliland K. (1980) The interactive effect of personality, time of day, and caffeine, a test of the arousal model, Journal of the Experimental Psychology: General, 109(1), 1-31.

Sanders A.F.(1983) Towards a model of stress and human performance, Acta Psychologica, 53, 61-97.

Thayer R.E.(1978) Toward a Psychological Theory of multidimensional activation (arousal), motivation and emotion, 2(1), 1-33.

Tilley A. and Warren P. (1984) Retrieval from semantic memory during a night without sleep, Quarterly Journal of Experimental Psychology, 36, 281-289.

Tilley A., Horne J. and Allison S.(1985) Effects of sleep loss on retrieval from semantic memory at two different times of day, Australian Journal of Psychology, 37(3), 281-287.

Tucker D.M. and Williamson P.A. (1984) Asymmetric neural control systems in human self-regulation, Psyhological Review, 91(2), 185-215.

part 2
COOPERATION AND COMMUNICATION: ANALYSIS AND EVALUATION

A Dynamic Tasks Allocation in Air Traffic Control : Effects on Controllers' Behaviour

F. Vanderhaegen, I. Crevits, S. Debernard, P. Millot

Laboratoire d'Automatique Industrielle et Humaine, URA CNRS 1118
Université de Valenciennes et du Hainaut-Cambrésis
B.P. 311 - Le Mont Houy - 59304 VALENCIENNES Cedex - FRANCE

ABSTRACT

A lot of investigations about Air Traffic Control are appearing in order to try to create optimal assistance tools for controllers whose workload becomes more and more heavy. In such a way, the purpose of our research is to propose and validate a new organization of the air traffic control, which allows air traffic controllers to stay active in the control and supervisory loop of the process, in order to maintain the current traffic safety level and to improve the global system performances. Our research is directed towards an horizontal cooperation that consists in a dynamic allocation of control tasks between human air traffic controllers and an assistance tool. In such a way, an experimental platform, called SPECTRA (french acronym for : Experimental System to Allocate the Air Traffic Control Tasks) has been built.

In a first step, this research has aimed at validating SPECTRA using qualified controllers who are air traffic control experts. According to the first conclusions SPECTRA was a really good assistance tool for loaded traffic context. In a second step, SPECTRA has been tested by another class of controllers who are less trained : air traffic control engineers who had followed a dedicated training during three months at least. This paper recalls the SPECTRA description with the dynamic tasks allocation principles integration and the experimental protocol and for each class of controllers, it details all the results about experiments and shows the effects of such an assistance on controllers' behaviours.

1. Introduction.

In the Air traffic control context, a lot of investigations are made in order to realize efficient assistance tools for the controllers who have an irregular and heavy workload. But the best software adaptation would be individual integrating all the particular features of the human operator and the tasks (Gillet, 1990). Indeed, some individual variables lead to different attitudes and performance levels in man-machine interaction situations.The introduction of automation in a complex domain like air traffic control, can lead to some difficulties, particularly about trust (Hopkin, 1991).

According to these remarks, the purpose of our research is to propose and validate a new organisation of the air traffic control which allows air traffic controllers to stay active in the supervisory loop of the process, in order to maintain the current safety level and to improve the global system performances (Debbache and colleagues, 1987). Because of the complexity of this air traffic control context, our approach first focused on the tactical level managed by the "radar controller". He supervises the traffic (respect of separations between planes) and dialogues with the aircraft pilots (information about traffic and modification of flight parameters). The new organization we propose is based on an horizontal cooperation (Millot and Kamoun, 1988; Kamoun and colleagues, 1988) which consists in a dynamic allocation of control tasks between the air traffic controller and an expert system for air traffic control regulation called SAINTEX. It can resolve conflicts between two planes in rather simple conditions (Morchelles and colleagues, 1989). For testing the dynamic task allocation in air traffic control context, a simplified but realistic experimental platform, called SPECTRA has been built (Debernard and colleagues, 1990).

In order to validate SPECTRA and to study the effects of such an aid on controllers' behaviour, experiments were realized with air traffic controllers who had different ability levels. In fact, these

experiments have been performed by two classes of operators who had opposite competence level. The first class was composed of nine qualified air traffic controllers who are air traffic control experts with the highest competence. The second one was composed of six air traffic control engineers who hadfollowed a dedicated training in real control room during three months at least, but who were not high specialist at all. In a first way, experiments concerning only the nine qualified controllers were already analysed (Vanderhaegen and colleagues, 1991; Debernard and colleagues, 1992). They have shown that, for highly qualified controllers, a dynamic tasks allocation gives a real help for loaded air traffic control. The aim of the study described in this present paper is to show that SPECTRA remains a good assistance tool for controllers whatever their competence or qualification level.

This paper first recalls the experimental protocol with the SPECTRA description including the dynamic tasks allocation principles and the workload and performance assessments, the experimental context and the experimental data. The three other parts concern the results and focus on subjective and general controllers' remarks, on controllers' workload and on controllers' performance for qualified controllers as well as ATC engineers.

These studies are realised in cooperation with the C.E.N.A (French Research Centre of Air-Traffic Control).

2. Experimental protocol.

2.1. Dynamic tasks allocation principles on SPECTRA.

The dynamic task allocation principle (Millot, 1988) consists in inserting a task allocator in the supervisory loop which shares control tasks between the air traffic controller and the expert system SAINTEX. This aims at both regulating the human workload, and improving the performance and the reliability of the air traffic controller-SAINTEX team. According to this principle, two types of dynamic allocation managements exist and are integrated in SPECTRA structure, figure 1.

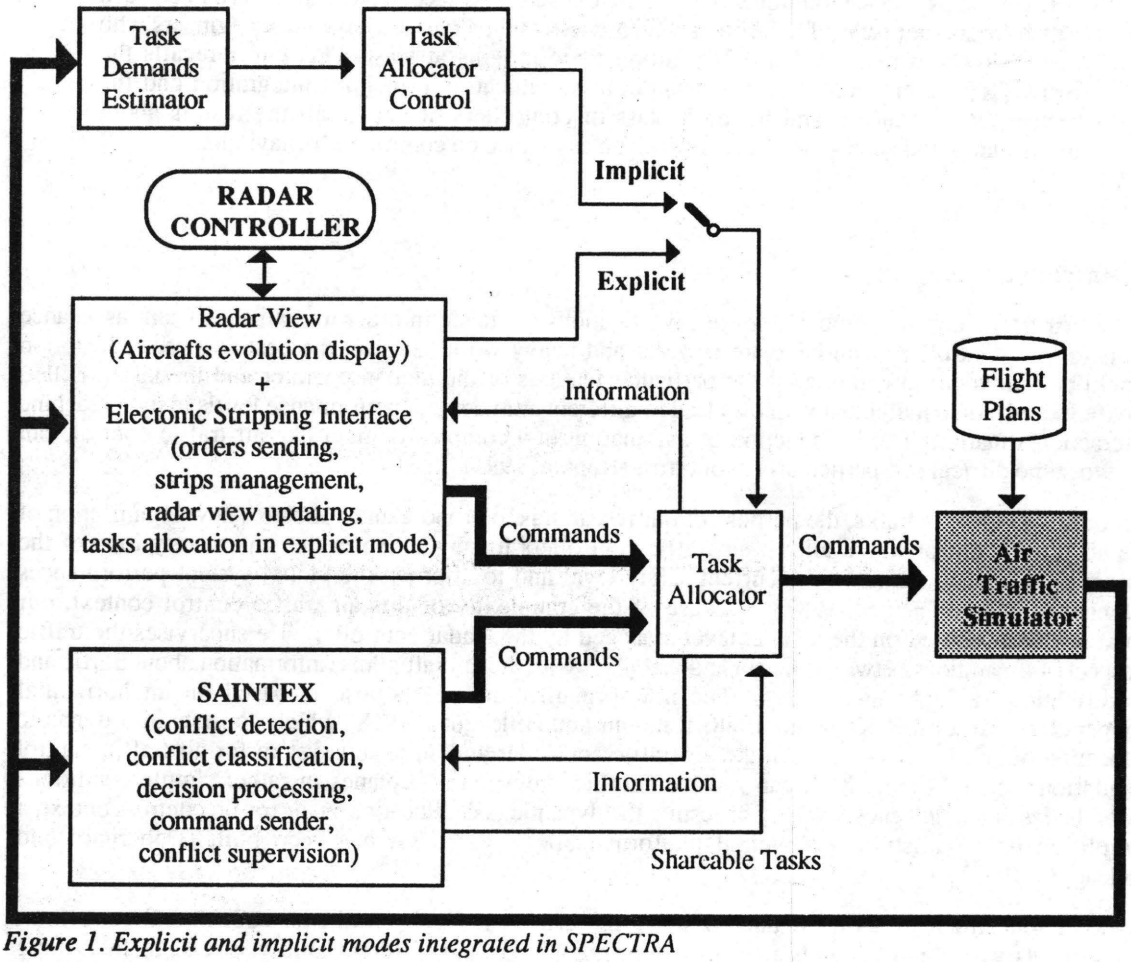


Figure 1. Explicit and implicit modes integrated in SPECTRA

The former concerns an "explicit" task allocation performed by the radar controller himself through a dialogue interface. The human air traffic controller manages the task allocator. He is his own estimator of performance and workload, and he allocates tasks either to himself or to SAINTEX. The conflicts SAINTEX can solve are indicated with a specific color on the operator dialogue screen and on the radar view. The deadline before which the conflict becomes unsolvable by SAINTEX and the deviation order calculated are displayed on the strips of the concerned planes. Initially, all planes are allocated to the controller. If he feels himself overloaded, he can select a conflict and transfer it to SAINTEX. As he is the system manager, he can take a conflict back. But at that time, if the order deadline given by SAINTEX is over, this conflict can not be solved by SAINTEX anymore .

The latter is an "implicit" task allocation controlled by an automatic management system implemented on the calculator. This allocation depends on the intrinsic capabilities of the two decision-makers. For SAINTEX, those abilities are functional ones : it can solve the conflits between two planes which are not evolutive in flight level; these conflicts are the shareable tasks. For the human radar controller, these abilities are linked with his workload. Currently, only the tasks demands are assessed. So, when these demands are too high and exceed a maximum level, the shareable tasks are allocated to SAINTEX. In that case, the conflict with the nearest order deadline is chosen and transfered to SAINTEX. In the contrary case, the conflict is affected to the human controller. The maximum tasks demands level were determined thanks to the cooperation of qualified air traffic controllers.

2.2. Workload and performance assessment.

It is difficult to assess the controllers' workload on line because of the complexity and the specificity of the air traffic control activity (Hopkin, 1991; Jorna, 1991). Even though formulas about objective measures of job difficulty already exist (Hurst and Rose, 1978), they are too much complicated and not really validated. Therefore, instead of introducing a workload estimator on SPECTRA, we have chosen a task demand estimator. This task demand estimator adds all the tasks demands the air traffic controllers are submitted to. These demands are only functional and take into account the difficulties encountered when assuming a plane, detecting a conflict, correcting and supervising a conflict and replacing a plane on its initial trajectory. So, it consists in affecting a weight to the demand of each event appearing during the experiments. In order to validate this assessment, a module was added on SPECTRA. It concerns the on line subjective evaluation of controller's workload. The air traffic controller estimates his own workload, clicking with a mouse on a small scale, graduated from 1 (lower load) to 10 (higher load). This board appears every five minutes without disturbing the air traffic control because the operator answers when he wants to do.

Another difficulty was the global performance assessment on line i.e. the performance obtained by the decision-makers team. However, in order to compare the experiments, an a posteriori global performance criterion was defined. This performance is composed of two factors that cannot be aggregated. The former is the traffic security, and the latter is the traffic flow optimization. Therefore, two criteria have been defined. The insecurity criterion is the ratio between the number of alarms and the number of conflicts appeared during the scenario. An alarm is activated when planes transgress separation norms. The traffic flow optimization criterion for the controller-SAINTEX team is based on an economic criterion which is the ratio between the real consumption of kerosene of each aircraft and the theoretical one. The theoretical consumption corresponds to the direct way between the input and output beacons. The real consumption corresponds to the way really taken according to changes in cape and level. The best global performance is obtained when the real consumption is equal to the theoritical one and when there is no alarm at all. This global performance is calculated a posteriori in order to compare the experiments.

2.3. Experimental context.

The scenarios used for these experiments have been created in order to overload the air traffic controller, in such a way that a dynamic allocation be useful. Therefore, they involve a great number of planes (between 40 and 50 per hour) that are generating a lot of conflicts (more than 20) of different natures, into a large geographical sector (Vanderhaegen and colleagues, 1991). Because of the capabilities of SAINTEX, the air traffic control context concerns the night control. The modelling of this situation is easier because the planes cross the sector in straight lines between two beacons : a sector input point and a sector output one.

One training scenario permits to controllers to get used to SPECTRA interface in explicit mode, during

two hours. Other scenarios have been created for three kinds of experiments. In the first type, the controller is not aided. In the second kind, he is supported by SAINTEX in an explicit dynamic allocation mode. In the last one, he is assisted in an implicit allocation mode.

These experiments have been performed by nine qualified air traffic controllers and six ATC engineers. The qualified controllers are air traffic control experts with the highest competence. Although the engineers are not high specialist at all because they have followed a dedicated training in real control context during three months at least.

2.4. Experimental data

Three kinds of parameters have been measured and recorded during each experiment :

- The first class concerns all the data needed to calculate a posteriori the global performance criteria.
- The second class is the task demands measured on line.
- The last one concerns the on line subjective evaluation of controller's workload.

Moreover, in the final step of each experiment, two questionnaires have been submitted to the controller. The first questionnaire is the Task Load indeX method (NASA-TLX). TLX is a subjective method developed by NASA that permits to calculate the global workload. This global workload estimator is based on six semantic descriptors : mental demand, physical demand, temporal demand, performance, effort and frustration. The second questionnaire is a series of questions that allowed us an oriented discussion with the controller after the experiment. These questions concern :

- General information about the scenario (for example, about radar interface and strip interface, about strategies used...).
- Specific information about the workload, the estimated performance and the alarms.
- Specific information about the explicit mode (questions asked after the experiment in explicit mode), implicit mode (questions asked after the experiment in implicit mode) and their comparison (questions asked after all the experiments).

In order to simplify the results analysis, in the following parts, the Air Traffic Control Engineers Experiments are quoted with the index "EE" and the Qualified Controllers Experiments are quoted with "QC".

3. Discussion using general results and Subjective evaluation.

3.1. Global results about SPECTRA.

For 56% of EE and 63% of QC, the number of planes is not too high, but for 22% of EE against 74% of QC, the air traffic is not structured. For 78% and 89% of EE the radar and strips interfaces are correct. On the other hand, the qualified controllers satisfaction is not so good (respectively 33% and 59% of QC). Qualified controllers estimate that the radar view should be enhanced because :

- the sector they have to supervise is too large,
- the traffic is not structured enough,
- and they cannot have sufficient marks.

The stripping interface is correct although the high number of planes involves heavy strips management.

Moreover, SPECTRA does not change usual controllers strategies because, like in a real control room, the stripping interface is used for conflicts detection (50% of EE and 74% of QC) and the radar view for conflicts solving (67% of EE and 96% of QC) and for conflicts supervision (67% of EE and 100% of QC). As in a loaded daily control, controllers immediately solve the conflicts for preventing an overloaded situation (44% of EE and 85% of QC).

3.2. Results about explicit mode.

After experiments in explicit mode, controllers estimate unanimously that this kind of allocation give a real assistance. They think it reduces workload (83% of EE and 100% of QC). Explicit mode did not cause any problem for the air traffic control (67% of EE and 89% of QC).

Nevertheless, even though this mode is a good assistance tool, an homogeneous policy about the use of SAINTEX by the controllers can hardly be found :

- Two qualified controllers and one engineer allocate conflicts to SAINTEX when they are overloaded and when they have too much conflicts.
- Two qualified controllers and one engineer use only their workload as allocation criterion.
- One qualified controller and one engineer use the number of conflicts as allocation criterion.
- One qualified controller and one engineer allocates conflicts to SAINTEX if the delay to send orders to planes is short. In the opposite case, they improve the SAINTEX's solution.
- One qualified controller allocates automatically one conflict upon two.
- One qualified controller uses the system as an horizontal cooperation tool : he uses SAINTEX as a conflict detector, and then refines its solutions.
- One qualified controller gives conflicts to SAINTEX if its solution is satisfactying.
- One engineer allocates a conflict to SAINTEX according to the conflict complexity.
- One engineer allocates automatically all solvable conflicts to SAINTEX.

Qualified controller take care of SAINTEX conflicts as much as their own ones (22% of QC) or if they have time enough (33% of QC). On the other hand, engineers do not really (50% of EE) or not at all (33% of EE) take care of SAINTEX conflicts. Moreover, they never take a conflict previously allocated to SAINTEX contrarlly to qualified controllers. Without assistance, the engineers were really overloaded. With the explicit mode integration, they have no time enough to supervise seriously the SAINTEX conflicts because they want to regulate at best their own workload and to control the air traffic very well. On the other hand, the qualified controllers check SAINTEX solutions and do not allocate the conflicts if they think that those solutions are not good. Valot (1988) named this behaviour as the active trust behaviour that is more obvious in the implicit mode.

3.3. Results about implicit mode.

Indeed, the qualified controllers supervise SAINTEX conflicts as much as their own (33% of QC) or if they have time (33% of QC) although they can not take them back. In fact, implicit mode imposes them a supervision workload because they are not overloaded at all, because they feel responsible for the whole air traffic control and because they are usually not satisfied by SAINTEX solutions. On the other hand, engineers do not supervise them at all (67% of EE). With this kind of allocation, they do not take care of them because they do not really have time to and because they do not want to take the risk of forgetting conflicts the system allocates to them.

Nevertheless, the implicit mode reduces controllers workload (83% of EE and 78% of QC) and provides a real assistance (83% of EE and 78% of QC). It does not provide any problem for the air traffic control (67% of EE and 89% of QC).

3.4. Comparison between explicit and implicit modes.

Qualified controllers prefer the explicit mode (44% of QC) whereas the engineers prefer the implicit mode (67% of EE). For the qualified ones, the supervision of SAINTEX conflicts is loader and more bothering in implicit mode than the shareable tasks allocation management and the supervision task in explicit mode. Nevertheless, controllers think that the implicit mode is the most efficient one (50% of EE and 78% of QC).

Engineers have no opinion about the choice of one mode to be integrated in a real control room because they are not used to controlling the air traffic. For qualified controllers, it is difficult to class answers. Some of them would prefer the implicit mode but they do not trust SAINTEX or any other future decisional stage. They ask for the possibility to take conflicts back, but they admit they have more time to solve their own conflicts. Other controllers would prefer explicit mode because they feel more responsible of the whole air traffic control. Nevertheless, they think that this mode is more dangerous because the human controller is the only responsible for the traffic, and the SPECTRA organization is not tolerant to human errors. For instance, when a controller forgets to allocate shareable tasks, a conflict may remain unsolved. But, we can also notice that this problem already exits in the current real air traffic control context.
Despite the different feelings about the use of the assistance tool in explicit or implicit mode, controllers think that the dynamic task allocation principle leads to decrease their own workload. We are going in the two following parts to try to confirm those subjective remarks and to explain with more details the

different controllers' behaviour.

4. Discussion on the results about workload.

4.1. Questionnaire about workload.

During the experiments, controllers had to evaluate their own workload on a graduated scale. They give higher priority to the air traffic management tasks than to the assessment scale answer (50% of EE and 78% of QC). Controllers generally carry out their current task and then evaluate their workload before begining the next task treatment. Nevertheless, they are satisfied with their own evaluation (50% of EE and 78% of QC). So, we can use these data as a workload index.

Moreover, the main factors that increase their workload are :

- number of planes (72% of EE and 85% of QC),
- number of conflicts (50% of EE and 70% of QC),
- strips management (33% of EE and 48% of QC).

4.2. TLX method.

The global workload has been calculated by the TLX method, for each controller and for each experiment. The figure 2 presents the global workload for each experiment.

All the controllers, except controller number 1, have estimated that the experiment without assistance was more difficult and more overloading than the others. For seven controllers (2, 3, 4, 5, 13, 14 and 15), their workload with the implicit mode was less important than with the explicit mode. For the others (6, 7, 8, 9, 10, 11 and 12), the explicit mode generates the less important workload.

Therefore, a dynamic task allocation mode (either implicit or explicit) seems to improve the air traffic control task and to reduce the global workload. The TLX method does not show clearly that engineers were overloaded without assistance tool. But it proves that all the controllers have similar feelings concerning their global subjective workload for experiments in implicit or explicit modes.

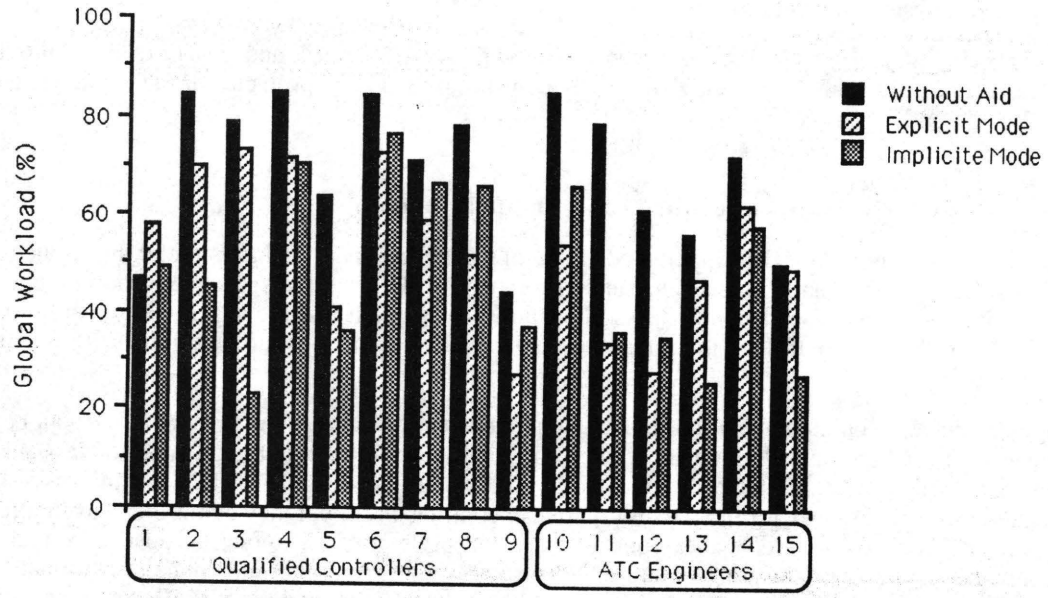


Figure 2. Global subjective workload with TLX method.

4.3. Task demands.

In order to compare the three experiments for each controller, the measured task's demands and the

subjective workload were gathered. Only one example is given on figure 3. We may use subjective workload because controllers think they they answered correctly. Globally, the values obtained with the experiments without assistance are greater than those obtained with assistance.

It is sure that each controller has his own mental representation of the graduated scale on which he has to estimate his own workload. Nevertheless, in general case, the greater the task's demands, the greater the subjective workloads and the lower the task's demands, the lower the subjective workloads. So, despite the basic difference between these measures, the correlation coefficient between the two variables have been calculated for each experiment. For 55% of EE and 63% of QC, it is greater than 0,8. For 17% of EE and 23% of QC, it is between 0,6 and 0,8. For 18% of EE and 11% of QC, it is between 0,4 and 0,6. Therefore, our tasks demands measure reflects fairly well the controllers' workload for both engineers and qualified controllers.

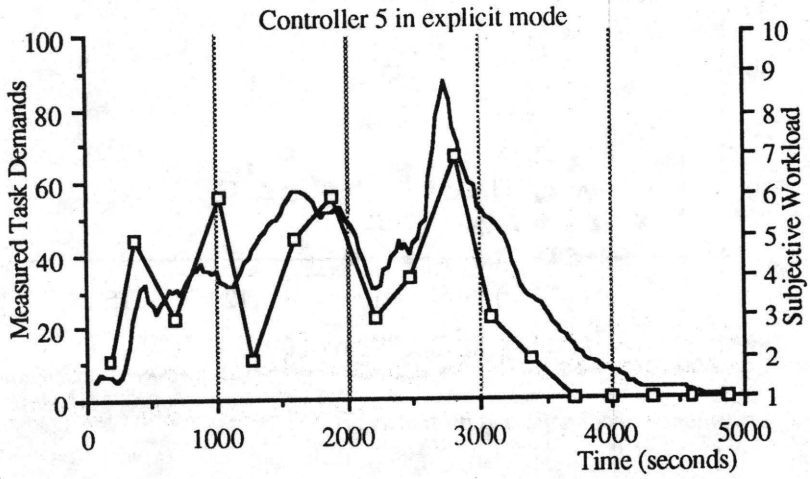


Figure 3. Comparison between measured tasks demands and subjective workload.

So, we will use this assessment for the comparison between the experiments and for the discussion that follows according to the performance criteria, the cooperation level and the tasks demands.

5. Discussion using results about performance.

5.1. Questionnaire about performance.

Performance rather satisfy for controllers (50% of EE and 70% of QC).

About insecurity criterion a lot of alarms (83% of EE and 78% of QC) appeared during experiments because the scenarios involved a lot of planes and the controller managed the traffic alone. Half of the alarms were due to a bad conflict solving because controllers had not sufficient marks on the radar view.

5.2. Economic criterion and insecurity criterion.

Figure 4 shows both the insecurity criterion and economic criterion relating to the performance. Here again, we can see that the global performance of a context without aid is not as good as global performance of a context with a dynamic task's allocation, particularly for the engineers. Moreover, we can notice that the nine experiments which have provoked no alarm at all, were performed with :

- implicit mode for one engineer and six qualified controllers,
- explicit mode for two engineers.

Therefore, the implicit mode is the most efficient one with regard to the insecurity criterion that is the most important one. Nevertheless, the results are homogeneous in both explicit and implicit modes for both engineers and qualified controllers. Indeed, in general case, those assistance tools permit to improve the security levels and to maintain the economic levels similar to those obtained without aid.

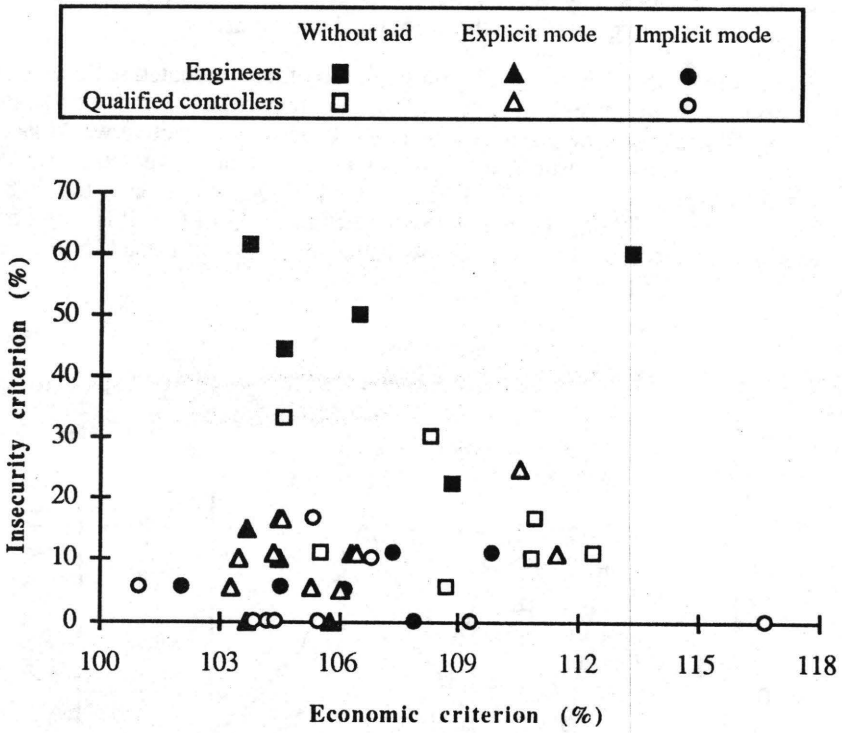


Figure 4. Global performance : economic and insecurity criteria.

5.3. Cooperation level effects.

We wanted to know if the homogeneous results concernig the global performance correspond to an homogeneous SAINTEX use. In such a way, the figure 5 presents the performance criteria according to the cooperation level that is the ratio between the number of conflicts allocated to SAINTEX and the number of conflicts appeared during the scenario. Therefore, high cooperation level corresponds to a high number of conflicts allocated to SAINTEX.

Firstly, the cooperation level in air traffic control does not penalize at all the economic level based on planes consumption. The SAINTEX performance is then as good as the controllers' ones. Secondly, an explicit or implicit cooperation integration permits to obtain a better security level in a loaded context. But the tasks allocation is really different for each controller. For the same cooperation level, controllers have obtained different security level or economic level.

In explicit mode, each controller has to allocate shareable tasks like he wants to. He has his own allocation policy. Nevertheless, global results of insecurity level are the best ones for engineers. Globally, those controllers allocate to SAINTEX more than 49% of conflicts although the qualified controllers give SAINTEX only 30% of them. This justifies the subjective remaks. Indeed, because of their active trust behaviour, qualified controllers are less satisfied by SAINTEX solving solutions. So they did not give it a lot of conflicts. On the other hand, engineers abilities for managing a lot of conflicts at the same time are more limited than qualified controllers' ones. So they used SAINTEX a lot in order to optimize the air traffic control. That is why two engineers whose cooperation level are 50% and 80% did not generated any alarm at all for two experiments in explicit mode.

In implicit mode, controllers' abilities are established automatically according to the tasks demands estimation. This evaluation is the same for both engineers and qualified controllers because it depends on a same maximum overload threshold. Therefore, in general case, more than 40% of conflicts were allocated to SAINTEX. The allocation policy permits to generate no alarm for six experiments with qualified controllers. It forced them to take in charge fewer conflicts and allowed those controllers to solve them more cleverly. On the other hand, the implicit mode were less efficient for the second class of controllers because of their abilities.

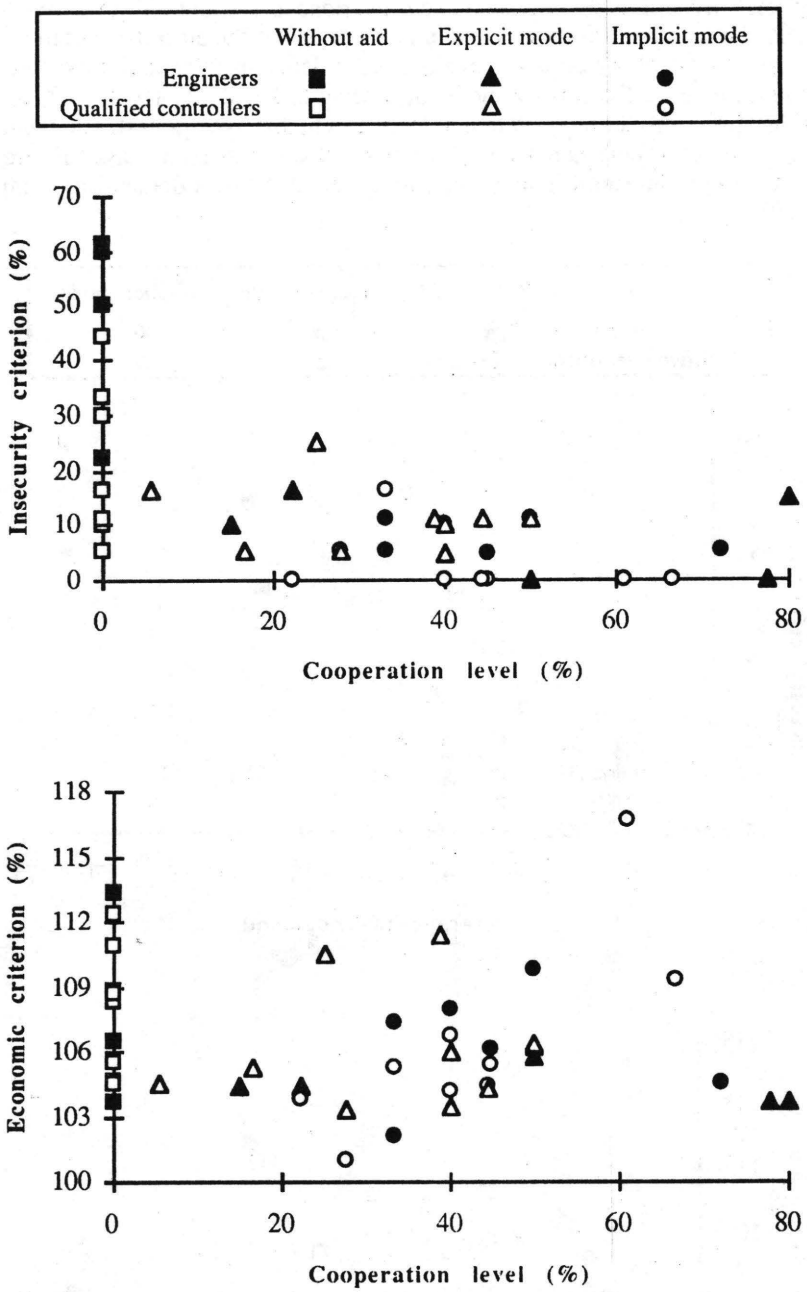


Figure 5. Global performance criteria and cooperation level.

Despite of different cooperation levels for both the implicit and explicit modes and for both the engineers and qualified controllers, the global performance results are coherent. But, what about the tasks demands ? The last part of our discussion concerns the average tasks demands study.

5.4. Tasks demands and global performance.

Figure 6 shows the tasks demands influence according to both economic and insecurity criteria. Two interesting areas can be defined : one for experiments realized without any assistance and one for the others. On the former, the more tasks demands increase, the worst the security level. During this kind of experiments, controllers, and particularly the engineers, are the most loaded. On the latter, for both explicit and implicit mode, the points distribution is similar to the figure 4.

In explicit mode, one engineer and two qualified controllers had task's demands more important than the others controllers because they allocated a little number of conflicts to SAINTEX (respectively 15%, 25%

and 5% of conflicts). Even though, engineers were overloaded without aid, their results are similar to qualified controllers' ones when an assistance tool is integrated in the air traffic control system. For each controller, according to global performance, average task's demands in explicit mode are similar to those obtained in implicit mode. In the explicit mode, according to his ability level, each controller uses the SAINTEX abilities differently in order to maintain an acceptable average task's demand. In the implicit mode, according to the controller's behaviour, the system allocates shareable tasks differently, despite the same maximum overload threshold, maintaining an acceptable task's demand in comparism with the explicit mode results.

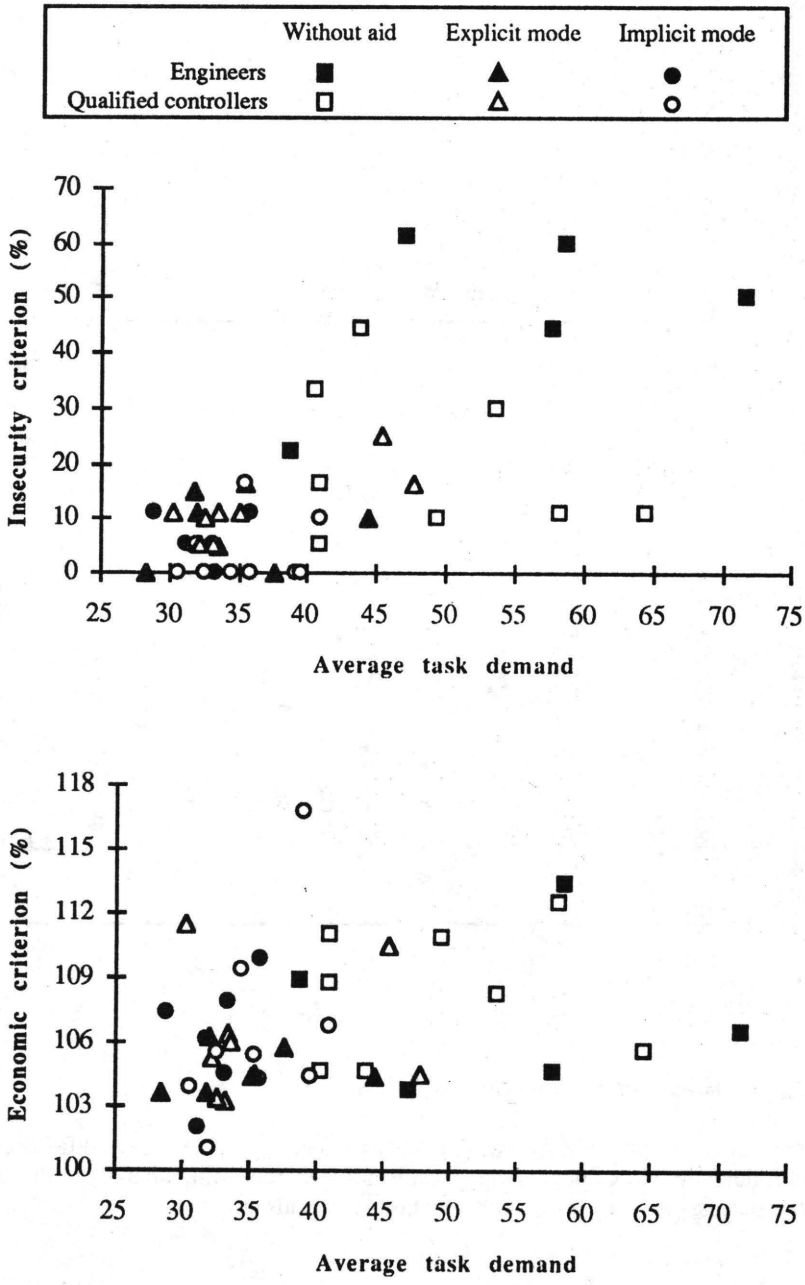


Figure 6. Global performance and task's demands.

6. Conclusion.

This paper has recalled the experimental protocol realized in order to test the dynamic task allocation principles in air traffic control using the experimental platform SPECTRA. And then, it has given an analyse of the different results obtained during experiments performed with air traffic control engineers and

qualified controllers, according to general and subjective remarks and according to workload and global performance.

The experimental results show the interest of such an assistance tool for loaded traffic control. Without any aid, the engineers were really overloaded and the traffic management was difficult. Nevertheless, they controlled fairly well the air traffic with the explicit or implicit mode. For both engineers and qualified controllers, the explicit and implicit modes permit to obtain a better security level and to provide homogeneous results concerning average task's demands, global performance and controllers' feelings. Controllers think that dynamic task's allocation is really useful for a loaded traffic context and that the implicit mode is the most efficient one. According to their different abilities and competences, they obtain homogeneous results using differently the system abilities in order to optimize at best the traffic control.

All the experiments have been made with the same difficulty level. Qualified controllers generated less alarms and they have obtained an optimal performance with implicit mode because they had more time to supervise the entire traffic and to refine their conflicts solving. Nevertheless, they prefer explicit mode, even though it generates a shareable tasks allocation extra load, because it appears less frustrating than implicit mode and it lets them remain the only responsible of the entire air traffic control. That is why they have an active trust behaviour for both the implicit and explicit allocation mode.

It is the contrary case for the air traffic control engineers. In explicit mode, their global performance is better because they allocate a lot of conflicts to SAINTEX. On the other hand, they prefer implicit mode because they don't have to manage the task allocation and because they don't supervise at all SAINTEX conflicts. We cannot really speak about trust in that case because those controllers were rather loaded in explicit mode as well as in implicit mode. In fact, they had a high traffic supervision task because they did not want to take the risk of forgetting conflicts.

Finally, this first experimental protocol on SPECTRA provide really convincing reesults. They show that controllers had to play both the tactical and strategic levels i.e. the radar and organic controllers activities. This remark introduces our future research for SPECTRA that has to include both radar and organic controllers (Debernard and colleagues, 1992) and to propose a new and more realistic experimental protocol. The organic controller has a strategical function in air-traffic control. He not only insures the coordination between the adjoining sectors but he also regulates the radar controller's workload. So, this new study concerns an anticipation and regulation module integration that has to optimize the tasks allocator policy and to regulate the radar controller's workload. This module has to be defined and tested (Vanderhaegen and colleagues, 1992). This new study announces another complex research domain : the collective work.

References.

Debbache, N. Debernard, D. Millot, P. Planchon, P. Angerand, L. (1990). Toward a New Architecture for ATC using Dynamic Task Allocation between Human and A.I. Systems. Second International Conference on Human Aspects of Advanced Manufacturing and Hybrid Automation. Hilton Hawaiian Village, Honolulu, Hawaii, USA, August 1990

Debernard, S. Vanderhaegen, F. Debbache, N. Millot, P. (1990). Dynamic Task Allocation between Controller and A.I. Systems in Air Traffic Control. 9th European Annual Conference on Human Decision Making and Manual Control. Ispra, Italy, September 10/12 1990.

Debernard, S. Vanderhaegen, F. Millot, P. (1992). An experimental investigation of dynamic allocation of tasks between air traffic controller and A.I. system. 5th IFAC/IFIP/IFORS/IEA. Symposium on Analysis, Design and Evaluation of Man-Machine Systems. MMS'92. The Hague. The Netherlands. June 9/11 1992.

Gillet, B. (1990). De l'usage-type aux types d'usagers ou l'influence des différences individuelles dans l'adaptation homme-ordinateur. Informatique et Différences Individuelles. Presse Universitaires de Lyon.

Hopkin, V.D. (1991). The impact of automation on air traffic control systems. Automation and Systems Issues in Air Traffic Control. Series F : Computer and Systems Sciences, Volume 73.

Jorna, P.G.A.M. (1991). Operator workload as a limiting factor in complex systems. Automation and Systems Issues in Air Traffic Control. Series F : Computer and Systems Sciences, Volume 73.

Kamoun, A. Debernard, S. Millot, P.(1988). Implicit dynamic allocation of tasks between man and computer based on the optimal control. VIIth European Annual Conference on Human Decision Making and Manual Control. Paris, October 1988.

Kamoun, A. Debernard, S. Millot, P. (1989). Comparison between two Dynamic Task Allocation. Cognitive Sciences approaches to process control. Siena, Italy, October 1989.

Hurst, M.W. and Rose R.M. (1978). Objective job difficulty, behavioural response, and sector characteristics in Air Route Traffic Control Centres. Ergonomics, 1978, Volume 21.

Millot, P. Kamoun, A. (1988). An Implicit Method for Dynamic Task Allocation between Man and Computer in Supervion Posts of Automated Processes. 3rd IFAC Congress "Analysis design and Evaluation of Man-Machine Systems" OULU, Finland, June 14-16, 1988.

Millot, P. (1988). Supervision des procédés automatisés et ergonomie. Edition HERMES. Paris, Décembre 1988.

Mordchelles, P. Angebault, H. Angerand, L. (1989). Système expert pour le contrôle automatique de nuit. Rapport de stage de fin d'études. Juin 1989.

Rieger, C.A. Greenstein, J.S. (1982). The allocation of tasks between the human and computer in automated systems. Proceedings of the IEEE 1982 International Conference on "Cybernetics and society", pp. 204-208. New York, IEEE, 1982.

Valot, C. (1988). Paradoxes of confidence in aid systems. Ergo-I.A. Biarritz, October 4/6 1988.

Vanderhaegen, F. Debernard, S. Millot, P. (1991). Man-Machine Cooperation in Air Traffic Control. 10th European Annual Conference on Human Decision Making and Manual Control. Liège, Belgium, November 11/13 1991.

Vanderhaegen, F. Crevits I., Debernard, S. Millot, P. (1992). Multi-level cooperative organization in air traffic control 3rd International Conference on Human Aspects of Advanced Manufactoring and Hybrid Automation. Gelsenkirchen, Germany, August 26/28 1992.

Computer Support for Cooperative Work in Space Science Activities

Giorgio De Michelis[◊#], Paolo Donzelli[*], Thomas Schael[◊¥], Buni Zeller[◊]

[◊] Istituto RSO, Milan, Italy

[#] DSI, University of Milan, Italy

[*] ISD, European Space Agency (ESA/ESRIN), Frascati/Rome, Italy

[¥] HDZ/KDI, University of Technology (RWTH) Aachen, Federal Republic of Germany

Abstract

This paper reports on a study of a number of working processes in space science activities for the European Space Agency (ESA). Methods and developments from the area of Computer Supported Cooperative Work (CSCW) are mapped to user needs for science information systems. The underlying thesis to understand user requirements for CSCW systems are first, that cooperative networks in space science work should be analyzed according to the type of cooperation: coordination, collaboration and co-decision are the basic patterns which can be identified. Secondly, dynamics of cooperative groups should be analyzed to understand the evolution of the cooperative networks: cooperation in the small vs. cooperation in the large are the proposed models. Thirdly, design spaces for CSCW systems can be identified in information sharing and in activity synchronization. The main technique for coordinating the work of a group is to provide facilities for synchronizing the various activities performed by the group, whereas the main technique for promoting collaboration and co-decision between group members is to provide information-sharing facilities.

The results of the analysis for a sample of work processes indicate a high degree of collaboration and co-decision, which reveals the need for information sharing. Coordination is more limited, but significant especially in work processes related to space mission experiments and project management. Dynamics of the cooperative groups reveals the prevalence of cooperation in the small which implies a need for specific and limited workgroup domains rather than generic and undefined communication or information needs. The results indicate a need for the development of a domain specific communication support beside already well-supported open communication in ESA's science information systems.

1. Introduction

The European Space Agency (ESA) and European space research institutes can be characterized as a networked organization with highly qualified scientists working in different locations all over Europe and with tight collaboration with scientists in other continents. This community of international space scientists represents a user group of computer networks which need a wide spectrum of services and supports for their work. One of these fundamental needs is addressed by systems managing space data which have to be consulted and interpreted by specialists and interdisciplinary research teams. The European Space Agency has also revealed a real need among scientists to communicate more efficiently with each other across Europe and with other colleagues in other continents making use of data and communication networks. The Information System Division (ISD) of the European Space Agency at the European Space Research Institute (ESRIN) is currently working on two main science information systems for the space science community: the 'Columbus Utilization Information System' (CUIS) (Alvisi & Douzal, 1989) and the 'European Space Information System' (ESIS) (Ciarlo et al., 1992). User requirements are met by applying information technologies and by taking into consideration new opportunities from innovative research and development areas. In this context of space research, user needs and demands for new science information systems, one can find solutions and design guidelines for specific application areas as well as for entierly new information systems from the area of Computer Supported Cooperative Work (CSCW).

The National Science Foundation in the USA is addressing a similar problem like ESA: the development of a *national collaboratory* as a general system to support the scientific community (Lederberg & Uncapher, 1989).

Today's scientific challenges can be characterized by the nature of the phenomena to be understood and the resources available to understand them. More than ever before science focuses on phenomena that are remote and inaccessible, are inherently distributed across space and time, and are conceptually and computationally complex.

Much of scientific research remains fiercely individualistic, with a single scientist displaying a high degree of creativity and versatility in the completion of a project from inception to tool-building, data collection, and theoretical analysis. However, even the isolated investigator works within the *social definition of discovery*, if only to demarcate what is *new knowledge*. Furthermore, what is discovered must meet some canonical criteria of significance. All this implies that a community of scientists and science is inherently a social enterprise. Its practice often requires highly organized teamwork in the coordination of highly specialized skills from different disciplines.

The scientific process, however, is not only coordination, but also a process of team science - the collaboration among scientists working on a common problem. This may be tacit *collaboration* of minds reached by searching and reading scientific literature, *explicit bilateral exchange* of ideas, tips, etc.. It may be a *formal cooperation* over extended periods of time, with contractually agreed division of labour and allocation of credit.

2. Cooperative networks in space science work

In order to analyze the main processes in space science work within the perspective of CSCW, the authors applied a methodology (TicutomiNet by RSO) which supports the identification of the main components of processes and of the types of cooperative networks. Within this methodology, three main types of cooperation are identified in order to distinguish different cooperative networks: coordination, collaboration and co-decision (Schael & Zeller, 1990). These fundamental types of cooperation help to describe aspects which are substantially different in terms of the nature and the scope of

work, the relationship between the participants, and the types of communication and information occurring within the group.

2.1. Identified work processes

The authors identified a number of scientists' working processes. The material for the analysis was gathered from documents and interviews with system developers and scientists from the space science community. The argumentation in this paper will not exploit all aspects for all processes, but give some examples for possible generalization in the development of science information systems. The following processes are a sample of processes which have been identified as a result of meetings with users and system developers; they cover a range of typical working processes in ESA's user comunity.

Experiment life-cycle processes:
- Announcement of Opportunities (AO): the AO is released by the Agency and users decide whether they are interested in participating or not.
- notification of interest: exploration of interest and opportunities for a possible experiment with a later constitution of a consortium in reply to a call by the Agency
- payload selection: verification of resources availability and selection of proposals
- payload design and development: development and initial testing of the payload
- mission preparation: experiment plan and all servicing and logistic aspects are finalized
- mission operation: experiments are performed on board, data is captured and stored, preliminary analysis of experiments

Post-experimental processes:
- research and study activities: process of modelling, data analysis, idea generation, etc.
- co-authoring: paper production and publications after the experiment where the results are written up. Each scientist has a well defined role in the writing process according to his competence or special interest defined during the experiment life-cycle.
- development of special tools: software or instrument development

Generic processes:
- re-use of software and/or data: exchange of models and/or data among scientists
- data research: looking for data on a specific issue
- finding opportunities for cooperation: establishment of interest groups, consortia, etc.
- looking for help: finding solutions to a problem/breakdown
- project management: managing scientific projects

2.2. Cooperative nature of the identified work processes

The matrix below summarizes the results of the analysis of the cooperative characteristics of the processes discussed here. The table makes evident that the processes have generally a very high degree of complexity. This implies by its own nature the need for a high level of information sharing, beside a varied level of activity synchronization which is a direct function of the coordination characteristics of some of the networks.

Characteristics / Process	result* (dec/rea)	com-plexity	inform. sharing	activity synchr.	type of network
Announcem. Opport. (AO)	↗/↘	↗	→	↗	co-dec + coord
notification of interest	↗/↘	↗	↗	↘	co-dec
payload selection	↗/↘	↗	↗	→	co-dec
payload design	↘/↗	↗	→	↗	coord + collab
mission preparation	↘/↗	↗	→	↗	coord + collab
mission operation	↘/↗	↗	→	↗	coord + collab
research activities	↘/↗	↗	↗	→	collab
co-authoring	↘/↗	↘	→	↗	coord
development of tools	↘/↗	↘	↗	↗	coord + collab
re-use software/data	↘/↗	↗	↗	→	collab
data research	↘/↗	↗	→	↘	collab
opportunities for coop.	↗/↘	↗	↗	↘	co-dec
looking for help	↘/↗	↗	↗	↘	collab
project management	↘/↗	↗	→	↗	coord + collab

The importance or presence of the characteristics of the processes are classified into: ↗: high, →: medium, ↘: low

2.3. Spaces for design in cooperative processes

On the basis of the previous points, the authors propose two design spaces for cooperative work (De Michelis, 1990, Schael & Zeller, 1991). The main technique for coordinating the work of a group is to provide facilities for synchronizing the various activities performed by the group, whereas the main technique for promoting collaboration and co-decision between group members is to provide information sharing facilities.

Striking the right balance between visible activity synchronization and information sharing is critical to improving the productivity of cooperative groups. Too little synchronization makes the group inefficient and often produce a duplicated effort, too much synchronization can lead to over-formal communication between group members. This reduces the effectiveness of working as a group. Likewise, too little information sharing makes the group ineffective, while too much sharing leads to inefficiency (De Michelis, 1990, Schael & Zeller, 1991).

The first step in a groupware implementation is therefore to identify the optimum balance between activity synchronization and information sharing, which will vary according to the nature of tasks to be undertaken in the cooperative network and the capabilities of the individuals in the group. It will also be necessary to assess the current levels of synchronization and sharing in the group. The final stage is to identify the support tools which will provide the optimum balance between activity synchronization and information sharing.

Activity synchronization and information sharing have to be understood here as malleable and visible systems, as opposite to predefined and transparent systems. The latter systems are suitable when people act in a well defined and recurrent environment, but are not appropriate when people need to cope with breakdowns in a complex environment. In such a case the transparency of a system is not effective it because does not provide the way (i.e. the rules, the tools, etc.) to change the system behaviour. On the contrary, visible systems should allow people to self-reset their behaviour, for example by

* the final result of a process can be a decision or a realization; the column contains the relation decision/realization

renegotiating the timing or by re-routing an activity (or an item of information). Of course, visible systems are less efficient then transparent systems, to manage a stable process, but are more effective to cope with the turbulence and complexity of breakdowns. Therefore, an optimal system should be able to offer transparency or visibility when needed.

The problem of cooperation in the small and cooperation in the large is the ongoing process of large groups becoming small, the community dividing itself into sub-groups and small groups growing up to large groups or looking for help (information) in the whole science community. This dynamic has a certain impact on the development of IT systems and services provided to space scientists because it has to take into consideration the cooperation in the small as well as in the large. This has to be combined with the need for activity synchronization and information sharing within the processes.

3. Dynamics of cooperative groups

An experienced researcher participates in different groups: they is a member of research units, divisions, etc. in his/her organizational unit of the laboratory or department; they participates in local, national, international, etc. research project teams where they has some specific tasks; they is a member of formal committees (national and international advisory committees, editorial committees of scientific journals, steering committees of scientific institutions and/or scientific communities, etc.); they takes part in groups created to perform a defined task (organization and program committees of scientific conferences, selection committees, evaluation teams for research programs, etc.). Each of these groups is characterized by the particular form of cooperation occurring among its members.

Furthermore, experienced researchers are frequently involved in occasional interactions with other members of the scientific community: they can be asked to help with respect to a topics they are experts for; they might need help with a problem they are facing within their activity; they can be involved in the creation of something new: a journal, a research project, a laboratory, etc.; or they can receive a proposal to move to another research laboratory, department or institute. These occasional relations with other members of the scientific community aslo show patterns, which distinguish them from others.

As a result, space scientists and researchers in general interact with persons of two main categories: persons belonging to one of the *groups* they are members of, and the other persons in the research community. Scientists have an ongoing communication with persons belonging to the first category: every new conversation has some relation with the previous ones, and there is a strong continuity in the history of their interactions. On the other hand, scientists interact in an episodic way with the other ones. There is no continuity. They sometimes need to open a conversation with one of them, but every new conversation with the same person has little to do with the previous ones.

Let us call the cooperation scientists have with persons of the first category, *cooperation in the small*, and the other one *cooperation in the large*.

3.1. Cooperation in the small

As we have mentioned, *cooperation in the small* occurs within already established groups, which might be characterized by relatively well defined roles, rules and a common history. This creates a setting where every new conversation is related to previous ones, where relations between two persons assume meaning for other members of the group, and where people do things together.

The boundary between such groups and the rest of the world is generally well-defined. This is due to the fact that each group member has a role within the group and everybody knows what the group members do together. There is also a continuity in the group because its past experience influences its future.

Within cooperation in the small two kinds of groups can be distinguished, which cover most of the variety of the small groups researchers are members of: Stable organizational units and temporary project groups.

Stable organizational units

Each experienced researcher generally participates in various organizational units which are stable. Examples are research units of a laboratory or department; divisions of a large research institution; committees guiding or controlling journals, other periodical publications or research programs; committees advising the Agency, governments or intergovernmental institutions. Some of these stable groups' activities occupy most of the researcher's working time, while others are characterized by rare interactions; some of them have a deep influence on what they do day by day while others define a limited and well distinguished field of activity; some of them are fully transparent for them while others appear with a high degree of opacity; some of them are important while to others they pay little or no attention.

Temporary project teams

Each experienced researcher participates also in many different temporary teams, as in a small research project; organizational committees for conferences or program committees; selection committees for an academical position or a scientific award, etc. Temporary teams are different from stable organizational units: they are generally oriented towards actions. The group has one (complex) thing to do and every person has a task/role within the team. Whenever a temporary team does not do anything, its existence is likely to lose interest, and it disappears.

4.2. Cooperation in the large

Cooperation in the large characterizes interactions a researcher has within his/her scientific community; an experienced researcher interacts not only with persons from groups s/he is a member of. The researcher frequently communicates and cooperates with other colleagues in an occasional manner during his/her activity. The community of the persons with whom they can have occasional interactions, is very large, is loosely coupled and does not have well defined boundaries. The persons in those interactions can vary in time, a new interaction can enlarge their number without any formal statement, and some of them can be lost due to lose relational patterns.

Interactions occurring within these ever changing communities are related to the subject and to the goals. The whole community is a world where people 'live' with or without any defined or specific roles to play. Researchers are looking for the right person(s) who can or want to do something with regard to their tasks. The community remains quite anonymous for the single scientist and there is no memory of past experience which might directly influence the future.

There are two main aspects to be addressed for cooperation in the large among researchers: The establishment of groups, and exceptional help.

Establishment of groups

Whenever a researcher, or several of them, want to start a new initiative - a journal, a research project, a research laboratory - s/he opens conversations within his/her scientific community. Within these conversations they discuss the features of the new initiative, the conditions under which the contacted persons could possible participate, the skills or profiles of other persons to be involved, etc. Generally speaking, the scope of their interaction is to generate conditions and groups, which make it possible to do something together in the future. While looking for people to create a group it is possible that some conversations are opened without indicating the partner. This means that the call is addressed to everyone.

Exceptional help

Scientific work is characterized by a high number of breakdowns which a researcher has to face during his/her activities. An experienced researcher frequently discovers that something is missing to accomplish his/her task: an item of technical information, a scientific reference, a new paper, a name or an address of an expert in a specialized field, etc. In all these cases he opens conversations within the scientific community to overcome the problem. At the same time it can happen that s/he discovers something which can help a colleague, or has been asked to him/her before. In these cases s/he opens conversations with his/her colleagues to distribute the information s/he has found. Also in the case of exceptional help some conversations are opened without indicating the partner (addressed to everyone).

The following matrix shows the types of cooperation related to the size of groups in the community (cooperation in the small or in the large) and the dynamics in changing from one type to the other.

There are four very dynamic processes (notification of interest, research activities, opportunities for cooperation, looking for help). The greater part of processes remains usually in a defined domain and cooperation in the small is largely prevailing. This result should be taken into serious consideration because it implies a need for specific and limited workgroup domain rather than generic and undefined needs of communication and information search.

Characteristics / Process	Cooperation in the small	Cooperation in the large	Dynamics in changing small/large
Announcement of Opportunity	↘	↗	→
notification of interest	→	↗	↗
payload selection	→	→	→
payload design	↗	↘	↘
mission preparation	↗	↘	↘
mission operation	↗	↘	↘
research activities	↗	↗	↗
co-authoring	↗	↘	↘
development of tools	↗	↘	↘
re-use software/data	↗	↘	↘
data research	↘	↗	→
opportunities for cooperation	↘	↗	↗
looking for help	→	↗	↗
Project Management	↗	↘	↘

The importance of the cooperation types and dynamics are classified according to:
↗ : high, → : medium, ↘: low

3.3. Support functionalities for cooperative groups

The functional needs can be divided in the two domains of *communication* and *information management* specifying functionalities which reflect this dynamics of growing and decreasing groups in different cooperative processes.

Communication

Communication can be further divided into:
- *open communication* (mail, teleconference, bboard, etc.) for cooperation in the large;
- *domain communication* (stable groups with common workflows in a specific context, e.g. a project) for cooperation in the small.

Information management

Information management can be further divided into:
- *open search for information* (relational DBs and concept search tools);
- *information search based on events* (workflows with an electronic mail system as its backbone and information objects' history modeled by hyper-links).

Giving the results of the analysis of the characteristics of the cooperative networks and of the dynamics of groups, the following table suggest a preliminary view of the functional needs, where:
- open functionalities are associated with work processes which are characterized by cooperation in the large or by highly dynamic cooperation in the small;
- domain and event functionalities are associated with work processes characterized by stable cooperation in the all or by highly dynamic cooperation in the large.
- open search and event search functionalities are associated with work processes characterized by information-sharing needs;
- domain communication and event search functionalities are associated with work processes characterized by activity synchronization needs.

Functionalities / Process	open commun.	domain commun.	open search	event search
Announcement Opportunities	X	(X)	(X)	(X)
notification of interest	X	(X)	X	X
payload selection	X		X	X
payload design		X		(X)
mission preparation		X	(X)	(X)
mission operation		X	(X)	(X)
research activities	X	(X)	X	(X)
co-authoring		X		X
development of tools		X	(X)	X
re-use software/data		(X)	X	(X)
data research	X		X	
opportunities for cooperation	X	(X)	X	(X)
looking for help	X	(X)	X	(X)
Project Management		X		X

X : important functionality
(X) : less important functionality

The matrix shows that most of the identified processes would benefit from domain specific communication support, while open communication support is associated with a limited range of work processes. Event-based and open information search are widely distributed among the processes identified and are often required together.

4. Development of a service portfolio

A service portfolio was developed, taking account of the different work processes analyzed. The science information system should support each of these processes and should integrate systems for break-down resolution. Some of these services for space scientists will be general (standard applications for the whole community) – the *open services* – while others relate to sub-groups (customized applications for specific user groups or domains in the community) – the *domain services* .

Open services

Open services are aimed at supporting cooperation in the large (establishing groups, looking for help). Open services should be conceived as basic services to support communication and information retrieval in the wide community. Open services like Email, teleconferencing and bulletin boards for group communication, and retrieval services for information management, are appropriately addressed by the present ESA's information systems ESIS and CUIS. Therefore open services are not further discussed in this paper. Nevertheless we put emphasis on the need to integrate open services with domain services in a way which enables the users to shift from one type to the other, as is typical in the dynamics of change between cooperation in the small and in the large.

Domain services

Domain services are aimed at supporting cooperation in the small (stable organization units or processes, temporary groups). Domain services should be conceived as specific services to support communication and information sharing within specific groups or processes; this is the type of functionality that is poorly addressed by the present ESIS and CUIS.

The types of services and types of functionalities can be crossed as described in the following matrix, where some examples of possible functionalities are reported:

Service\Functionality	Communication	Information management
Open services	Email, Teleconf, Bboard	Information retrieval
Domain services	Workflows$_{1, n}$	Hyper-links, Event research

The following table shows the level of coverage of the current ESIS and CUIS functions with respects to the work processes identified:

Current systems / Process	ESIS	CUIS
Announcement Opportunities	open services	open services/questionnaires
notification of interest	open services	open services/questionnaires
payload selection	open services	open services/questionnaires
payload design	not foreseen	currently not developed
mission preparation	not foreseen	currently not developed
mission operation	not foreseen	currently not developed
research activities	open services	open services
co-authoring	currently not developed	currently not developed
development of tools	currently not developed	currently not developed
re-use software/data	open services	open services
data research	fully supported	fully supported
opportunities for cooperation	open services	open services
looking for help	open services/directory	open services/directory
Project Management	not foreseen	possibly in the Activity Office

ESIS and CUIS are well developed systems for open communication services and search for information in relational data bases. Therefore both system are a valid support for all processes which mainly need open communication. The same counts for the need for open information search. However, among the processes identified open services are exclusively needed only for the 'data search' process. All other processes need domain communication and/or event-based search functionalities in combination with open services. Furthermore, there are processes which mainly need domain communication

services and/or event-based search facilities. These processes are not at all well supported with the current systems.

4.1. Domain service portfolio

Domain services are the area in which significant innovation is possible for ESA's information systems, as seen from the comparison of domain service requirements and support levels in ESIS and CUIS. We provide below some guidelines to approach this area of domain communication and domain information management.

Domain workflows

A *workflow* is a unit of work that happens repeatedly in the organization of work. Workflows reflect the recurrent organization of work, but their organizational and communication patterns are based on both standard and *ad hoc* actions taken by specific persons in order to fulfil a particular condition of satisfaction requested by someone. Some workflows are formally defined. Others are inherited from experience, without any explicit design.

The basic unit of processes as a four step action workflow is described in the following figure where black arrows represent the requestor's speech acts and gray arrows the actor's speech acts.

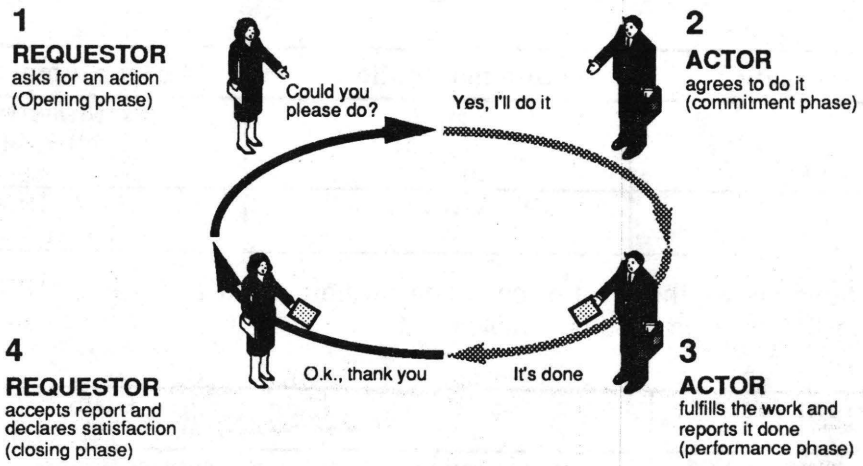


Figure: Basic action workflow (ATI, 1990; Dunham, 1991)**

The underlying model sees persons as continuously communicating individuals who open and close conversations in their working life relations with colleagues and others to accomplish their tasks in the work processes. The flow goes through the phases of opening, committing, performing and closing.

A different type of workflow can be identified when instead of a request for an action an exploratory or declarative conversation is opened. In such cases the communication flow is less structured and can be seen as recurrent communication steps between the participants which usually end with a mutual final declaration.

A software for modelling workflows can be seen as a 'toolbox' which allows the generation of applications supporting more or less structured processes and procedures. This electronic process support will address domain specific services and might be implemented e.g. in the Columbus Activity Office (CAO). For the Columbus Activity Office there is a general need to monitor progress in the defined processes. This also concerns

** from Action Technologies Inc. Business Design Language

the information that material sent officially from the agency has been received by all parties.

CUIS has taken into consideration the development of a prototype for the automation of activities related to handling questionnaires and forms and to extracting requirements for analogous developments.

We will develop an example for a much used process in different areas of the Agency, the Review Item Discrepancy process, called RID. The RID process is initiated by a Technical Officer who will ask a third party to review a deliverable (software, document, instrument, etc.). The reviewer records his/her comments about the reviewed item on RID forms. The reviewer can also recommend a solution and insert it on the same form. The Technical Officer will pass the form to the author/deliverer of the item for his/her statement on the issue. In a final meeting a decision about the review discrepancy closes the RID process. The process is modelled in the following figure as a workflow with connections to the stored data in the information technology system.

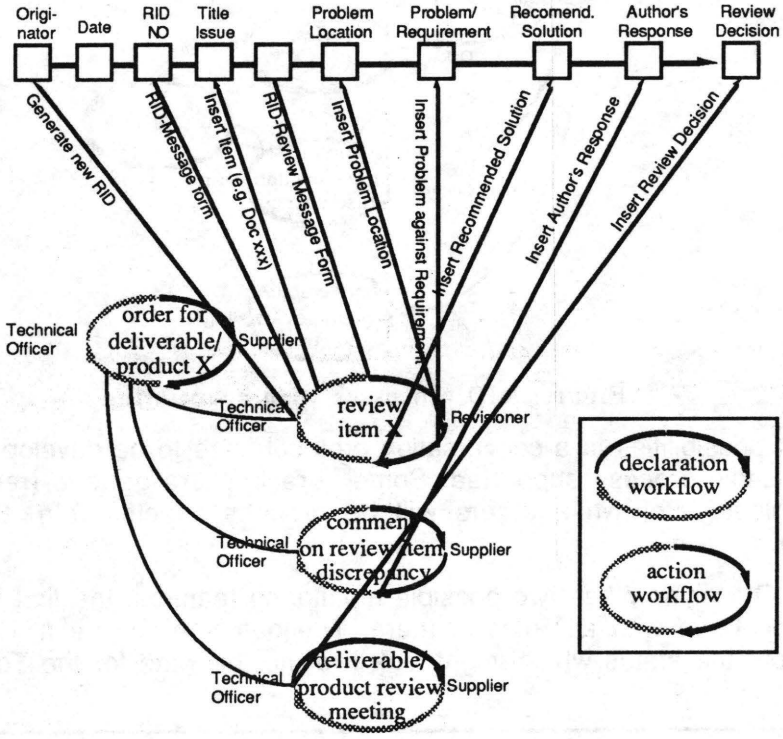


Figure: Workflow and information processing in the RID process

The combined modelling of information and conversation flows makes it possible to trigger the information about a specific RID in the RID-information-base, and the status of the process in the RID-conversation-base (Schael & Zeller, 1992). The four steps in each action-workflow (opening, committing, performing and closing) are flags which help to understand the progress in the RID-process management. This concerns both the three main sequential steps of review, comment and meeting, as well as the steps within the single workflows. The previous figure showed only the basic workflows. Every phase can have secondary workflows which will be mainly negotiations and delegations by the involved persons. The following figure shows the case of negotiations among the requestor (Technical Officer) and his actors (Reviewer in the review flow and supplier in the comment-on flow).

The steps for completing a RID expand with these quite obvious escalations of the basic workflows but are not reflected and supported in the present developments. Let us make an example for a hypothetical RID process. The Technical Officer generates a new RID and sends it to a reviewer. After the officer's request the reviewer should find several

alternative answers on the communication system which supports the RID workflow. Possible answers are e.g.:

1.) Yes, I'll do the review (promise)
2.) OK, except that ... (counteroffer)
3.) No, however ... (counteroffer)
4.) Here is my review (work done)
5.) I cannot/will not do the review (decline) (Winograd & Flores, 1986)

The first choice (promise) would bring the workflow in the simplest way to a completion; the reviewer will fill his bids of information in the RID (reports completeness) and the Technical Officer will thank him for the review (closes the workflow). The reviewer might also directly send the complete RID to the Technical officer (4th choice) without a prior explicit acceptance. The second and third choice will open a negotiation about the review process. The final option will simply decline the Officer's request.

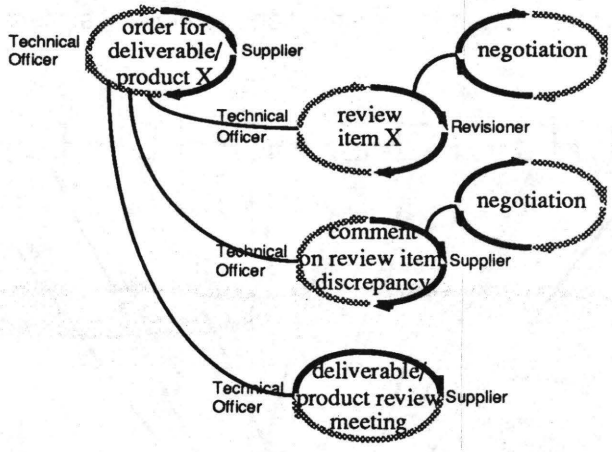


Figure: RID processes with negotiation

These different possibilities in a conversation protocol have to be developed according to the domain specific process supported. Some iterations are generic (request, promise, counteroffer, close, etc) while others will be process specific (e.g. the RID review message form).

The Technical Officer now has two possible monitoring features. the first is the review of his conversation base with all RIDs in there. The outcome will be a list of RIDS with information about the status which might look like the following for the Technical Officer:

Date	Originator	Type	Receiver	Status	Object
1-2-91	Officer	request	reviewer 1	accept	RID 123-item abc
3-2-91	Officer	request	reviewer 2	work done	RID 124-item cde
4-2-91	Officer	request	supplier X	counteroffer	RID 103-item aaa
5-2-91	Officer	request	supplier Y	closed	RID 110-item aab
9-2-91	Officer	declare	supplier Z	open	RID 101-item ccc

This list explains the following: reviewer 1 agreed to review RID 123 and the Technical Officer is waiting for the filled in form. Reviewer 2 has already sent him the form and the Officer has to verify it and close the conversation with him. Supplier X has entered into a negotiation about RID 103 by making a counteroffer. The review and commenting phases for RID 110 have already been closed and RID 101 is in the review meeting phase.

Domain specific and event-based information search

Problems concerning information management are relevant in cooperative processes. This is true in general for the production, revision, filing and distribution of documents.

This is also true when cooperation is not directly related with creating and modifying documents, because people collaborating on a common task need to share the information characterizing that task. The re-use of information in the science community depends on the capacity of the organizational and technological support. Developing Space Science Information Systems for cooperation means to introduce tools which allow people to structure information in a way that reflects how it was created or exchanged between different persons.

When we access an information source for the first time, we adopt a 'logical' criterion of selection. However, when we access information we have created together with others or we have accessed before, the natural method is to reconstruct the process by which it was created, or used for the last time. Concepts and techniques related to hypertext provide a promising model for coping with this issue.

Let us make an example: When a researche needs a document which s/he remembers related to a specific person during a conference and a following discussion about its content on the Email, the scientist has two possibilities for his/her search. The search in structured information storages (bibliographic index in the library, personal conference folder in his shelfs, conference proceedings, etc.) is one possible way to find the needed document. The structured retrieval methods are efficient, but not always successful. In this case he has to trigger back to the event. Consulting again his Email messages in the period around the conference he might find out about the person who wrote the document. He might also find the message with which the paper was sent to him as a file.

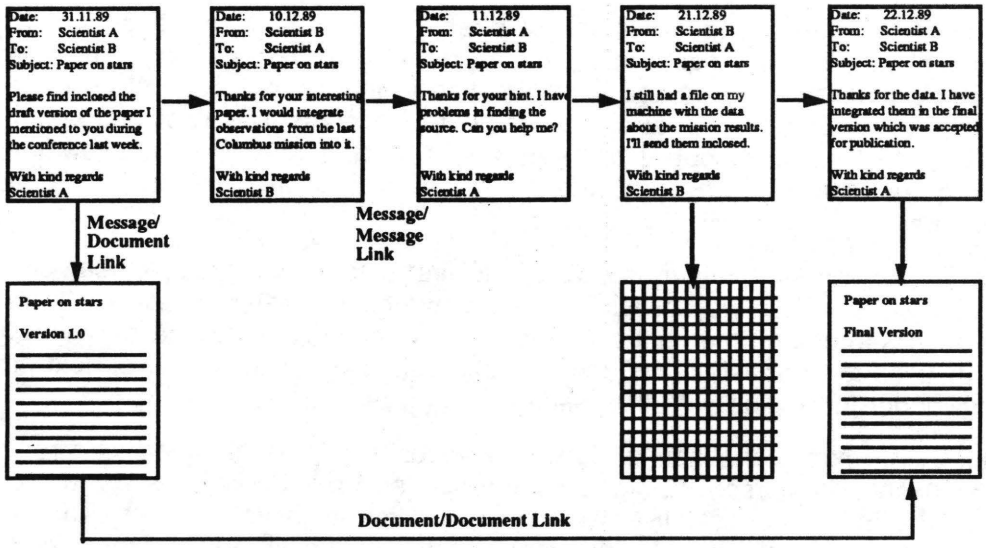
A system which supports this event-based research will be based on communicative events and on information-treatment events. The basic concept for system design is the linking of messages and information objects in order to exploit the overall context of these single events.

The first type of link is among messages because the unit of communication is not a message, but a conversation, which is the set of related messages identified by a subject. A conversation relates, e.g., to a commitment under negotiation or execution. People can better understand the event and related messages if they can re-view the history of the conversation. This also makes it posible to review the whole conversation by finding only one message and looking forward and backward by following time-sequential links.

The second type of link is between different information units following the hypertext philosophy.

The third type of link is probably the most important for event-based information search. It is the link between the message and the information (document) related to it. Taking our example, the file of the document has to be directly accessible from the message it was sent with and all the related messages in the conversation. This would make it possible to retrieve the document starting from the communicative event. If it is a text file, it can be shown in a window together with the message in a second window. The scientist could scroll in the paper and follow again the history of his discussion in the other window. If there are more versions of the paper or other documents related to this discussion, they can all be linked to single messages and the conversation. This would mean that they too can be displayed in further windows on the scientist's screen.

This concept of links for event-based information search should be combined with the domain specific workflows. This would mean in the case of the RIDs that e.g. the Technical Officer can see the history of a RID as linked messages in the workflow, open related documents by message/document links and consult the RID-database following document/document links.



Types of links for event-based information research

Conclusion

To implement the proposed guidelines for a service portfolio, the following CSCW technologies should be considered: Workflow technology and Hyper-linked data-bases are a technical response to the lack of domain services identified. Workflow technology supports processes within specific domains. Workflow management technology addresses the combination of communication and information flows in a process. It controls the routing, commitments taken and provides status reports on the work in progress. A hyper-linked data base can be created and maintained that tracks the team interactions and provides an ongoing record of the related interactions and information objects. This would provide a long-term record of messages, documents, research projects, all connected by hyper-links. Electronic mail along with the support for structured interaction (workflows) can be coupled with hyper-links tools.

Another area of innovation is addressed by CSCW technology for remote collaboration. Developments in this direction are screen-sharing technology and multimedia conferencing. Screen sharing technology allows users to see on their own workstation what is on the other people's screens. Some products make it possible to view and control part or all of the screens (file-transfer), while more sophisticated systems allow more complex and structured activities providing specific support for screen manipulation.

Multimedia conferencing focuses on collaboration in meetings and remote working sessions, such as between scientists working together at distance. The scientists' workstations include shared windows, a remote video and high-quality audio link. The technological setting for meeting suites usually includes a computational whiteboard.

References

Alvisi G., Douzal M. (1989) *CUIS: The Columbus Utilization Information System*, ESA Bulletin No. 59

ATI (1990) *Essentials of Workflow Management Technology*, Action Technologies Inc., Emeryville, California

Ciarlo A., Giommi P., Torrente P. (1992) *Access, Retrieval and Analysis of Data from Large Heterogeneous Databases: The ESIS Approach*, in Proceedings of the International Space Year Conference on Earth and Space Science Information Systems, JPL, Pasadena, California

De Michelis G. (1990) *Computer Support for Cooperative Work*, Position Paper, Butler Cox Foundation, London, October 1990

Dunham R. (1991) *Business Design Technology: Software Development for Customer Satisfaction,* in Proceedings of the 24th Hawai International Conference on Systems Sciences, Vol. III, IEEE Computer Society Press, New York

Keen P.G.W. (1991) *Shaping the Future: Business Design through Information Technology*, Harvard Business School Press, Boston

Lederberg J., Uncapher K. (1989) *Towards a National Collaboratory*, Report for the National Science Foundation

Schäl T., Zeller B. (1990) *A Methodological Approach to Computer Supported Cooperative Work.* In Proceedings of the Fifth European Conference on Cognitive Ergonomics, Urbino, Italy, September 3-6, pp. 291-304

Schäl T., Zeller B. (1991) *Cooperative Office Support Systems for Distributed Process Management*, in Verrijn-Stuart A. et al. (editors). *Support Functionality in the Office Environment*, North Holland, Amsterdam, pp. 85-101

Schäl T. (1992) *Information Systems in Public Administration: From Transaction Processing to Computer Supported Cooperative Work*, paper presented at the IFIP WG 8.5 Working Conference on Computer Supported Cooperative Work and Public Administration, Linz, Austria, February 25-27

Winograd T., Flores C.F. (1986) *Understanding computers and cognition - a new foundation for design*, Ablex Publishing Corporation, Norwood, New Jersey

A Cultural-Cognitive Approach to Collaborative Writing

Charles C. Wood

School of Cognitive and Computing Sciences,
University of Sussex, Falmer, Brighton, BN1 9QH, United Kingdom.

&

Rank Xerox Cambridge EuroPARC,
61 Regent Street, Cambridge, CB2 1AB, United Kingdom.

ABSTRACT

In the last decade, studies of writing have focussed on the internal writing processes of solitary authors. However, it is not clear that a purely cognitive approach, with its concentration on hypothetical internal representations and cognitive processes, is most appropriate to guide the design of individual writing systems, let alone collaborative authoring systems. This paper argues for a cultural-cognitive approach to the study of human activity, which yields insights which are more informing for designers of systems than purely "classical" cognitive ergonomics. An approach, *cultural-cognitive ergonomics,* which focuses on the group and its attendant cognitive artifacts as a cognitive system is outlined, and the notion of the mediation of cognition through collaborators and artifacts is discussed. Some illustrative examples from a study in which participants worked closely together on a conceptual authoring task using a shared work surface are presented and the phenomena that emerge are interpreted within this cultural-cognitive perspective. Finally, it is considered how this approach might inform the design of systems for collaborative writing.

1. Introduction

Physical ergonomics and anthropometrics study human physiology and attempt to inform the design of artifacts and work environments such that they have a good "fit" with the human body. Similarly, cognitive ergonomics attempts to use an understanding of human cognition to inform system design. But while the human body "stops at the skin" I will argue that mind extends beyond the physical confines of the body, and so a *"cultural-cognitive ergonomics"* is called for. Cognitive ergonomics draws on cognitive psychology and experimental psychology, which have been concerned with mapping out aspects of the individual mind, including perception, attention, memory, reasoning abilities, etc. Cognitive ergonomics draws on cognitive science and artificial intelligence, where cognitive tasks have been modelled in terms of internal causal mechanisms with little regard for the role of the environmental situation in cognitive activity or for the context provided by co-workers.

In recent years, systems to support collaborative work have been made possible by advances in computing and communication technology and so there is greater need to consider the cultural and social context in which cognitive work is done. At the same time approaches which focus on the group and its associated cognitive artifacts, which I will refer to generically as shared cognitive approaches, have emerged (or re-emerged, see below), and researchers in Human-Computer Interaction (HCI), and especially in Computer Supported Collaborative Work (CSCW), are displaying increasing interest in these approaches. Recently, several systems supporting collaborative authoring or editing have been developed (Leland et al, 1988; Neuwirth et al, 1990, ShrEdit, 1990) and in this paper I take the design of authoring support tools as an example domain in which to illustrate the argument for a cultural-cognitive ergonomics. In the next section I briefly characterise a cognitive model of writing which has been influential among designers of

systems, and mention some criticisms of it. In section three I outline the central features of the anthropological and cultural-psychological approaches which contrast with the "classical" cognitive approach. Here, I introduce the notion that cognition is mediated in a dialectical relationship with cognitive artifacts and collaborators such that its character, structure and functionality is radically changed. This theoretical stance suggests we will understand cognitive tasks better if we study groups working closely together using external artifacts to help them think, and in section four I describe such a study. In the following sections I consider aspects of the mediation of cognition using examples from this study, demonstrating that cognitive "faculties" of a very different nature than individual cognitive faculties are distributed throughout the system as a whole. In conclusion, I briefly suggest how this cultural-cognitive ergonomic approach might inform the design of systems to support individual and collaborative authoring.

2. The cognitive approach to writing

In the last decade studies of writing have focussed on writing *processes* of *solitary* authors. In the tradition of cognitive science, these have often used verbal protocols, together with an analysis of the representations subjects use, to induce the internal cognitive processes producing the writer's behaviour. An influential example of the cognitive approach is Hayes and Flower's (1980) use of verbal and written protocols of writers to build an internal cognitive model of writing, incorporating various sub-processes which can be called in different orders by a control process, allowing the model to account for different writing styles and strategies. The sub-processes are represented in terms of more primitive commands which operate on the knowledge represented primarily in the writer's semantic memory. Semantic memory is assumed to be some propositional notation incorporating concepts, relations and attributes, perhaps represented in a network of nodes connected by associations (Caccamise, 1987). Although the Hayes and Flower model may be virtually unfalsifiable, as given the right control structure virtually any protocol could be accounted for, it has provided the most important conceptualisation of the writing process in cognitive science.

Holt (1989) criticises such cognitive models, which treat writing as a problem solving activity with a solution that can be related to the problem through a series of cognitive processes. Holt argues that these models do not place sufficient emphasis on the interaction between the writer and the external representations used in the activity (notes, plans, drafts, etc). Also, where writers are forced to attempt to verbalise internal processes they would not normally attend to, which may in any case be inaccessible to consciousness, this may interfere with their normally unconscious activities. In this paper I argue that though purely cognitive models may be interesting from a psychological point of view their concentration on the internal has limited their usefulness in system design, as computer systems exist to support and manipulate *external* representations with which users interact. Even where system designers of authoring systems have used cognitive models, it has not really been the internal *cognitive* aspects that have informed design, but rather what the cognitive model implies for the use of external representations.

Two recent authoring support systems draw on the work of Flower and Hayes, providing the writer with a hypertext-browser-like network view of the text for arranging and linking ideas, and linking this to a hierarchically structured outline view, and a conventional editing view. Sharples and O'Malley (1988) suggest that the "Writer's Assistant" might "externalise cognition" (Ibid, p276) and in later work (Sharples and Pemberton, 1988) it is argued that cognitive models must be extended to say more about the "intermediate representations" (notes, networks of idea labels, etc) which are a bridge between internal mental structures and the final text, and about the external media that support those representations. The "Writing Environment" (Smith, et al, 1986) is similarly described as an "intelligence amplification" system (Smith and Lansman, 1989, p18) whose different views are designed to correspond with the external "intermediate products" (Ibid, p33) that are generated in writing. In CSCW, where we must understand not only the role of the external environment or the computer system in the cognitive activity, but also the interaction between different individuals, a purely cognitive approach is yet more inadequate.

These critiques of the work of Flower and Hayes provide many useful insights, particularly emphasising the importance of external representations in the writing process, though their language seems to attempt to conceptualise external activity in rather simple "classical" cognitive terms. For example, though an external network diagram of ideas may capture aspects of its creator's internal knowledge organisation

and associations it cannot be said to "externalise" an internal semantic network which would otherwise be implemented in the neural architecture. Similarly, a cognitive process that would otherwise be performed internally cannot be "externalised" and performed using external representations, though it may be replaced by a different cognitive process that uses external representations. Norman (1991) points out that external representations improve performance (when one considers the person and supporting artifacts as a whole) but they do this by changing the nature of the task being done by the person rather than by "amplification". External representations are not used only to represent ideas in a notation intermediate between semantic memory and fully formed prose, but to *mediate* cognition. In this paper I use the term *mediating representation* (Wood, 1992) to emphasise the dialectical relationship between person and representation through which the character, structure and functionality of cognition is radically altered. Similarly, the term *collaborative mediation* (Sibun and Shrager, 1992) is used to refer to the process by which collaborators support one another's activity. In sections 5 to 9 I deliberately use the terms of cognitive psychology and cognitive science in "scare quotes" to refer to the "cognitive faculties" of the complex cognitive system formed by a pair of individuals and their attendant cognitive artifacts. Here, I use these terms in order to emphasise how *different* the "cognitive faculties" of the complex cognitive system are from those of the unaided individual.

Flower (1989) has herself recently argued that a more situated model of writing is called for, which recognises the dialectic between cognition and context. In the next section I outline the central features of the cultural psychological approaches to cognition which will inform such a model.

3. Cultural psychological approaches to cognition

In the last few years a nexus of related approaches which are predominantly anthropological have been gaining currency in cognitive science. In her work on "situated action" and "situated interaction" Suchman (1987) draws on ethnomethodological insights to show that we do not simply plan a course of action internally and then act on the world, but rather our behaviour is structured in interaction with the moment-by-moment contingencies in the situation of action. Lave (1988) similarly emphasises the dialectical relationship between the mind and the setting and activity in which it is engaged, whereby mind and world shape one another in a subtle interaction. "Distributed cognition" (Hutchins, 1991) considers the individuals and their attendant artifacts, which interact in the performance of a task, as a single *complex cognitive system,* and attempts to describe how the different components of such a system are coordinated when individuals work together with shared representational media and shared cultural knowledge. Rather than attempting to induce hypothetical internal cognitive constructs distributed cognition focuses on the observable external representations which comprise the "mental state" of the system. Fuller accounts of these and other approaches to socially shared cognition may be found in a number of current books (see e.g. Resnick, Levine and Teasley, 1991). The cultural-cognitive ergonomics I propose here draws most heavily on Hutchins' notion of distributed cognition (it could probably equally well be called "distributed cognitive ergonomics" or "socio-cognitive ergonomics"), and interprets it within a cultural-psychological framework.

Though these approaches differ significantly in emphasis and in their language they are commensurate with a core of cultural psychological ideas which I abstract below. Cole and Engeström (1989) show that a cultural psychological approach was propounded by Wilhelm Wundt (the "father" of scientific psychology), alongside his experimental psychology, to allow for the study of "higher mental functions", and likewise by Hugh Münsterberg ("the father of applied psychology"), who made an early argument for the distributed nature of mind in 1914. Among Western social scientists the followers of Mead and Dewey have used such ideas to a limited extent (Damon, 1991; Cole and Engeström, 1989; Resnick, 1991) but it is the sociohistorical school of psychology developed in the former Soviet Union by Vygotsky (1978), Leont'ev and Luria which embodies the most complete and cohesive psychology which integrates the cognitive and the cultural. This has recently become more influential in the West, and it is from this formulation that the characterisation of approaches to socially shared cognition that I will present here draws most heavily. In particular, I here suggest that some notion of "activity" as the appropriate unit for scientific study, and a concept of "mediation" through culture, are important in these approaches to socially shared cognition.

Whether one argues that the classical cognitive approach is ill conceived and that we should rather study situated action (e.g. Suchman, 1987), or one suggests that the science of internal individual cognition might be extended to cover the part that the situation plays in cognitive activity (e.g. Hutchins, 1991), may be more a question of terminology than a very substantive difference. In either case the result is an approach concerned with "activity" rather than individual cognition. Kuutti (1991) gives an excellent account of the concept of activity in relation to CSCW research when he presents it as an intermediate unit for scientific research between the individual (where context is ignored) and the social system (which is generally too large and confused to be used as a context in ergonomic research). Activity is "a minimal meaningful context for individual actions" (Ibid, p254), in other words it includes consideration of just those co-participants and mediating artifacts which play a part in the cognitive process at hand. It is important to appreciate that an activity does not exist solely in the moment but is historically grounded in culture through the mediating artifacts which may have been created by preceding generations.

Mediating artifacts, or products of culture, may be considered "tools" in the wide sense that they alter human's relationship with the world and one another. Hammers and slide-rules are most obviously products of culture which condense cultural knowledge and enable people to shape the world, and in turn alter the character of cognitive activity. Cognitive artifacts, which may be as simple as a knot in a handkerchief, or as complex as a computer or encyclopaedia, are also products of culture which allow people to control their own and others' attention and memory. The full import of the cultural psychological perspective becomes apparent when one considers that shared cultural schemas which guide collaborative activity and conversation, and even language itself, are all mediating artifacts. When one does mental arithmetic or uses syllogistic reasoning in the head, one is using forms of thought appropriated in the cultural milieu which are themselves tools, not simply one's own innate unaided "animal" intelligence.

Our relation with the world and others is also mediated through our collaborators in activity. We learn skills by sharing a task with a parent, teacher or expert who structures the activity for us and makes manageable parts of it available to us while performing other parts of the task for us. As we learn, the teacher progressively performs less of the task herself, makes more of the task available to us, and guides us less, until we eventually can perform independently. In this way an individual's cognitive skills are grounded in collaborative activity and are essentially social (Wertsch and Addison Stone, 1985). Just as a teacher can provide this kind of "scaffolding" (Bruner, 1983) for a learner, when adults work together they can mutually support one another through a process of "collaborative mediation" (Sibun and Shrager, 1992).

The cultural-cognitive ergonomics that I am proposing is concerned with cognition as it is mediated through cultural artifacts and collaborators. In the remainder of this paper I will illustrate this transformed cognition using examples from a study I recently performed of participants working closely together using external representations in an idea sketching task. Working within the theoretical perspectives I outline above we are led to study the activities of groups in the context of mediating artifacts, as opposed to attempting to isolate individual cognition in a "pure" or "disembodied" form. Ideally, an activity should be studied in situ rather than in the laboratory as one cannot know in advance what aspects of context will play a part in cognitive activity, and these approaches tend to use methodologies from anthropology, ethnography and conversation analysis, rather than those of experimental and cognitive psychology. Due to limitations of time and money in the study I report here participants were engaged in an activity which was realistic, but not real. Where groups are studied there is no need for verbal protocols. Here, the utterances individuals made for one another, their interpretations of one another's utterances, and the representations they used, were naturally available to the researcher. Their activity was recorded on video-tape and later transcribed and analysed in fine detail.

4. A study of authors working closely together

Writing consists of different tasks (planning, editing, etc) and these are executed in large groups and pairs, synchronously and asynchronously, with varying degrees of co-presence. The idea generating and planning aspects of collaborative writing are often done in close collaboration, and such a task best illustrates the cultural-cognitive approach. In this study, six pairs of (mostly doctoral level) students were asked to generate and organise the ideas for a short paper. The participants were told they need not

actually write the paper, but should stop when they had a clear idea what it would look like if they did write it. By analogy with "conceptual design" (Tang, 1989) this was a "conceptual authoring" task in that it was concerned with the predominantly early activity concerning the ideas for the text rather than their instantiation in fully formed grammatical prose. The authors sat side by side at a large desk, and were videoed from above (to show their drawing, writing and pointing, see figure 1) and from the front (to show their gaze, bodily orientation and gestures).

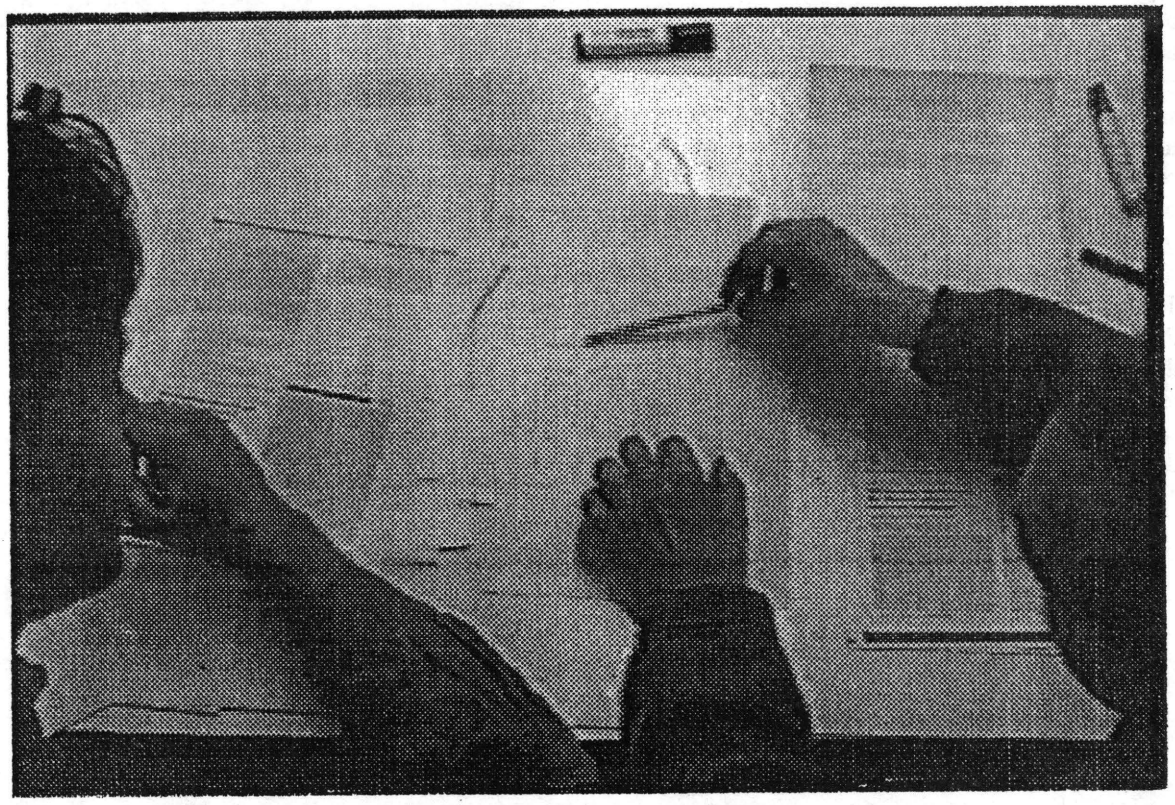


Figure 1. The view from above shows the authors and their workspace activity.

The authors were provided with a variety of external media (a horizontal whiteboard, different sized pens and paper, post-it notes, etc) and were allowed to bring their own media and literature to refer to. Though the situation was not fully natural the authors generally chose topics of genuine interest to them and were drawn into the task. They reported that they felt they were behaving naturally. The experimental paradigm is similar to studies of use of shared drawing surfaces by designers (e.g. Tang, 1989; Scrivener and Palmen, 1991) though here the authors were concerned with concepts rather than physical artifacts so that they used graphical and spatial notations only in metaphorical ways. The authors generally worked closely together over shared sheets of paper or the whiteboard, discussing ideas and representing these ideas and their structure in the shared workspace. In some pairs one author would take the role of "recorder" for long periods, but more usually both authors would take turns at drawing marks[1], and these turns were closely synchronised with turns in the spoken dialogue. Each session ran for approximately one hour. In a structured post-interview the participants were asked about their experiences of the exercise and about their use of mediating representations in their everyday work. They were also played parts of the video tapes where the activity was particularly interesting or ambiguous and were asked to comment on what they or their partner were doing. These interviews were themselves recorded.

5. Mediating representations ("Knowledge representation")

The *mediating representations* produced in the workspace mainly consisted of single words and short phrases which chunked and labelled complex ideas discussed at, or just prior to, the time they were

[1] In this paper I use the word "draw" generically to refer to the production of any mark in the shared workspace. The word "mark" is similarly used to mean any textual, typographical or non-textual mark.

produced (see figure 2). I will refer to these words and short phrases as "labels". These verbal labels were invariably augmented with the typographical marks which are well established in our culture. Case, size, and underlining, were often used to indicate something like importance or emphasis. Bulleting and numbering were used to enumerate lists of ideas. Indentation was used to indicate the hierarchical organisation of ideas. These typographical cues added a perceptible access structure and the subjects reported that they made the representations more visual and memorable. The participants also sometimes used informal non-textual graphical and spatial notations. Related ideas were linked with a line or placed adjacent to one another. The spatial positioning of a label in the two dimensions of the paper relative to other labels often showed in what phase of the work it had been produced, its place in an argumentative structure, or its projected place in the final text. Ideas were grouped by being clustered or enclosed together in a rectangle or circle. These words, typographical cues and graphical signs originate in the conventional languages of English, typography and graphic design, established and experienced daily in our culture. However, the meanings of these mediating representations would be highly ambiguous or opaque to someone who had access only to the mediating representations, as single words at a high level of abstraction were used to represent complex ideas and graphical signs were used in idiosyncratic ways. Their meaning was generally not explicitly negotiated but rather was established by the drawer through the speech uttered concurrently with the drawing action. Therefore, the signs were "informal" in the sense that knowledge of the context of the sign's production was necessary for its interpretation. In this way, the representation of knowledge in the complex cognitive system was distributed between the physical signs (as they would be interpreted in the cultural context of English, typography and graphical languages) and the minds of the participants who collaborated in the sign's production.

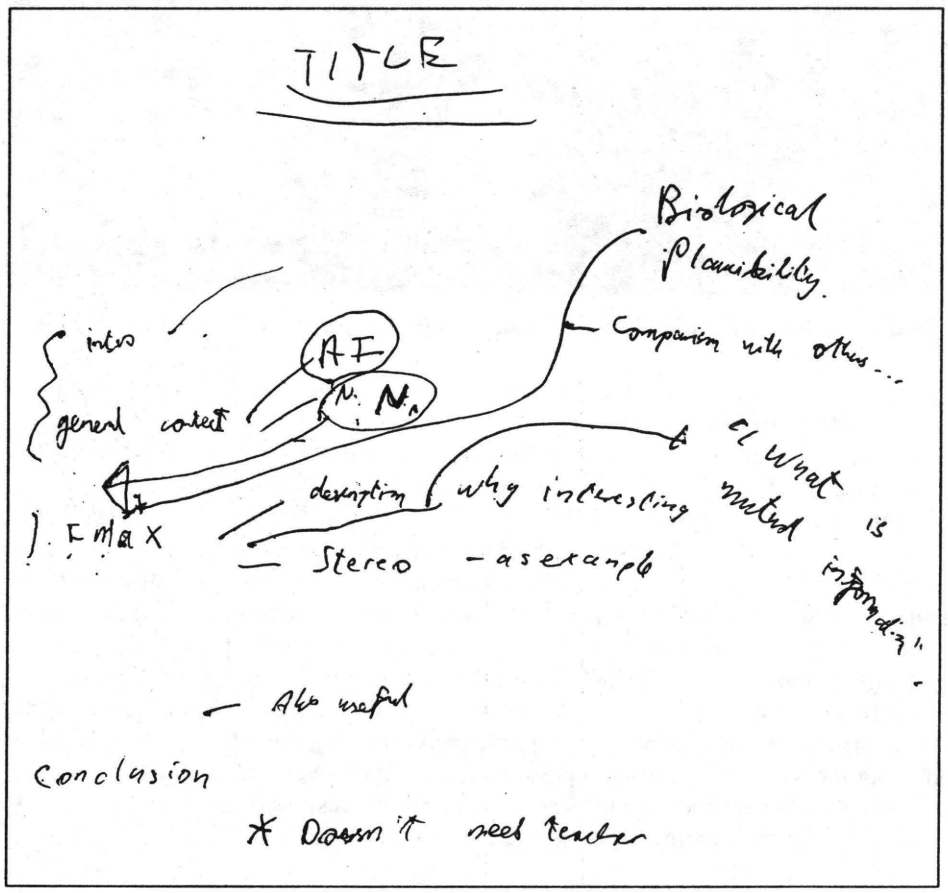


Figure 2. An idea sketch incorporating verbal idea labels, typographical cues and graphical signs.

In addition to these informal graphical languages some graphical marks were more pictorial. For example, one author drew a large funnel shape over the whole of a sheet of paper to represent the structure of a paper (a general introduction followed by increasingly tighter focus on a particular topic). Later both authors pointed to places in this shape, and drew on it when talking about parts of the paper, but no text

was ever added to the representation. Even this very unusual non-textual representation had its meaning almost entirely established implicitly *in situ*, without explicit negotiation, through its context in the pair's activities, and the authors reported that the representations they used seemed "natural". I would suggest that the representations used were strongly grounded in culture and not totally idiosyncratic. In Western culture it is natural to represent the beginning of a paper by the top of a page and the end of a paper by the bottom of the page. It is natural also to represent a broad general introduction by the wide part of a funnel, and the narrow focus of the central part of the paper by the narrow part of the funnel.

6. Modes of communication ("Information flow")

Just as representations "in the head" and in the workspace complemented one another in carrying meaning for the complex cognitive system, so speech, gesture (including eye movement and body orientation) and drawing were used closely in concert with one another. In the situation studied communicational events can take place in several modes (see figure 3). First, interlocutors may simply speak to one another (1). They may accompany speech with gesture, or occasionally use gesture alone, including eye movement (perhaps for "back channel communications" like the "um hmm" sounds that a listener produces to indicate that she is "with" the speaker) (2). Speech and gesture may make reference to the shared representations in such a way that the shared representations are required to be present for the communication to carry its meaning (3). In their synchronous conversation the interlocutors may communicate through the shared workspace, making changes to the representations in the workspace, usually accompanied by speech (4). Asynchronous intentional communication through the shared workspace, not accompanied by gesture or speech, may also take place where marks are made in the workspace by one participant who intends that the other will later read them (5). When one of the participants uses an area of the workspace for their own use this may be accessed by the other and "unintentional communication" may take place (6). Mediating representations may be used for "reflexive communication" (communication with the self) where marks in the shared workspace are both written and accessed by the same author (7). In practice it is hard to observe unintentional communication (6) or reflexive communication (7) (though it is assumed that reflexive communication goes on all the time in this situation). Asynchronous communication (5) did not seem to be common in the situation studied. These communicational events take place in the context of shared knowledge (including knowledge generated in the history of the collaboration), represented in the shared workspace, in the minds of the collaborating participants, and embodied in the culture that surrounds the system.

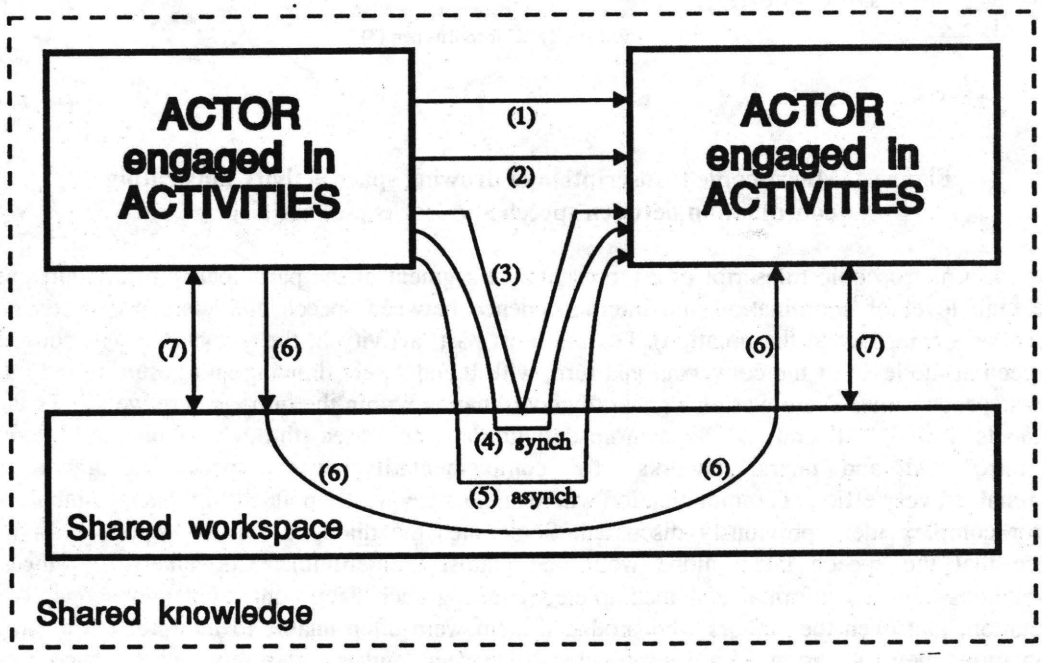


Figure 3. Modes of communication with and through the workspace.

These modes are perhaps in some ways rather arbitrary points on a continuum as it was often hard to discriminate them. For example, sometimes a "fossilised gestural representation" is produced which augments speech in the way that gesture does but leaves a mark which serves as a cue to memory for the gesture and so for the ideas expressed. Conversely, "ephemeral drawing actions" (gestures which seem to "draw" with an invisible pencil) may be made on an "airboard" (Olson, 1990) overlaid above the work surface, so that no physical mark is left behind. Also, these modes of communication can be further sub-categorised. For example, synchronous communication through the workspace (4) includes both (a) fossilised gestural representations and (b) proper meaningful symbols often accompanied by more redundant or incidental speech. However, these categories do give a conception of the space of possible modes of communication that could take place in this situation.

J: [a]What does **that** say?[b] (.)

P: [c]general [d]sort of context[e] and [f](.) sort of AI (...) [g](.....)AI [h]a:nd neural networks[i] (...) ummmm [j]description of(...) [k]its just [l]w>here where< to [m]() talk about stereo >cos thats< I mean cos obviously you might write an article in itself about [n]stereo (.) I mean () [o]maybe [p]vision (put in pap[q]er) (.5)

J: [r]**yeah** >[s]I think ye:ah >cos a think< [t]this [u]thishh (.) should happen (...) [v]as **fast as possible** cos theres >**quite** a lot of ground to cover going through the [w]**intro** and the [x]**AI** and into [y]neural netw⌈orks

P: ⌈Yea

a: J moves pen to active region, leans in (2)
b: J touches "description" with penpoint (3)
c: P moves penpoint to "description" (3)
d: J removes penpoint from active region (3)
e: P lifts penpoint from "description" (3)
f: P gesticulates over active region, then withdraws (2)
g: P draws line, "AI" (4)
h: P draws line, "NN" (4)
i: P withdraws, makes tiny point or unseen alteration and withdraws again (3)
j: P points at "description" with pen (3)
k: P gestures over "description" (3)
l: P gestures over article (3)
m: P gesticulates, in air, orienting to J (2)
n: P circles pen in air (2)
o: P touches penpoint to "description" (3)
p: P draws "stereo" (4)
q: P withdraws (3)
r: J moves pen into active region (3)
s: J taps "AI" (3)
t: J taps AI and description (3)
u: J circles around "intro", "general context" and "AI" (3)
v: J meanders around "intro", "general context", "AI" (3)
w: J touches "intro"with pen (3)
x: J taps "AI" then raises pen (3)
y: J taps "NN" then lifts pen (3)

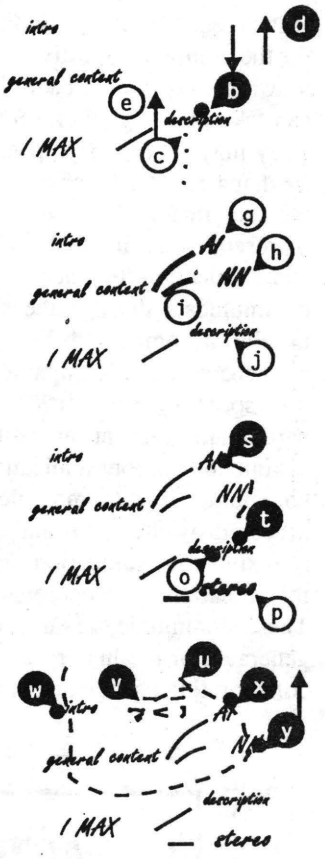


Figure 4. Microscopic transcription of drawing space activity illustrating coordination between speech and workspace activity.

In figure 4, a microscopic transcript of a representative segment of the participants' behaviour, one can see the high level of coordination and interdependence between speech and workspace activity (see appendix for a brief key to the notation). For the most part, activity in the workspace was coordinated with speech at the level of the conversational turn, with P and J here drawing and gesturing only during their own speech turns. There was also generally coordination within the turn, as here we see J's tapping of the labels "intro", "AI" and "NN" synchronised with the emphasised (through volume and intonation) words "intro", "AI" and "neural networks". This complementarity between speech and drawing space activity enabled very efficient communication where authors frequently pointed to terse idea labels which stood for complex ideas previously discussed. Sometimes the dialogue was so replete with deictic reference that the speech taken alone would be almost unintelligible. Conversely, the mediating representations were so informal and incomplete, requiring such deep contextual knowledge for their interpretation, that even the authors who produced them were often unable to interpret the meaning of representations they had produced a week previously. (Indeed, authors often reported that such "back of an envelope" representations are discarded almost immediately after they have been produced.) It is

therefore evident that the communication between the participants was profoundly affected by the presence of the shared drawing space.

A methodology within distributed cognition, "STORM analysis" (Systematic Transformation Of Representational Media) (Flor, 1991), considers the trajectories and transformations of representations in the wider cognitive system. In writing it is possible to follow long "idea careers" (Tang, 1989) from ideas' first generation and utterance to their final instantiation in well formed prose (Plowman, 1992). In the study, the authors frequently spoke an ill-formed idea which passed between them in discussion, being developed and transformed at each turn, before being represented in the shared workspace. Once represented the label served as a cue to memory and often was referred to and pointed at, taken up again and further transformed.

7. "Memory" in the complex cognitive system

The drawing activity fell into two fairly distinct classes where marks were produced either to store ideas, or in the process of communicating them. Ideas were stored by being reified as labels and recorded in the workspace as they were discussed and agreed or immediately thereafter. Mediating representations were produced in conversation when marks were produced, together with speech, which helped to communicate substantive content. Conversational use generally went together with gesturing and pointing at the representations. The representations produced included crossings out, doodles and other marks which were a function of the process of discussing and exploring ideas. It was reported that the often seemingly incidental marks which recorded the history of the work provided contextualising information which allowed more of what was meant by a representation to be reconstructed when it was read. For example, the authors would often construct an ordered list (e.g. of headings) and would later insert an item by drawing a label for the item and an arrow from the item to a location in the list (see figure 2). Later they might scribble out the arrow and draw an arrow to a different location. This resulted in an admittedly messy, but also visually structured and memorable representation which made evident not only what had been arrived at but also what might have been, and the order in which the possibilities had been considered. The participants could easily be mutually aware of these representations, and so the shared workspace could be considered as part of the "working memory" of the complex cognitive system.

When a single individual uses mediating representations the individual's cognitive faculties are mediated through the representations and the "working memory" and "cognitive processes" that can be performed by the complex cognitive system differ radically from those of the individual. In addition to this, where mediating representations are shared they also play a part in the coordination of the participants' activity. It has been shown that interlocutors establish and maintain a high level of intersubjectivity through a process of "grounding" whereby a common ground of shared beliefs and assumptions is constructed and constantly maintained (Clark and Brennan, 1991). The common knowledge and culture of the authors (the pairs knew one another well and worked in the same academic field) permitted very close coordination, and in the extreme case sometimes resulted in jointly produced utterances (where one author completed the other's utterance). In this study mediating representations were profoundly important in grounding communication as the results and history of the participants' preceding work were continuously accessible to them in the workspace. The complex cognitive system was therefore able to maintain a cohesive "perception" or interpretation of the domain under discussion, partially reified in the external mediating representations.

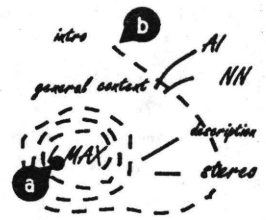
Flor (1991) argues that the shared cultural schemas and organisation of a complex cognitive system structure the behaviour of the system and so constitute a "procedural memory". In this study, for example, we can see how shared knowledge of the rules governing interaction in discourse in our culture allowed the workspace to be used in the negotiation of conversational turn taking. The participants rarely gestured or drew in the workspace outside their conversational turn, and they sometimes vacated the active region of the workspace to indicate the end of a turn. An author would sometimes gesticulate in the active region of the workspace to signal a wish to speak. The workspace therefore had the status of "the floor" in a debating chamber.

8. "Attention" in the complex cognitive system

The external, spatial nature of the mediating representations allowed the authors to maintain "referential identity" (Clark and Brennan, 1991), whereby they mutually understood one another to be talking about the same thing. When they were referring to ideas represented in an area of the shared workspace authors oriented together to that area, gesturing and drawing in it, and looking to one another occasionally to ascertain where their interlocutor's attention was focussed. These deictic references were tightly synchronised with speech. The addressee would similarly orient to, look at, and point at, those representations, confirming that she had identified the referent. Thus, speech, body movement, gesture, and eye movement were used to maintain a shared focus of attention. At the beginning of the transcript in figure four (events b, c, d, e) there is a good example of simultaneous pointing at the label "description". Throughout figure 5 there is only one small drawing event, but there is constant gestural activity accompanying speech and it is clear that this is not simply redundant handwaving. It is important to emphasise that one can control one's own attention through pointing and gaze as well as the attention of one's partner. Without mediating one's innate cognitive abilities through external cultural tools one's thoughts are likely to drift into daydream. Sharing one's thoughts interactively with another person can also serve to maintain a stable focus of attention. In event f of figure 5 J deliberately leaves a pen pointing at the representation that he and his partner were considering, allowing their joint focus of attention to be controlled through the mediating artifacts.

J: and then try to spend [a]the **bulk** of the article [b]on on this stereo and Imax

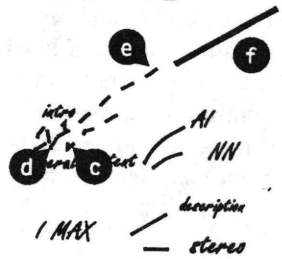
a: J puts left hand on bottom of sheet, and pen on "I MAX" (3)
b: J circles "I MAX" three times with penpoint then moves penpoint up to "AI" (3)



P: []right

J: [][c]so we try and get rather() [d]even though if we try and go through it quickly [e]its going to [f]take (...) you know **two** or three full **pa**ragraphs=

c: J meanders up to "intro" and "general context" (3)
d: J wiggles between "intro" and "general context" (3)
e: J drags pen up to top right (3)
f: J leaves pen pointing in direction of "AI", "general context" and "intro", withdraws, leans back and looks to P (3+2)



P: [g]or we could maybe just put this into [h]one (.) one thing like [i]the **in**tro will be[j] (.) uu:m [k]() unveiled blah blah blah this thing which doe:s: [l]>this wonderful thing< [m]and then quickly into (.)[n]y'know why this is the sort of thing people have been trying to do in AI for a long time [o]and=

g: P moves pen to "intro" and "general context" (3)
h:P draws vertical wiggly line starting to left of "intro" and ending to left of "context" (4)
i: P moves pen to above "AI" (3)
j: P lifts pen (3)
k: P handwaves above active area, then gestures independently of drawings (2)
l: P handwaves over article (3)
m: P positions pen around "NN" and wiggles left and right (3)
n: P makes many handwaving gestures above active region (3)
o: P positions pen over "NN" (3)

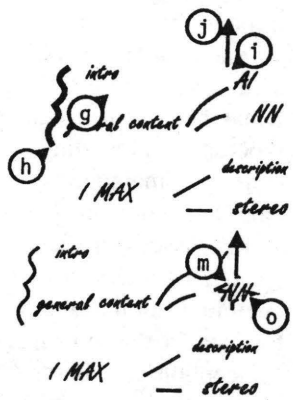


Figure 5. Microscopic transcription of drawing space activity illustrating the role of the workspace in the mediation of attention.

9. Thinking with mediating representations

Authors commonly reported that the workspace was important in supporting their thinking. Many terse labels (which sometimes stood for several paragraphs of text) could be placed in relation to one another,

allowing a gestalt perception of structure. The overview that a graphical representation gave them helped the participants to achieve a wholistic conception of the content they were considering. Reasoning was most obviously distributed between the participants and the external representations where, for example, blank spaces in a table of arguments for and against different positions reminded the participants to consider arguments for and against each point.

It emerged as important that the workspace allowed the fluent externalisation of ideas. The authors were able instantly to choose their "font" (e.g. level of finishedness, style of handwriting), size, the angle and place at which a label should be drawn, the weight of the line, the shape of arrowhead, whether they would produce text or graphical marks, etc, without having to pull down menus or select tools. (One author commented in interview that in a conventional drawing program one might draw a line from an idea label to a place where one wanted to create another idea label, and forget what the second idea was during the operation.) This ease of drawing was accompanied by an ease of reading as a result of the visual structure of the mediating representations which made them more memorable and provided cues for access and navigation (e.g. lines between ideas, typographical cues, spatial layout) so that authors could locate ideas and their relations with ease.

The representations' high level of abstraction and potential to be interpreted in different ways (which I term *semantic potential*) meant that the meaning of a label could be allowed to change as a result of further thought and discussion without it being necessary to edit the external physical mark. Though the representations were reported not to be reliable, as well formed prose is, for storing meaning over time this very reliance on the context particular to the representation that the reader brings for interpretation meant that the marks had a low "viscosity" (resistance to editing and restructuring) (Green, 1989) even though physical ink marks may be difficult to alter or delete. This allowed nascent ideas to be considered without "premature commitment" (Green, 1989) to an inappropriate conceptualisation or structure.

These efficiencies allowed the participants a speed of thought and communication both with their interlocutor and with themselves. Through producing marks which denoted their ideas in an idea sketch they were better able to perceive, clarify and structure their own thoughts. The workspace activity fulfilled a function very similar to the function that sketching plays in fine art (e.g. Fish and Scrivener, 1990) where the character of the artist's imagination and visualisation is changed through a mind-media dialectic where her perceptual faculties can be applied to her own ideas through the external representation.

10. Conclusion: towards a cultural-cognitive ergonomics

The perspective that overwhelmingly emerges from the study considers that the authors' cognitive activities were profoundly mediated through the mediating representations and through collaborative mediation. Bound together by shared cognitive artifacts (language, gesture and mediating representations) with their genesis in human culture the individuals and their associated tools formed a complex cognitive system with quite different structure and functionality than is supposed by internal cognitive models. Norman (1991) criticises the fact that psychology has concentrated almost exclusively on cognitive faculties such as memory, attention, perception, and thought, unaided by cognitive artifacts. Similarly, cognitive science has little to say about how individuals support one another's cognitive activity. Throughout this paper I have used the terms of cognitive psychology and cognitive science, such as "knowledge representation", "information flow", "working memory", "procedural memory", "perception", "attention" and "thinking" to refer to aspects of the cognitive activity which is distributed throughout the complex cognitive system. It cannot be too strongly emphasised however that these "cognitive faculties" are not simply naive externalisations of their internal counterparts but rather are radically different phenomena and must be studied as such.

The study focussed on synchronous generation and organisation of ideas, as a component of the collaborative writing task which was most predisposed to this cultural-cognitive approach. A model for collaborative writing (and for other shared and individual conceptual activities) of a very different character than the "classical" cognitive models of the writing process, is beginning to emerge. Instead of focussing on the internal architecture of cognitive processes in the individual cogniser this cultural-cognitive approach conceives of mutually supporting individuals and their cognitive artifacts more

holistically. In such a model words like "generate" and "organise" are used to refer to writing *activities,* rather than internal cognitive *processes* as they are in the Hayes and Flower model. The focus is on the mediating representations and other cognitive artifacts the authors use, their closely synchronised modes of communication, and the shared knowledge and intersubjectivity which permeates and binds the complex cognitive system.

I have suggested that some activities may be best studied and considered in the context of collaborative work as natural verbal protocols are produced and the interlocutor's interpretation of those protocols can be used as a clue to their meaning by the researcher. But this cultural-cognitive ergonomics informs the design of systems to support the activities of individuals as well as the activities of groups, as when an individual works alone her cognition is mediated through artifacts which have their genesis in culture. Important observations in the study reported here, which echo Tang's (1989) findings and those of the Colab project (Bobrow, et al, 1990), were that gestural and drawing activity in the shared workspace are closely coordinated with speech, and a fluent mix between drawing non-textual marks, textual marks and gesturing is used in collaborative work. However, participants also reported that they commonly used pencil and paper rather than their workstations when working alone to plan a text because of the flexibility and immediacy afforded by conventional media. I would suggest that the properties required of drawing spaces both in communication in collaborative work and in the support of individual creative work are similar in this respect for similar reasons, as properties which are necessary for fluent communication with others are often also necessary for fluent communication with the self. I have argued that where individuals use external representations there is a dialectical relationship between the individual mind and the representations through which cognition is mediated. Considering the drawing space as forming part of a complex cognitive system we can see that drawing space activity must be coordinated not only with speech (in collaborative work) but more generally with thought itself.

Though some systems that support single person authoring or collaborative ideas organising provide a hypertext-browser-like window for arranging and linking idea nodes (e.g. Sharples and O'Malley, 1988; Smith et al, 1986; Stefik, et al, 1987) these more constraining formal representations seem less expressive than the mediating representations that conventional media such as pen and paper support. It may be that a synchronous object oriented shared drawing space with a video channel which allows one interlocutor to see where another is looking (e.g. the ROCOCO sketch pad of Clark and Scrivener, 1991) would support more appropriate mediating representations for idea generating and organising. Such a drawing space, with an underlying representation which allowed operations to be performed on the mediating representations, might combine the expressiveness of conventional mediating representations with the benefits of computer supported representations (low viscosity, ease of copying and rearranging, etc). Ideally a system might be implemented in Wellner's (1991) DigitalDesk, which aims to seamlessly mix conventional and electronic media by projecting computer generated windows onto the desk-top and recognising and reading paper which is placed on the desktop. In this system one can imagine that actual pencil marks and projected electronic marks (drawn with a stylus) on a restaurant napkin could be linked to nodes in a network, or marked parts of an outline, in sophisticated projected views on the desktop. These ideas are more fully described elsewhere (Wood, 1992). In this paper I have used conceptual authoring as an example activity within which to discuss a cultural-cognitive approach, but other writing activities (such as editing, which is not generally done in close collaboration) may not have the same requirements.

Though we may sometimes be able to change internal cognitive processes through training we have more control over the design of cultural artifacts. An understanding of internal cognitive processes can inform system design through its implications for the external, but studying the activities of the complex cognitive system formed by individuals or groups and their attendant artifacts may tell us more about the part that cultural artifacts play in supporting thought and communication. Such an approach will tend to use observation of activity *in situ,* or at least in naturalistic settings, together with interviewing, rather than more formal psychological experimenting. Especially where a system being designed will replace some existing system which is already embedded in its user's activity, context and culture, a cultural-cognitive ergonomics which draws on the shared cognitive approaches outlined above may be more informing of the design of that system than more cognitive approaches.

Acknowledgements:

I especially thank Yvonne Rogers for many useful conversations concerning distributed cognition. I thank Eevi Beck, Lydia Plowman, and Mike Sharples at Sussex University for useful comments on this work. I also thank Allan MacLean of Rank Xerox Cambridge EuroPARC, and Ed Hutchins and his research group for stimulating discussions relating to this work. This work was funded by Science and Engineering Research Council CASE award No. 9055550X in collaboration with Rank Xerox Cambridge EuroPARC.

References:

Bobrow, D. G., Stefik, M., Foster, G., Halasz, F., Lanning, S. and Tatar. D. (1990). The Colab Project Final Report. Xerox PARC technical report SSL-90-45, System Sciences Laboratory, Palo Alto Research Centre, 3333 Coyote Hill Road, Palo Alto, California 94304.

Bruner, J. (1983). Child's Talk. W. W. Norton: New York.

Caccamise, D. J. (1987). Idea Generation in Writing. In: Matsuhashi, A. (Ed.). Writing in Real Time: Modelling Production Processes. Ablex: Norwood, NJ.

Clark, S., and Scrivener, S. A. R. (1991). The ROCOCO Sketch Pad Distributed Shared Drawing Surface. Unpublished paper, LUTCHI Research Centre, Loughborough University, Leicestershire, UK.

Clark, H. H. and Brennan, S. E. (1991). Grounding in Communication. In: Resnick, L. B., Levine, J. M. and Teasley, S. D. (Eds.). Perspectives on Socially Shared Cognition. American Psychological Association.

Cole, M. and Engeström, Y. (In press). A Cultural-Historical Approach to Distributed Cognition. In: Salomon, G. (Ed.). Distributed Cognitions: Psychological and Educational Implications. Cambridge University Press: New York.

Damon, W. (1991). Problems of Direction in Socially Shared Cognition. In Resnick, L. B., Levine, J. M. and Teasley, S. D. (Eds.). Perspectives on Socially Shared Cognition. American Psychological Association.

Fish, J. and Scrivener, S. A. R. (1990). Amplifying the Mind's Eye: Sketching and Visual Cognition. In: Leonardo, Vol. 23, No. 1, pp. 117-126.

Flor, N. V. (1991). Modeling Distributed Cognition in Complex Cognitive Systems: A Case Study of Collaborative Programming During Software Maintenance. Unpublished manuscript. Distributed Cognition Laboratory, Department of Computer Science, D-015, University of California, San Diego, La Jolla, CA 92093-0515.

Flower, L. (1989). Cognition, Context and Theory Building. Occasional Paper No. 11, Center for the Study of Writing, University of California, Berkeley, CA 94720 and Carnegie Mellon University, Pittsburgh, PA 15213.

Green, T. R. G. (1989). Cognitive Dimensions of Notation. In: Sutcliffe, A. and Macaulay, L. (Eds.). People and Computers V. Cambridge University Press: Cambridge, UK.

Hayes, J. R. and Flower, L. S. (1980). Identifying the Organisation of Writing Processes. In Gregg, L. W. and Steinberg, E. R. (Eds.). Cognitive Processes in Writing, pp. 3-30. Lawrence Erlbaum Associates: Hillsdale, New Jersey.

Holt, P. (1989). Models of Writing: A Question of Interaction. In Williams, N., (Ed.). Computers and Writing: Models and Tools. Intellect: Oxford, UK.

Hutchins, E. (1991). Individual and Socially Distributed Cognition. Cognitive Science 234 Course Notes, Cognitive Science Department, University of California, San Diego, La Jolla, CA 92093.

Kuutti, K. (1991). The Concept of Activity as a Basic Unit of Analysis for CSCW Research. In Bannon, L., Robinson, M., and Schmidt, K., (Eds.). Proceedings of the Second European Conference on Computer-Supported Cooperative Work. Kluwer Academic Publishers: Dordrecht.

Lave, J. (1988). Cognition in practice. Cambridge University Press: Cambridge, UK.

Leland, M. D. P., Fish, R. S. and Kraut, R. E. (1988). Collaborative document production using Quilt. In: Proceedings Of The Conference on Computer-Supported Co-operative Work, pp.206-215. ACM SIGCHI and SIGOIS.

Neuwirth, C. M., Kaufer, D. S., Chandhok, R., and Morris, J. H. (1990). Issues in the Design of Computer Support For Co-authoring and Commenting. In: Proceedings of the Conference on Computer Supported Cooperative Work. ACM SIGCHI and SIGOIS.

Norman, D. A. (1991). Cognitive Artifacts. In: Carroll, J. M. (Ed.). Designing Interaction: Psychology at the Human-Computer Interface. Cambridge University Press: New York.

Olson, G. M., 1990. Collaborative Work as Distributed Cognition. Unpublished manuscript. Cognitive Science and Machine Intelligence Laboratory, The University of Michigan, 701 Tappan Street, Ann Arbor, MI 48105-1234.

Plowman, L. (1992). Tracing the Evolution of a Co-Authored Text. Unpublished manuscript. School of Cognitive and Computing Sciences, Sussex University, Falmer, Brighton, BN1 9QN.

Resnick, L. B. (1991). Shared Cognition: Thinking as Social Practice. In: Resnick, L. B., Levine, J. M. and Teasley, S. D. (Eds.). Perspectives on Socially Shared Cognition. American Psychological Association.

Resnick, L. B., Levine, J. M. and Teasley, S. D. (1991). Perspectives on Socially Shared Cognition. American Psychological Association.

Scrivener, S. A. R., and Palmen, H. (1991). An Analysis of Face-to-face Drawing Activity. In: Proceedings of DATER'91. (Loughborough, UK, September 1991).

Sharples, M. and O'Malley, C. (1988). A Framework for the Design of a Writer's Assistant. In: Self, J. (Ed.). Artificial Intelligence and Human Learning: Intelligent Computer-Aided Instruction. Chapman and Hall: London, UK.

Sharples, M. and Pemberton, L. (1988). Representing Writing: An Account of the Writing Process with Regard to the Writer's External Representations. Cognitive Science Research Paper CSRP 119. School of Cognitive and Computing Sciences, Sussex University, Falmer, Brighton, BN1 9QH.

ShrEdit. (1990). ShrEdit 1.1: A shared editor for the Apple Macintosh. Cognitive Science and Machine Intelligence Laboratory, University of Michigan, 701 Tappan Street, Ann Arbor, MI 48105-1234.

Sibun, P. and Shrager, J. (1992). Collaborative Mediation of the Setting of Activity. In: Proceedings of the Annual Conference of the Cognitive Science Society, Bloomington, IN. Lawrence Erlbaum Associates, Hillside, NJ.

Smith, J. B., Weiss, S. F., Ferguson, G. J., Bolter, J. D., Lansman, M. and Beard, D. A. (1986). WE: A Writing Environment for Professionals. Textlab Report TR86-025, Department of Computer Science, University of North Carolina, Chapel Hill, NC.

Smith, J. B. and Lansman, M. (1989). A Cognitive Basis for a Computer Writing Environment. In: Britton, B. K. and Glynn, S. M. (Eds.). Computer Writing Environments: Theory, Research and Design. Lawrence Erlbaum Associates: Hillside, NJ.

Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. (1987). Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings. In: Communications of the ACM, Vol. 30, No. 1, pp. 32-47.

Suchman, L. (1987). Plans and Situated Actions: The Problem of Human-Machine Communication. Cambridge University Press: Cambridge, UK.

Tang, J. C. (1989). Listing, Drawing and Gesturing in Design: A Study of the Use of Shared Workspaces by Design Teams. Xerox PARC Technical Report SSL-89-3, System Sciences Laboratory, Palo Alto Research Centre, 3333 Coyote Hill Road, Palo Alto, California 94304.

Vygotsky, L. S. (Cole, M., John-Steiner, V., Scribner, S. and Souberman, E. Eds.). (1978). Mind in Society. Harvard University Press: Cambridge, MA.

Wellner, P. (1991). The DigitalDesk Calculator: Tactile Manipulation on a Desk Top Display. In: Proceedings of ACM Symposium on User Interface Software and Technology (UIST'91), Nov. 11-13, 1991, pp. 27-33.

Wertsch, J. V. and Addison Stone, C. (1985). The Concept of Internalisation in Vygotsky's Account of the Genesis of Higher Mental Functions. In: Wertsch, J. V. (Ed.). Culture, Communication and Cognition: Vygotskian Perspectives. Cambridge University Press: Cambridge, UK.

Wood, C. C. (1992). A Study of the Graphical Mediating Representations Used by Collaborating Authors. Cognitive Science Research Paper CSRP 230, School of Cognitive and Computing Sciences, The University of Sussex, Falmer, Brighton, BN1 9QH.

Appendix: Key to microscopic transcriptions:

The microscopic transcription scheme used in this paper uses three columns. The leftmost column is a detailed transcript of the participant's speech, mostly using the conventions of Conversation Analysis. The rightmost column records the participant's gesturing and drawing activity in the shared workspace using a choreographic notation. The central column is used to relate the other two columns. It is a list of gesturing and drawing events, enumerated with letters of the alphabet. (The bracketed numbers in this column show the predominant mode of communication in which the event takes place - see section 6.) These enumerating letters appear in the leftmost (speech) column, usually together with an underline which shows where, in relation to the speech, the event occurs. These letters also appear in the rightmost column in a circle at the location in the workspace at which the event takes place.

Speech:		Drawing and Gesture:	
bold	emphasis (volume or intonation)		pointer, letter indicating event, colour indicating participant
>	speech speeds up		pen lifted up
<	speech slows down		
(. . .)	pause (1 point = 0.1 seconds)		pen touched down
(.5)	0.5 second pause	. . . . .	gestural movement near workspace
:	preceding sound lengthened		
[	start of overlapping speech	_ _ _	gestural movement touching workspace
[]	simultaneous onset of speech		
=	other participant interrupts	●	emphasis (tap or jab)

The interest in the notion of shared functional representation between the operators in change of shift phase.

C. Grusenmeyer

Industrial Psychology and Ergonomics department
Systems ergonomics section

INRS, Nancy, France

ABSTRACT

The growing complexity of systems along with the increasing size and diversity of tasks emphasize the problems of the division and coordination of activities between operators. Thus studies on collective work are more and more numerous. However, most of them are focussed on the cooperation between operators with different abilities and a well-defined problem to solve. Seen from this point of view, the change of shift seems to be a specific phase in collective work. Generally, the operators have similar abilities. Their cooperation is not necessarily determined by the fact that they must work together all the time and the pooling of each person's cognitive resources is the real objective of this work phase. Within this framework, the notion of shared functional representation between operators would appear to be a perspective worth developing. This is the point we will try to demonstrate in this paper. We will present two examples of analyses carried out within a study of the change of shift phase. We will see to what extent these analyses show that the lack of shared functional representation between operators can contribute to a non-optimum shift change and which mechanisms can play a part in such a situation.

1. Introduction

Increasing automation in most industrial sectors is accompanied by deep-rooted changes as much at organisational level as in terms of operators' tasks and activities.

This automation is characterised by (Neboit, 1990):

- growing complexity of systems (numerous interactive variables, relatively long response time delays), which assumes certain operational strategies and poses the **problem of the operators' mental representation** : the necessity for the operator to represent the synchronisation of different parametres and all the factors contributing to the evolution of a variable, of planning its activity within important time delays, of estimating time, of making long-term inferences...(Alengry, 1988; Samurçay and Hoc, 1989)
- mediation of information and operations via man-machine interfaces. The thus transformed and abstract information strongly activates the **operators' cognitive functions**: diagnostic and anticipation activities, but also the effort of memorizing imposed by the sequential presentation of this information (De Keyser, 1985); possibly even reconstruction or structuration activities due to a loss of clarity in the process.
- the evolution of tasks at organisational level. Tasks rigorously defined and attributed to a particular job gradually incorporate an increase in the number of these tasks. The operator must work to a global objective and he is given a wider range of less defined tasks. (Chabaud and De Terssac, 1989). **Systems collective management** is, therefore, set up by the operators. Thus, the management of the information flow between operators, the rapidity of its circulation and its reliability are crucial for the system. Hence, the necessity for coordination, mutual understanding between operators, and the pooling of eachother's cognitive resources (Chabaud and De Terssac 1989).

The real world with which the operator interacts is no longer limited to the task, to the interfaces or to the process itself. In fact it is largely made up of **interactions with other operators.** It is without doubt that these interactions can also create, question, modify or consolidate the operator's representations and lead to certain strategies, know-how and a common language.

This last point shows the reasons for more and more numerous studies on collective work.
However, most of them focus on the cooperation between operators with different abilities and of well-defined problems to solve together (Navarro, 1991; Rogalsky, 1989; Chabaud and De Terssac, 1989). Therefore, the operators must coordinate their activities, adapt to eachother and pool their cognitive resources in order to create a common operational system of reference. (Chabaud and De Terssac, 1989) Thus, cooperation consists of coordinating eachother's knowledge, dividing tasks and evaluating the results obtained and the consequences in terms of workload. However, there is always one person specialised in a specific task, upon which the operators previously agreed.

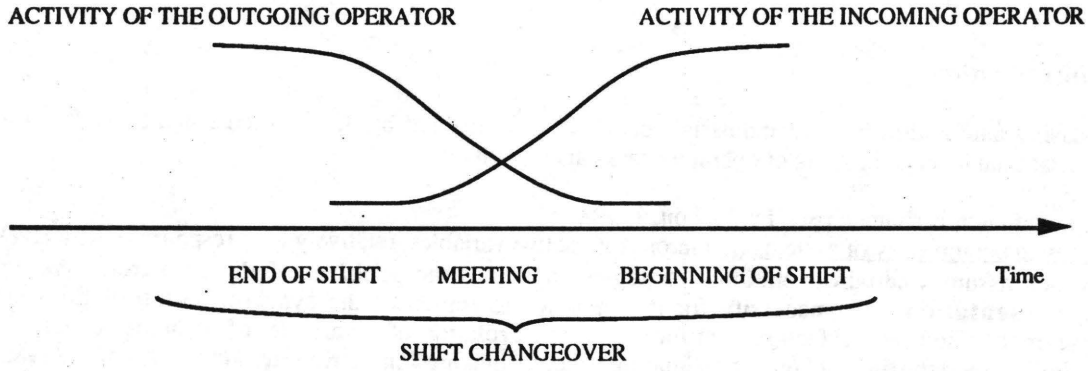
In this light, the shift changeover appears to be a specific phase in collective work.
Indeed, we would consider the shift changeover as a work phase whose aim is to prevent any break at production level given the succession of teams working on the same process and which leads to a certain number of specific work activities for these teams in view of an explicit or implicit cooperation.

Generally speaking, in this situation, the operators concerned possess similar skills and, contrary to the previous situation, their cooperation does not necessarily involve everybody being present at the same time. Furthermore, this type of cooperation does not only depend on the coordination of each person's representation and the division of tasks between operators, but on the operators' cooperation to carry out these tasks. On the other hand, this pooling does not fulfil the need to resolve a specific problem collectively, as is the case in the majority of collective work situations. It is the objective itself of this work phase.

Thus, it seems that :

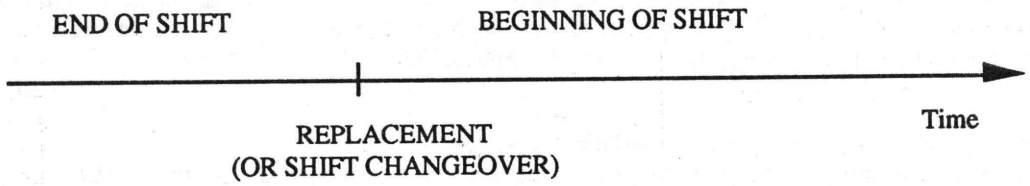
- the problem of the operators' functional representation is fundamental in these work situations and that the study of this is a necessary "detour" as much in the understanding of the operators' functioning as in the proposal of ergonomic solutions ;
- but above all, the notion of shared representation (1) is a perspective worth developing within the framework of collective management of processes and particularly in the case of shift changeover.

We can therefore illustrate shift changeover in the following way:



In contrast to the current definition, in which shift changeover is conceived as the replacement of one team by another in a continuous work process carried out by successive teams and taking only a very short time.

(1) We shall call shared functional representation, the functional representation shared by two or more operators during interaction.



The shift changeover thus appears to be a situation requiring specific analysis because:

- it is a collective work situation: the two operators (or shifts) have a common objective, to run the process, and must be able to carry out the same tasks ;
- this work phase is fundamentally interactive: exchanges, especially verbal exchanges aim at mutual adaptation, cooperation and coordination in order to avoid any interruption in production ;
- the incoming operator needs to bring his representation of the process up to date. The outgoing operator is in a position to convey to his collegues certain information linked to his own representation of the process ;
- by its very nature and objectives, this phase calls for communication between shifts and for the passing on of information, instructions, expectations, diagnoses and so on....;
- the fact that the incoming and outgoing operators share common tasks mean that their representations should not differ too substantially ;
- shift changeover therefore calls for some shared knowledge and a common reference system which is no doubt decisive.

Consequently, an analysis of the "shared functional representations" between operators or teams, at the time of a shift changeover, could lead to a better understanding of this work phase and provide elements for the developement of ergonomics solutions in order to optimise cooperation.

2. Method

The analysis of shift changeover in the context described above, ie. the analysis of functional representations, brings to light the problem of access to representations, but this methological difficulty is doubtlessly increased in the case of access to shared representations.

In most cases the question of functional representation in work has been tackled on an individual basis. The usual methods of access to representations are fundamentally individual. The implicit idea is retained that different "responses" are the manifestation of different representations. However, to our knowledge nothing justifies the existence of a bi-univocal link between the operators' representations on the one hand, and their know-how and their verbalizations on the other. Thus, differences in know-how and verbal explanation do not necessarily indicate differences in functional representations.

We therefore need to have access to shared representations. However due to these fundamentally individual methodologies, the notion of shared representation could well take on a very restrictive meaning: "*shared representations are identical or similar representations described separately by different persons*" (Trognon and Larrue, 1988).

We therefore assume that the analysis of spontaneous (or evoked) communications between operators is an adequate methodological option, in so far as:

- communication is an observable " natural " manifestation (and integrated into the work itself) that may give access to representations ;
- this option lies within a fundamentally interactive perspective ; common characteristics would be inferred from the interactions between operators ;
- it is not limited to a methodological approach based essentially on individual data ;
- and moreover, communication plays "*a decisive part in the comparison of experiences, the formation and development of knowledge and the negociation of each operator's field of action*" (Lacoste, 1983).

Furthermore, increasing the amount of data gathered and "confronting" it can constitute a "guarantee" in terms of the validity of the represention infered. Consequently, the collection of this communication data could be made in conjunction with other more individual means of access to the representations (for example, verbal statements after completion of work and analysis of information gathering).

Thus, we will consider that the only possible methodological option to gain access to shared representation consists of interactive situations analysis. The analysis of communication is thus a means of attaining this objective. The alternative options seem only to give us access to common representations (that is to say identical representations produced individually by operators) and not shared representations (whose particularity is to be created during interaction).

An analysis of shift changeover in the paper industry was carried out.
A method of analysing activities in the changeover phase was used and was specifically aimed at analysing functional communications, the gathering of information and operator's actions in order to gain access to their representation of the situation and the process.
The function concerned was that of machine conductor.
The midday shifts were recorded.

The following techniques were used:

- systematic on-site interviews and observations.
  A work analysis of the outgoing operator was carried out for a period of one to one and a half hours before a shift changeover with the aid of on-site observations.
- recording of communication between the operators at the time of shift changeover by means of a micro-transceiver.
- systematic observations (using a video-camera), gathering of information and actions when two operators meet and during the shift changeover.
- consecutive verbal explanations (verbalizations) when shown the video tape.

The paper machine on which the study was carried out, was chosen according to its degree of computerisation, operation mode and the resulting organisation of the shift work.
Apart from the preliminary observations, 14 shift changes were analysed. 20 verbal reactions to video tapes were obtained; these could not be systematically carried out with the outgoing conductor.

3. Results

We will present three types of result:

Firstly the analysis is aimed at confirming the interest in this specific phase, in identifying the different phases of the shift changeover and in presenting some descriptive quantative elements.

These results will not be presented in this paper (Grusenmeyer,1990; Grusenmeyer, Davillerd, Krawsky, 1992). However, we will bear the following two points in mind:

- It seemed essential to distinguish between different periods during this work phase, periods corresponding to work demands and different objectives.
  - The end of the shift. The time when the outgoing conductor prepares to meet his successor, by gathering information on the site and the man-machine interfaces. This is also the time when he can plan the work he must do during his next shift.
  - The arrival of the incoming operator. Period when he finds himself alone for a few minutes as his colleague is not immediately available for the changeover. The operator then takes information from the machine and the written documents in order to obtain a feedback of his previous day's activity and to find out general elements concerning the functioning of the machine.
  - The meeting characterized by a high rate of exchanges about the process, the work to be done, and the whole production system allowing the update of the incoming operator's process state representation and the facilitating the planning of the subsequent process operation.
  - the taking up of the post aiming at an update of the process representation by the incoming operator by means of direct information gathering on the process.

- in a very busy situation it can be seen that activities strictly connected to the shift changeover are delayed or reduced. This allows immediate activities regarding the process driving to be focussed upon. Consequently, in an incidental situation, a reduction in the amount of verbal exchanges at the time of meetings is observed. In this case the operators work together to start the machine again. Similarly, a longer shift changeover is observed in these situations. On the one hand, these are made up of immediate activities to remedy the incidental situation, or in the case of changes in production, recovery activities or dealing with the next order.

- **Systematic analysis of information gathering from the incoming conductors. Summary of the 14 cases analysed**

Secondly, the gathering of information and the communication between operators was analysed in more detail in order to understand how the operators proceed in updating their representation of the process.

This analysis concerns the activities of the incoming conductors at the time of taking up their shift. The incoming conductors are, indeed, often alone for a few minutes before the actual meeting as the outgoing conductor is not always immediately available to carry out the changeover.

This situation was interesting in so far as it was quasi-experimental as it permitted the understanding of the conductors activities before any information was received. Consequently, such a situation enabled us to really understand the incoming operators needs from the moment of his arrival at work. Obviously, this is not possible, if on arrival, he communicates with his colleague. In this situation the exchanges may, in fact, correspond less to the needs of the incoming operator than to the information the outgoing conductor considers necessary to pass on.
The aim of this study was to observe this gathering of information and to understand how the operator proceeds to take over the control of the process.

In order to analyse this, we noted for each of the situations observed, by means of video recordings, the conductors' activities at the time of taking up the post and we compared them with the results obtained regarding these same conductors during this period when they watched themselves on the video tape.
On the one hand, tables including the operators' activities and on the other hand, his remarks regarding the activities carried out were created for each of the situations.
We later constructed organigrams of the questions asked and the gathering of information from the incoming operator at the time of taking up his post on the basis of this analysis. Different organigrams were created according to the type of situation at the time of the changeover : normal working conditions, production change or incidental situation.
These organigrams show the questions asked by the incoming operator, his gathering of information in order to answer these questions, the conditions observed, but also what the operator remembers as this could be a source of information.
Finally an organigram was constructed grouping together all of the situations observed. (Cf. Figure 1).

These analyses enable us to distinguish two types of information gathering from the conductors during that time.

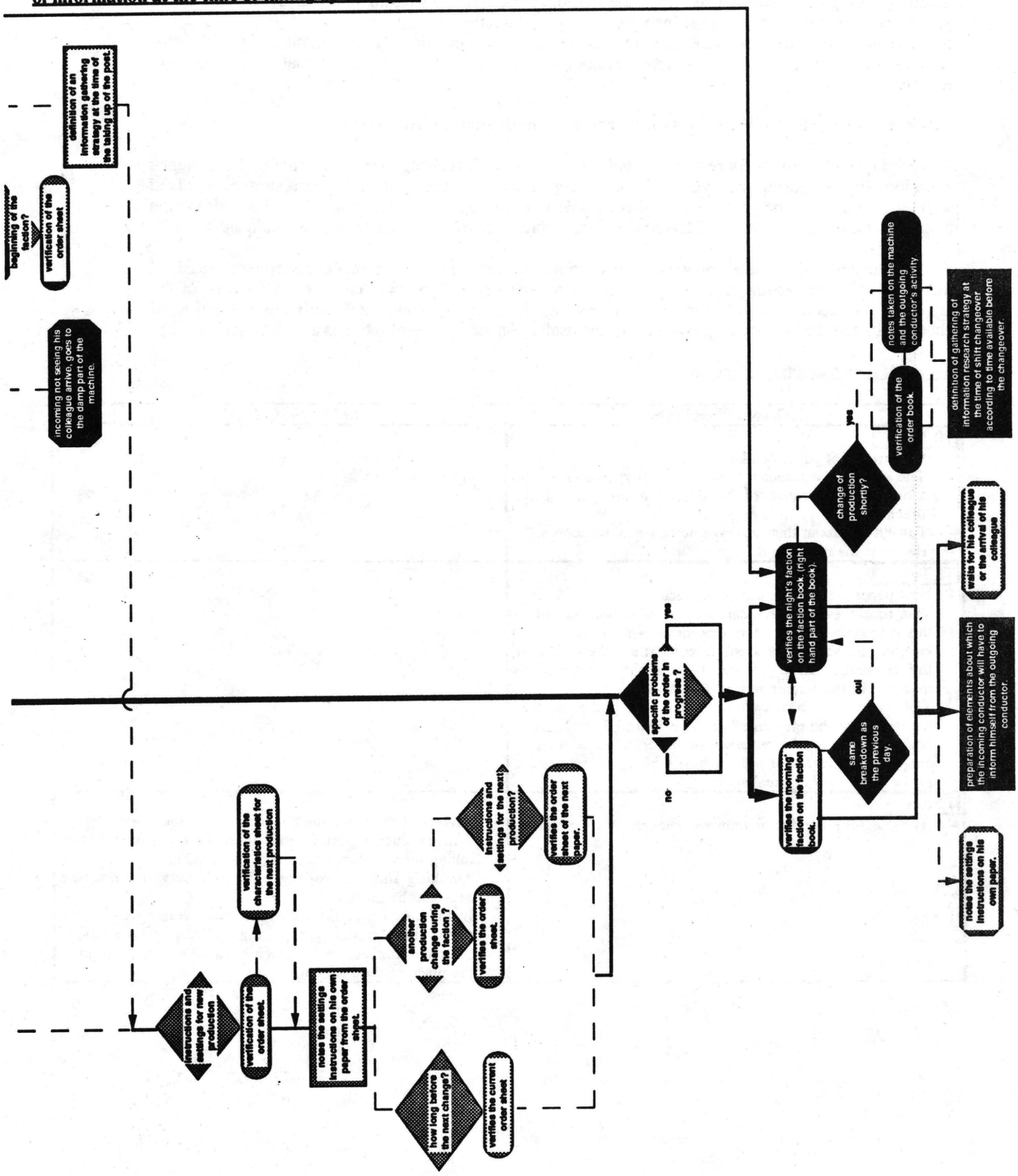
- information search in order to globally identify the production phase. This means that the operator has to check if the machine is working, if he is dealing with the same order as the previous day, if a change of production has been planned for the beginning of the shift.
  This gathering of information is common to all the situations observed. The operators move within a network of the systems conditions. There is therefore a global strategy common to all the conductors. This would reinforce the existence of shared functional strategies, probably representative of shared functional representations.
- the gathering of more circumstantial information linked to the type of situation encountered and concerning the more precise variables of the machine's operation : for example, modifications in functioning, quality of the paper obtained, the width of the product being manufactured ...
  These gatherings of information are, contrary to the previous ones, specific to the production phase. This means that the conductors have chosen a strategy adapted to the previously described situation, from a group of possible information strategies.

Thus the controller has to take information from the production phase on his arrival at work, allowing him to choose a information gathering strategy adapted to his own global situation representation : these strategies for information searching being specific to the type of situation and each one more or less clearly defined from eachother.

Furthermore, these results make additional information available. We have shown all the "routes" taken by the operators on the 14 shift changeovers observed. The repetitiveness of the different information gathering strategies is shown on the organigram by the thickness of the lines.
We can, thus, see that contrary to the first information searches, chosen strategies for information gathering are not common to all of the operators observed once the production phase was totally evaluated. Certain routes were used by only one conductor having a strategy different from the aim, orientated towards paper quality for example. This is the case for information gatherings concerning the quality of the paper at the present time, through checking the characteristics sheet.

Figure 1. **Organigram of conductor's research and gathering**

of information at the time of taking up their post



This organigram demonstrates different operators representations of different aims.
This means, therefore, that conductors have partial shared representations concerning the aims, and consequently, different information gathering strategies at the time of taking over the job. These are not fully common to all operators. However, this lack of global sharing regarding the ways of carrying out a shift changeover and regarding the aim representations can cause a problem if the information supplied by the outgoing conductor to his incoming colleague does not correspond to the information the latter would automatically gather, but to that which the outgoing operator, in keeping with his own aims, deems necessary by putting himself in the incoming operator's situation.
The incoming operator, thus, risks having more difficulties in creating a representation of the situation and the process as obtained information does not adequately correspond to the information he would like to receive.

- **Analysis of the failure in transferring a malfunction diagnosis.**

The third type of analysis concerns the detailed analysis of a shift changeover in the case of an incidental situation. By comparing the systematic recording of a changeover with the two operators involved watching themselves on video, we have been able to observe the failed transmission of a malfunction diagnosis, meaning an absence of shared functional representation in the case of a given incident.

To better understand the situation we have tried to reconstruct all the events that occurred on the basis of the morning's observation grids, the activities and exchanges of the two operators at the time of the arrival of the incoming conductor and their meeting, the incoming conductor's activities at the time of changeover and the two conductors watching themselves on video. The situation is as follows (Figure 2):

Figure 2: Reconstruction of the activities

OUTGOING OPERATOR	INCOMING OPERATOR
T0 Time. - Occurrence of paper breakage. - The outgoing conductor puts forward hypothesis A, the most frequent cause of breakage in this part of the machine. - To find out about this breakage he consults the screen. - He starts the machine off again.	
T1 Time. - New paper breakage in the same place - This time the conductor can see no indication related to this type of breakage when hypothesis A is cheked. - He puts forward Hypothesis B concerning the origin of the breakage based on the fault noticed during the morning in the functioning of the machine. - The conductor has two conflicting hypotheses, but knowing that the element X at the origin of hypothesis A probably needs to be replaced, he nevertheless decides to change this element of the machine which willin any case prevent any further breakages.	
T2 time - The outgoing conductor removes element X (Hypothesis A).	- Arrival of the incoming conductor from the machine. - The conductor notices that there is a breakage when taking direct information from the machine. - Thinking that his colleague is switching the machine back on, he takes up his post. - In the absence of his colleague, the incoming conductor thinks that the process is taking too long to be a simple switching back on of the machine and he decides to go to the place of the incident.

T3 time. - the outgoing conductor has removed the element X. The element X needs to be replaced. - The outgoing conductor tells his colleague that it is necessary to change X. - The two conductors fit element X'.	- The incoming conductor sees that his colleague has removed element X. As this element causes type A breakages, he deduces that hypothesis A caused the breakage, as this is one of the most frequent causes of breakages in this part of the machine. - The incoming conductor is persuaded that X is at the origin of the breakage (Hypothesis A). - He goes to find a replacement element X'.
T4 time. - Shift changeover. The outgoing conductor then informs his colleague about the two breakages and about his decision. He fails to tell him about hypothesis B due to the busy situation.	
T5 time. - The outgoing conductor leaves. He notes down the two breakages in his machine notebook.	- The incoming conductor is convinced that X is the source of the two breakages. - His priority is to switch the machine back on which he does immediately. - He does not consult the data, when the machine is stopped, because he knows that this interferes with the computer system. - On taking up his post, he does not look for further information about breakages which have occurred.

We must remember that all this information is a summary of all the collected facts through observations and operators watching themselves on video. All of these elements are factual, having been directly observed or announced by the operators. It is, thus, a retranscription of observed elements during recordings or the verbal explanations from video tapes.

During this situation, the outgoing and incoming conductors therefore have a partly different representation of the cause of the breakages which occured at the end of the morning, and consequently of the machine's functioning. Whilst the outgoing conductor envisages two conflicting hypotheses regarding the cause of the breakages, the incoming conductor has only one hypothesis himself. Thus, the outgoing conductor has a richer representation of the situation in comparison with that of the incoming conductor.

Several elements contribute to this partially different representation on the part of the two conductors regarding the events of the morning. These elements are the following:

a. The two operators' different levels of information.
Firstly, the two operators have different levels of information. The outgoing conductor has information that the incoming conductor does not have. Part of this concerns a malfunction noticed in the morning in the running of the process. It is the outgoing conductor's knowledge of this malfunction which allows him to put forward a second diagnosis regarding the incident which occured. Knowledge of this malfunction is linked to the past history of the running of the process in the long term, a past probably not known to the incoming conductor.
There is another type of information which is not available either to the incoming conductor. It is the instant representation of the breakage and particularly the absence of an indication related to this type of incident. It is this instant representation of the incident which will allow the outgoing conductor to initially question his first diagnosis, and later to give a conflicting diagnosis linked to his knowledge of a malfunction in the running of the process.

b Interpretation of the outgoing operator's behaviour by the incoming operator.
Secondly, the incoming operator interprets the situation according to his own knowledge of the process and of the procedures to apply in certain situations. Indeed, on arriving on the site of the incident, the conductor sees his outgoing colleague carrying out an action which is normally done to remedy the incidental situation. The acknowledgement of this action combined with the acknowledgement of an incident occuring at the end of a shift, contributes to the fact that the incoming operator interprets this action as a remedial action to an incidental situation caused by element X of the machine.
Therefore, he interprets his colleague's behaviour in terms of the incident remedial procedures he remembers and in terms of his overall knowledge of the process. Firstly, he interprets the removal of element X as the manifestation a recovery to an incidental situation; secondly, he knows, given

his experience of the operation of the machine, that this element is the most frequent cause of breakages in this part of the machine, and he comes to the corresponding hypothesis.
In this situation, two factors come into play. The first concerns the interpretation by the incoming operator of his colleague's behaviour according to his own understanding of the process and of the procedures to adopt in such a situation. The second is the very short exchanges relating to the procedures to be adopted immediately and not to the origin of the breakage. This second element is similar to that observed by Cuny (1967) and Savoyant (1977) in a repetitive situation, that is "*the reduction in verbal exchanges to the minimum necessary for the performance of a task, information gathered on site replacing that provided by the language*". In the present case, it seems that it is less the repetitiveness of the situation than the experience of conductors in this type of situation and their mutual understanding of eachother's way of working which leads to this "failed" transmission diagnosis.
Thus, the incoming conductor interprets the situation in terms of the information gathered from the work situation (notably his colleague's behaviour) and not by the means of exchanges with him. Yet, the outgoing conductor had chosen to replace this machine part, not because this corresponded to his diagnosis of the incidents that had occurred, but it enabled him to anticipate future breakdowns caused by wear and tear on this part of the machine. The incoming conductor thus interprets his colleague's behaviour not as an anticipation activity as is the case here, but rather as a means of remedying the incidental situation which has just occurred. He therefore interprets the situation through his understanding of his colleague's behaviour.

c. Overload.
The third element contributing to the explanation of the reason for the incoming conductor knowing less than the outgoing conductor is the fact that the latter fails to inform his colleague of the morning's malfunction due to the loaded situation. This is what the outgoing conductor explains when watching the video tape of himself. In reality, an analysis of the information provided by the outgoing conductor to his colleague at the time of shift changeover shows that the former does not focus on the diagnosis of the incidents, but on what he has done. Thus the incoming conductor will interpret this information as a diagnosis concerning the origin of the breakdown and not as a description of the work done.

d. Productivity / reliability conflict.
Finally, a fourth element will reinforce the situation. The incoming conductor's priority is to start the machine again. He therefore concentrates on this task. Moreover he does not consult the history file neither immediately nor later, knowing that, with the machine stopped, it would interfere with the computer system. The conductor concentrates on production and checks neither the reliability of his information nor the diagnosis he made.

In this situation, the shared representation is partial. The sharing takes place at a general level which relates more to the general knowledge of the most common breakdown diagnoses than to a specific representation of the breakdown which has just occurred. The sharing which takes place at this representation level does not appear to be sufficient as the incoming conductor does not have as complete a representation of the machine as the outgoing conductor's.

4. Conclusion

This analysis brings several factors to light contributing to the establishment of a partly erroneous representation of the incidents by the incoming conductor:
- some information is no longer available ;
- the outgoing conductor seeks information and analyses the situation in terms of elements coming from his workplace and not from verbal exchanges with his colleague. These exchanges are minimum and only relate to the immediate tasks to be carried out.
- the outgoing conductor focuses his exchanges on the work he has carried out and not on the diagnosis of the the situation.
- the incoming conductor cannot consult the history file straight away.

Taking these analyses as examples and not generalisations, they do enable us however to bring to light a certain number of mechanisms leading to a non optimum shift changeover. Thus they form a basis first for propose different hypotheses to be experimentally tested and for ergonomic recommendations for companies.

Moreover they emphasize the importance of shared functional representations between operators at the time of shift changeover.
An increase in the number of such analyses should in the future, allow a better understanding of the cooperation processes and of the possible risks due to a sharing of knowledge which could be weak or too strong between successive operators.

Furthermore, these analyses show that it is possible to study representations shared by operators by comparing the conductors' activities with their reactions to video recordings of themselves. The establishment of a partial shared functional representation which led to a failure in the transmission of an incident diagnosis has been brought to light thanks to a systematic analysis of activities, actions, and operators' verbal exchanges and comments about their activities.

References:

Alengry P. (1988). Comportements anticipateurs et activité prévisionnelle dans une tâche de contrôle d'un système dynamique, rapport INRIA.

Chabaud C., De Terssac G. (1989). Dimension collective des savoir-faire individuels. Premières Journées de Psychologie du Travail "Ergonomie et psychopathologie du travail", PIRTTEM-CNRS.

Cuny X. (1967). La circulation de l'information dans un système élémentaire d'un service de transport. Bulletin du CERP, 16.

Godimus J., Furon D., Frimat P., Cantineau A. (1977) Les problèmes médicaux posés par le travail posté. Communications sur les résultats d'une enquête relative au travail posté, CRAM du nord.

Grusenmeyer C. (1990). La relève de poste: une phase critique du travail en équipes successives. Vandoeuvre- lès-Nancy, INRS, sept, NST 80.

Grusenmeyer C., Davillerd D., Krawsky G. (1992). Les procédures de relève de poste dans différents systèmes industriels: incidences sur le travail et la sécurité. Rapport au Minitère de la Recherche et de la Technologie.

Keyser V. de (1985). Les communications dans les systèmes uatomatisés: Champs cognitifs et supports d'informations chez les opérateurs. In: La communication, Symposium de l'Association de Psychologie Scientifique de Langue Française, PUF.

Lacoste M. (1983). Des situations de parole aux activités interprétatives. Psychologie Française, 28, 3/4.

Navarro C. (1991). Une analyse du travail cognitive de l'interaction dans les activités de travail. Le Travail Humain, 54, 2, pp 113-128.

Neboit M. (1991). Ergonomie et fiabilité humaine dans la conduite de systèmes complexes. Revue Générale d'Electricité, 5, pp 36-40.

Rogalsky J (1989). Cooperative work in emergency management: analysis of verbal communications in distributed decision making. Communication au colloque "Cognitive Science Approaches To Process Control", Sienne, Italie, 24-27 Octobre, 1989.

Samurçay R., Hoc J.M. (1989). Spécification et évaluation d'aides logicielles aux activités de contrôle d'un environnement dynamique: conduite de hauts fourneaux. Premières Journées de Psychologie du Travail "Ergonomie et Psychopathologie du Travail", PIRTTEM-CNRS.

Savoyant A. (1977). Coordination et communication dans une équipe de travail. Le Travail Humain, 40, 1, pp 41-54.

Trognon A. Larrue J. (1988). Les représentations sociales dans la conversation. Connexions, 51, 1.

part 3
COOPERATION AND COMMUNICATION: THEORETICAL APPROACHES

Designing IBC services which enable users to reach their goals

Jon May, Ian Denley †, Ulla-Britt Voigt*, Sibylle Hermann*, Paul F. Byerley**

*Terminals Research Centre
Standard Elektrik Lorenz AG, Postfach 1760, D-7530 Pforzheim, Germany.

†Ergonomics Unit,
University College London, 26 Bedford Way, London WC1H OAP

ABSTRACT

This paper concerns the needs of users of telecommunication services. In particular, it concerns human users accessing telecommunication services via terminals, to realise their goals, such as co-operative distributed working, or interpersonal communication. A design method is described which incorporates design principles, and which aims to improve end-user service integration. A case history in the design of a cooperative multimedia multiauthor document production system is given.

1.0 Introduction

This paper concerns the needs of users of Integrated Broadband Communication (IBC) services. In particular, it concerns human users accessing telecommunication services via terminals, to realise their goals, such as co-operative distributed working, or interpersonal communication. A service can be defined as a facility which is provided to a user, and the user obtains a service by interacting with a service provider.

The GUIDANCE project (see acknowledgements) has developed principles which can assist the designer in improving end-user service integration and the usability of services. These design principles are applied within a design method, which GUIDANCE has incorporated into a software tool (see Terrins-Rudge, 1992).

This paper focuses in particular on how an IBC designer should use the design principles and the method and illustrates these ideas with a worked example of the design of a system to support simple group editing and communication tasks. This illustration is based on a prototype cooperative document production system designed by the project, and which has acted as a vehicle for the development and the testing of the principles and the method.

1.1 IBC and End-User Service Integration

In a typical current office an office worker might use on a daily basis a number of services, including: telephone; fax machine; a remote database; wordprocessor; spreadsheet, and so on. With IBC technology, all these services (along with many others) will be delivered to users at the same physical terminal, since the various technologies will be integrated in the same network. However, although the physical problems of moving between and accessing separate items of equipment will largely disappear in IBC systems, the mental problems of learning, conceptualising and recalling how to use services will not disappear, and may even be compounded by an increasingly large number and variety of services becoming available. Users will almost certainly be faced with service integration problems affecting ease of use, such as: variations in the semantics of input device events; inconsistent screen layout conventions; confusion between operating procedures; incompatibilities between the representations of objects; and differences between representations to support collaborative working. Each of these integration problems will tend to have a negative effect on the ease of use of the set of services. This effect might manifest itself in various ways, for example, increases in learning times, or in data input times, or in command language syntax errors, or in user dissatisfaction, and so on.

2.0 Two criteria of usability

The analysis of usability requirements for IBC begins with the concept of a network of users (humans or machines) which are connected by communication channels. This network as a whole has the function of supporting the achievement of an application goal. Figure 1 shows an example network description for a co-operative authoring scenario that has been used as a test-bed by the GUIDANCE project, multimedia multiauthored document production, or MMMADP. This example will be used throughout this paper to illustrate the use of the GUIDANCE method and design principles.

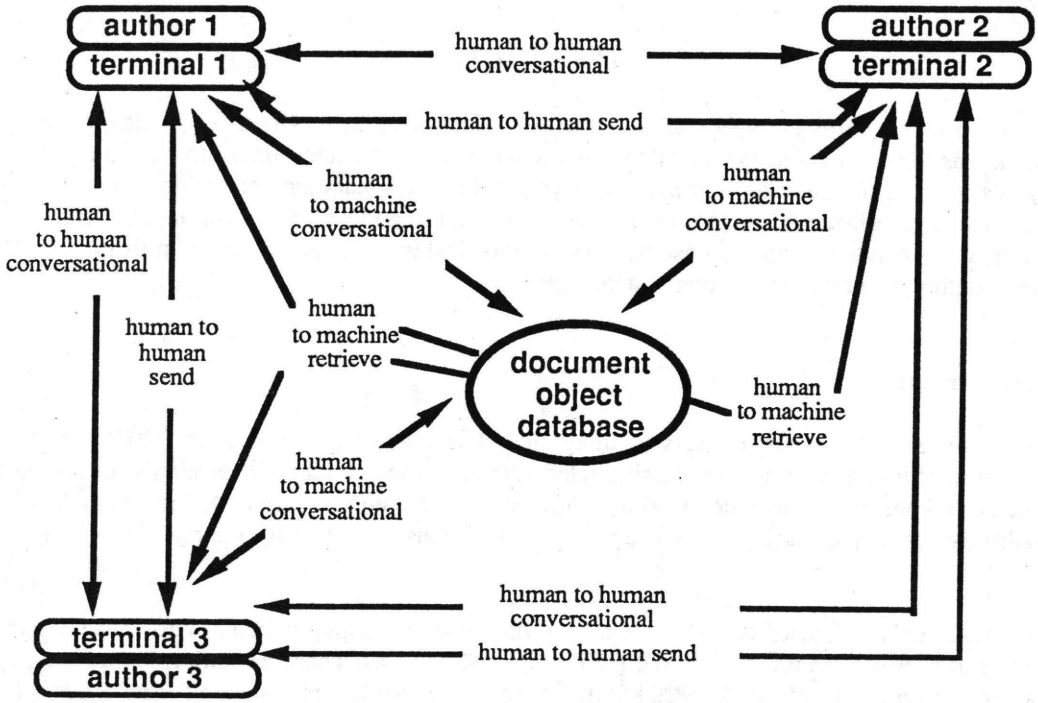


Figure 1: An Example Network for Multimedia Multiauthored Document Production (MMMADP)

Each communication channel illustrated in Figure 1 can represent one or more services, and each service may involve more than one service provider. Usability engineering takes the perspective of the human user within such a network. The human user will have certain goal tasks which they must accomplish in co-operation with other users (humans or machines), with which they must interact via communication services. To achieve these goals, the user will require particular combinations of services. These combinations will vary between different tasks, and also between different instances of the same task.

GUIDANCE has proposed a distinction between two types of task that must be performed. There are tasks by which the users intend to move closer to their goal (*goal tasks*), and there are also tasks which create or modify the system to bring it into a state which enables the user to execute their goal tasks. In IBC, these *enabling tasks* include accessing and modifying services, by interacting with service providers.

Goal tasks and enabling tasks are distinct in a number of ways. Since usability engineering focuses on design for a human communicating entity, the goal tasks are those for which the human is the agent. Enabling tasks, by contrast, could be allocated to any part of the system. In fact a system is more usable if these tasks do not have the user as the agent, but rather have them allocated to a machine entity.

Another way of looking at this is to say that whereas goal tasks are what the user wants to do, enabling tasks are what the user has to do, given their goal tasks. Enabling tasks can therefore be derived from an analysis of the preconditions for goal tasks, that is, from an analysis of the *enabling states* of the IBC system. The enabling state of a system is one which allows the user to perform their goal task.

In line with this analysis a usable IBC system:

- ensures that the appropriate enabling states exist for each of the users' goal tasks
- reduces to a minimum the costs to the user in reaching the appropriate enabling states for their goal tasks

The GUIDANCE method assists designers of IBC systems in meeting these criteria, and in improving end-user service integration, by providing design principles which are applied at appropriate points in the design method. Denley et al (1992) have suggested that design principles are a useful way of helping designers solve part of the IBC service integration problem provided that they are seen an adjunct to other design techniques. This approach has been taken in the GUIDANCE project where principles are an integral part of the design method developed on the project.

2.1 Twelve Usability Design Principles

Principles can be classed along a dimension of generality. For example, a generic principle might be applied across a range of IBC work domains, whilst a specific principle might only be applied to a particular work domain. Principles with varying degrees of generality between these two extremes are also possible.

The research strategy employed has been to develop specific principles in the context of MMMADP and then to 'generify' them to other IBC domains. Some 150 specific principles have been identified during the design of the MMMADP system and these have been classified within a broad generic set developed from a survey and review of over 300 design recommendations (see Esgate et al, 1990). The following list presents these 12 principles with short, non-technical descriptions, followed by a detailed description for *Coherence*. Examples of MMMADP principles are given in Section 2.2.

Compatibility: The user should be able to use knowledge they have gained from outside the system.
Coherence: The user should be able to generalise their experience from parts within the system to other parts within the system.
Simplicity: The number and complexity of necessary actions should be reduced to a minimum.
Redundancy: The user should be provided with summary information in several ways.
Salience: Critical information should be presented to the user in a sufficiently intrusive way.
Transparency: Information about tasks which can be performed, as well as about the states of the machine, should be stated unambiguously and clearly.
Completeness: Information about tasks which can be performed, as well as about states of the machine, should be complete in such a way that all options are presented simultaneously.
Support orientation: If the information to be presented is too complex or covers more than is possible to present at one time, the user should be helped to find the relevant information by giving them support in orientation.
Feedback: The user should be informed about the consequences of their actions.
Reversibility: The user should be able to restore pre-existing states of the machine.
Controllability: The user should be able to control actions of the machine.
Flexibility: The user should be able to choose the modalities of their task performance with respect to the input media available and the experience they have.

Coherence

Definition

The user should be able to generalise their experience across different parts of the system.

Rationale

This allows the user to use already existing knowledge, and reduces the memory load of the user.

Detailed information

Coherence, with respect to *integration of services*, is to be understood as enabling the user to predict the range of possible machine states across the different services by using knowledge from a known service.
Coherence, with respect to *hypermedia*, leads to the design specification to offer all media in all applications and allows the user to use the same set of commands in all these applications.
Coherence, with respect to *CSCW*, has the consequence that all users should be able transfer their existing knowledge to other parts of the system. For example, commands should have the same meaning across applications and across users (team-members).

2.2 Defining the principles

The expression of the GUIDANCE design principles relates together users, their tasks, service environments, desired task product quality and cost (e.g. cognitive load) with a system description. Figure 2 gives an example of the relationship between these parameters and the design prescription with its associated rationale.

IF the **task** is to gain an overview of all members of the workgroup, objects and functions available to plan and carry out actions

AND the **users** are a limited number of co-operatively working multi-media authors and editors creating/modifying objects and exchanging messages

AND the **service environment** supports graphical representation of information

AND **performance** should be not slower or more error prone than if working in an open plan office

THEN APPLY PRINCIPLES OF:
a) **COMPATIBILITY**
b) **SIMPLICITY**

a) **COMPATIBILITY**

THEN **Layout:** use an open plan office model: (all members of the workgroup, objects and tools represented at the same time)

Interactions: similar interactions/actions possible as in an open plan office

BECAUSE: the users can recruit existing knowledge about open plan offices to plan and carry out actions

b) **SIMPLICITY**

THEN **Layout:** distinct and easily understood display for objects, persons and functions available

BECAUSE: an unambiguous overview should reduce user memory load

Figure 2: Examples of Specific MMMADP Principles

These specific principles are restricted to tasks related to multimedia document production, users who are authors (multiple and distributed) and machines running integrated services to support document preparation. Both the conditions and the prescriptions of the principles are closely tied to components of the method.

For example, the conditions reflect possible values specified in the specification tables of the method. As the designer fills in a specification table a number of values for a particular parameter may be suggested. The selection of these parameter values influences the presentation of appropriate principles in support of necessary design decisions. For instance, if the designer specifies an audio/video editing task then relevant principles for such a task may be accessed if required. These principles' prescriptions would provide recommendations for the completion of particular 'slots' in the specification tables, and would reflect the design decisions required for those tables. The prescriptions applying to functional software design are expressed in terms of conceptual/semantic design (e.g. how domain concepts are represented by the system), whereas the prescriptions applying to physical software design are expressed in terms of lower level design decisions such as dialogues, interactions, layouts and device considerations.

2.3 Meeting the First Usability Criterion

In considering the first criterion we need to define more precisely what is meant by an enabling state. An enabling state describes the preconditions for a goal task. A goal task is conducted by a particular user on a particular object. The task may require the support of a service in order to be executed. Therefore, an enabling state description consists of a three tuple:

Enabling State=(User, Service Relationship, Objects)
where:
User = required characteristics of the user, e.g. knowledge, attention
Object= required characteristics of the object, e.g. 'file available',
Service Relationship = the interface between the user and the object.

Design principles which aim to assist the designer in meeting this first criterion will therefore need to provide recommendations concerning all three components of an enabling state.

For instance, the enabling state will describe the state of the user that is required for them to execute their goal task. If there is a mismatch between the **required** user characteristics described by the enabling states description and the **actual** user characteristics then a principle can be applied which suggest a solution to this problem. For example, a principle might recommend help pages or training to provide the user with the knowledge necessary to execute their task.

Similarly, a principle may provide recommendations about an **object** of a goal task. An obvious example in telecommunications is that a user who wishes to talk to another person needs that person (the object of their task) to be alert and receptive. This state of alertness could be created by an enabling task conducted by the system, such as making their phone ring, or could be conducted by the user saying "excuse me". The design recommendation in this case would concern that object state which is conducive to the goal task to be performed.

The **service relationship** is a crucial part of the enabling state description. The service relationship is similar to the concept of a user-interface used in the design of stand-alone computers. However, for a distributed system of communicating entities sharing tasks, the interface between a user and an object becomes more complex, as described below, and the term service relationship is therefore preferred.

Four basic service relationships are proposed. They are generated by considering the user's goal task actions and the goal task object. The *action* and the *object* are subdivided according to the following criterion:

"Does the Service Relationship require a model of either the action or the object?"

If the Service Relationship contains a model of the user's actions then that action is said to be *interpreted.* This type of Service Relationship is shown in Figure 3.

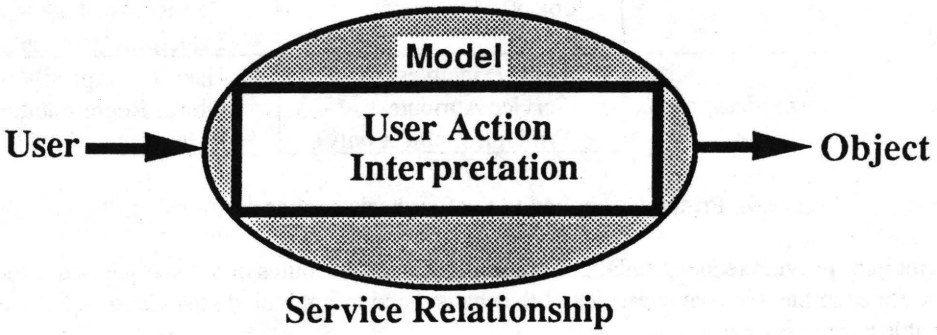


Figure 3: A Service Relationship Containing a Model of User Actions

If the Service Relationship contains a model of an object then that object is said to be *represented.* This type of Service Relationship is shown in Figure 4.

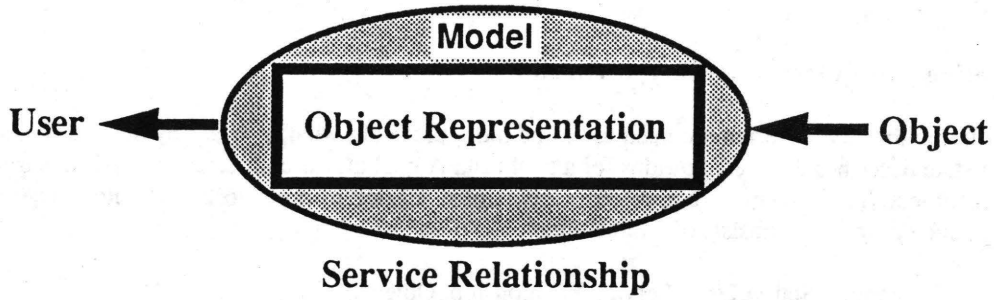


Figure 4: A Service Relationship Containing a Model of Goal Objects

Although the presence of any Service Relationship means that the user's task actions are indirect, since their input requires an input device, this does not necessarily imply interpretation. Thus spoken words in a telephone are not normally interpreted in the Service Relationship, and text input via a keyboard to a database is not normally interpreted in the Service Relationship. Of course, both may be interpreted by the goal task object. This schema provides four kinds of Service Relationships, as shown in Figure 5.

		Goal Task Object	
		unrepresented	*represented*
User's Goal Task	*uninterpreted*	1. e.g. real time human to human conversation using audio/video	3. e.g. voice activated database retrieval
	interpreted	2. e.g. human to human conversation using text to speech synthesis for a dumb person	4. e.g. graphical/text editing using a mouse

Figure 5: Examples of the four basic service relationships

Meeting the first usability criterion, therefore, requires guidance concerning the **service relationship**. Design principles can provide recommendations about service attributes (e.g. media and media quality) and about the representation of objects and the interpretation of task actions. Figure 6 summarises the types of principles applied for the four different service relationships.

		Goal Task Object	
		unrepresented	*represented*
User's Goal Task	*uninterpreted*	1. Service Attribute Principles (task and object).	3. Object Representation Principles; Service Attribute Principles (task only).
	interpreted	2. User Principles; Service Attribute Principles (object only).	4. User Principles; Object Representation Principles

Figure 6: Principles applied to the four basic service relationships

In case 1, principles provide recommendations about the service attributes of the service in question. These may concern, for example, the characteristics of the channel and might include the video and voice qualities needed to enable a conversation.

In case 2, principles provide recommendations concerning the mapping of the user's actions to the model. For example, a *flexibility* principle might recommend that users should be able to perform tasks in different ways, for example with different input media. Additionally, because the task action is interpreted no channel descriptions are necessary for the action but may be provided, however, for the object.

In case 3, principles provide recommendations concerning the mapping of the object to the model. For example, the representation of a goal object should be *compatible* with the user's experience, and so a compatibility principle might recommend that a text file might be represented as a small page of text. Additionally, because the object is represented no channel descriptions are necessary for the object but may be provided, however, for the actions.

In case 4, principles provide recommendations concerning the mapping of the user's actions to the model and the mapping of the object to the model. There will be no prescriptions concerning channels.

Meeting the first usability criterion, then, involves the use by the designer of a number of different types of principles. Further examples of all these principles are given in Section 4.

2.4 Meeting the Second Usability Criterion

Design advice can also be given on the second criterion, concerning the **costs** for the user which are associated with acquiring an enabling state.

Different types of user costs can be interpreted with respect to the twelve generic usability design principles. For example, each of the twelve generic principles can be interpreted with respect to mental cognitive costs. An example of mental cognitive costs is the cognitive workload for the user associated with presentation of multiple sources of complex information. A *compatibility* principle recommends that such information should be structured and presented to the user in such a way that the user can use their existing knowledge from outside the system to locate appropriate information and establish its relationship to other information sources. Similarly, a *salience* principle recommends that critical information should be clearly identifiable, whilst *reversibility* suggests that users should be able to reverse their action thus bringing the worksystem back to its former state.

Another aspect to reducing the users costs concerns the decision of who (or what) does the enabling task. For this reason, there are also principles which give advice about the **function allocation** between user and machine. One overall aim is the system property of *simplicity*, which says that the actions that the user has to perform should be reduced to a minimum. Other important function allocation principles are *controllability* which recommends that the user should be allocated control over critical system functions, and *flexibility* which recognises that there is not always a single best way for operating a worksystem and recommends that where performance varies with organisational contingencies there should be flexible allocation such that a function can be performed either by the machine, the user or by both.

2.5 Principles and Service Integration

So far we have discussed how design principles provide design recommendations that assist the IBC designer in meeting the two criteria for a usable system. That is, the principles provide prescriptive advice concerning the three-tuple of users, service relationships and objects which comprise the enabling states description, and also provide prescriptive advice concerning the acquisition of an enabling state, via the enabling tasks.

Recent GUIDANCE publications (Whitefield et al 1992, Denley et al 1992) have suggested that lack of integration between services may seriously affect the uptake of IBC systems. End-users will still face problems caused by , and these problems may make the integrated systems usability even worse than existing independent services. A major aim of the design principles proposed by GUIDANCE is to help the designer overcome the problems of service integration.

In effect, then, GUIDANCE has delivered two sets of interelated principles: service integration principles and non-integration principles. Obviously, both types of principles have important roles in promoting the usability of IBC systems and services.

Integration principles address the interactions or interleaving of the end-users goal tasks. For instance, taking a medical example, a radiologist seeking expert advice from a remote expert may wish to retrieve

patient information including x-rays, stored as image or text files in a remote database and send these to the remote expert whilst continuing to communicate using a multimedia conferencing service. Obviously, there is an interleaving between the goal tasks communicating with a colleague and retrieving and sending files, which may cause service integration problems for the end-user. For instance, the usability of these services will be reduced if there are inconsistent screen layout conventions (e.g. different positions for common buttons in screens or windows for different services); or if there is confusion between operating procedures (e.g. similar database services may require very different access commands depending upon where they are physically stored).

Service integration principles assist the designer in solving such integration problems between services. Section 3 illustrates where and how the different types of principles described above are used in the IBC system design process.

3.0 Using Service Integration Principles in Design: A Simplified Case History

3.1 A Multimedia Conferencing Scenario

Consider a Computer Supported Cooperative Work (CSCW) task that would be quite typical within an IBC environment: the creation of a multimedia virtual document by several authors located at different sites. An example of a subtask within this might be that two experienced authors have to agree and to implement changes to a section of the document within a few hours. This would involve the authors in planning the required changes between them, in making particular changes, in commenting on the other's changes, and so on. These are the activities that will achieve their goal of revising the document section. To carry out this subtask, the authors will use telecommunication services, and the services they use will vary as they proceed through the subtask. Despite the fact that the authors are not directly interested in the services themselves, they will have to carry out a number of activities to be able to use the services; for example, setting up a videotelephone link, accessing databases, establishing an electronic mail link, the authors to achieve their goals, but they do not themselves achieve the goals.

The specific scenario used here takes a snapshot of the group authoring task to illustrate how the interleaving of tasks affects the design through the application of principles. Imagine that two remote authors wish to discuss and edit a video document that is stored on the document object database. Section 4 works through the design of the services to support *two* of the tasks the authors would have to carry out, and demonstrates the importance of the principles in the design. The two tasks considered are:

- Human to human conversation: the authors discuss proposed changes to the video clip
- Human to machine conversation where the authors use a remote video editor

The following illustration is a simplified example, which although based on a real design, is used for illustrative purposes only, and the details of the specification method itself have been omitted. The purpose of this illustration is primarily to show how design principles influence the design of services. Readers interested in the design method are referred to Whitefield et al (1992), and the GUIDANCE Position Paper (1992).

3.2 Four Design Stages Resulting In End-User Operational Requirements

There are a number of stages in the design of any IBC system such as the multimedia telephony service described above. These stages are based on two perspectives of the system to be designed: the customer (application level) perspective and the human communicating entity perspective. The customer is external to a system, and a human communicating entity is internal to the system. Obviously any individual could have one or both roles. Both these perspectives are concerned with identifying the operational requirements of the system to be designed. We are only concerned here with the network design from the human communicating entity perspective, and readers interested in the customer perspective are referred to Byerley and Bruins, 1992.

The GUIDANCE project has developed a method for the design of IBC systems from the perspective of the human communicating entity. Prior to software design there are four stages to the GUIDANCE design process:

Stage 1 = Analysis of task usage context.
Stage 2 = Analysis of users' goal tasks
Stage 3 = Analysis of required enabling states for the users' goal tasks
Stage 4 = Analysis of the required enabling tasks which create these states

Stages one and two, the analysis of the usage context and the users' goals tasks, provide a description of the need for communication, i.e. the requirements for communication for particular tasks rather than the specific features of the services that will allow these communications to occur. Stages three and four, the enabling states description and the enabling tasks description, describe the service attributes needed to support the communication tasks described. The output of all four stages is a set of operational requirements which form the basis of the software design.

The usability engineer begins the design by breaking down of the user's goal tasks into smaller subtasks. The goal task breakdown is continued until the tasks require only one kind of service relationship. For each of the subtasks an enabling state description is then made in terms of user, service relationship and object.

Following this stage, service integration issues can be considered by examining the task sequences. If two or more subtasks are to be allowed to interleave, for example, then the integration of their enabling states must be designed. This means that the two enabling states must be integrated and a third *integrated* enabling state description produced.

Thus, service integration issues are treated after the initial enabling states analysis of the separate subtasks, by forming task combinations and examining the resulting sets of service relationships, and that this process results in an integrated enabling states description for the interleaving tasks. Figure 7 shows the relationship of each type of principles to this design process:

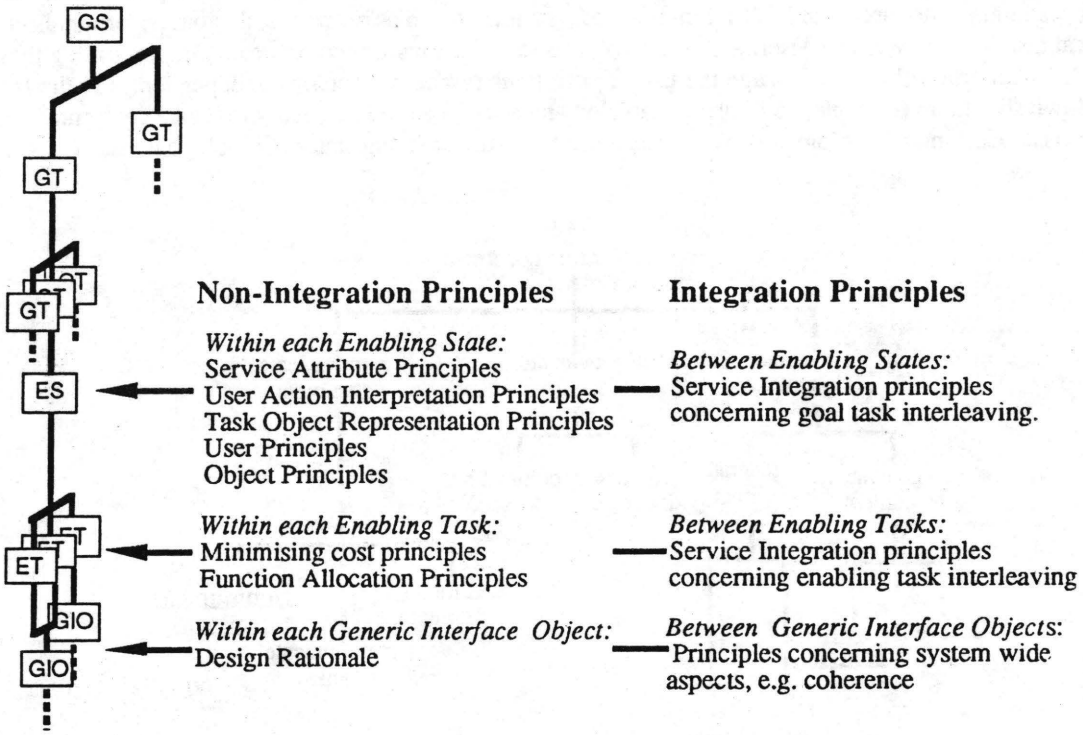


Figure 7: The identification of service integration principles within the GUIDANCE methodology

3.3 Stage 1: The Usage Context

The goal of this stage is to describe the need for communication rather than the recommendations for the services that would allow this communication. The specification table expresses the aims of the system to be designed and the context in which it will operate, e.g. the work to be performed and the characteristics of the users and the their organisation. The usage context description should include the following parameters:

Users: more than one, simultaneous, either gender, adult, at least average intelligence, sharing European cultural norms, working in the area of holiday industry and/or document production, experienced in the use of interactive computer work systems.

Work goal: to have a multimedia document which allows holiday makers to select a suitable holiday.

Work domain: transformations of control information (e.g. comments, plans),transformations of document elements to form a complete document (i.e. assembly and (some) editing of pre-existing elements into a document, but no creation of those elements).

Performance parameters: task quality, human and machine resource costs (i.e. time, number of errors, effort, cognitive demands,...)

System: could include all possible IBC services, in any combinations, operating in standard office environments.

Organisation: a large travel agency or tour operator,physically distributed, with democratic management style, and hierarchic management (one person has final responsibility for the document).

Service Environment: multimedia videotelephony, and a multi media document database. These service environment assumptions are provided at the application level analysis and should include the Reference for User Services diagram shown in Figure 1 above.

3.4 Stage 2: The Goal Task Analysis

The aim of this stage is to produce a description of the goal tasks the user wants to carry out. The result of this goal analysis stage of the GUIDANCE Design Method is a breakdown of the major goal of the work system (ie human end–users, and other communicating entities such as machines, databases, terminals etc) into goal task hierarchy, and a description of the objects that are transformed or created in performing these tasks. It is important that at this stage the task descriptions remain technology independent, i.e. the task breakdown should, in principle, be implementable on any available technological or service configurations. Decisions concerning technology are taken when specifying the enabling states for each goal task.

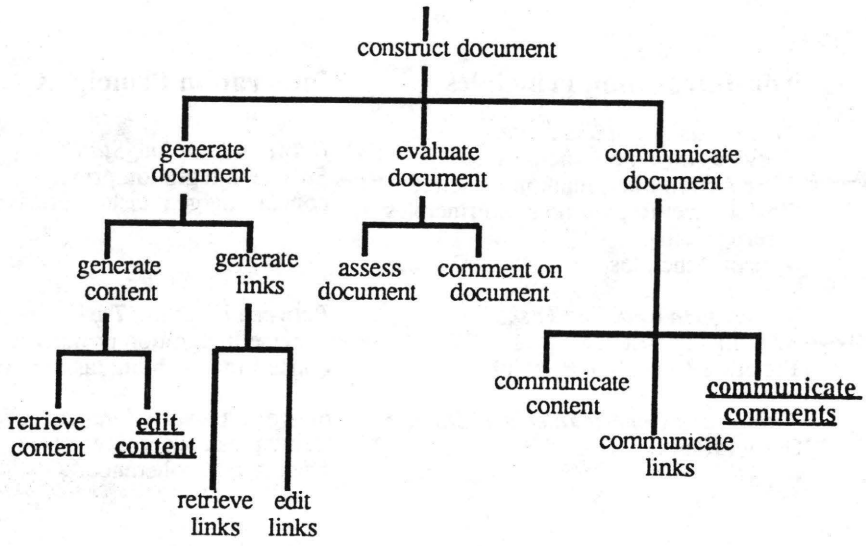


Figure 8: Goal Hierarchy for high level goal tasks 'Construct Document'.

There is no goal task *Create Link/Content* shown in Figure 8 because, for the sake of simplicity, we have made the assumption that the authors are not concerned with creating the document data (e.g. filming the video clips), but merely in their collation and editing. Each of the highlighted goal tasks above (i.e. those important to our simplified example) is described with a specification table. These tables contain the information that describes what should be achieved by the task, with any constraints that might affect the way in which it is done.

3.5 Stage 3: The Enabling States Analysis

An enabling state describes the state of the work system: what knowledge and processing skills the *Users* must have; the characteristics of the *service relationship*, and the state that the *objects* must be in. Each of two goal tasks highlighted above must have an enabling state. This is the first point where design principles can be applied. The designer must take four steps to describe an enabling state:

1) Describe the states of the **users** for the goal task, and apply any *User principles* that are appropriate
2) Specify the characteristics of the **service relationship** for each goal task, and apply any *Service Attribute principles*, *User Action Interpretation principles* or *Task Object Representation principles* that are appropriate
3) Describe the states of the **objects** for each goal task, and apply any *Object principles* that are appropriate
4) Consider any interleaving of goal tasks, apply any *service integration principles* that are appropriate, and then provide an integrated enabling states description for the interleaved tasks.

Step 1: Describe the User for Each Goal Task
The purpose of this step is to describe the **required** user characteristics for a given enabling state. If there is a discrepancy between the required characteristics described by the enabling states description and the **actual** user characteristics then a principle can be applied which suggest a solution to this problem. Example principles are given below. This step also describes the required enabling tasks that produce the enabling state. These enabling tasks are then described in Table 5 (see Section 3.6). Table 1 gives an example of a user description.

The User Description
User Description
The user has to know how to start and end communication
The user has to know how the recipient will receive the image of the user
The user has to be able to change between services
The user has to be able to determine and control quality of the communication
The user has to be aware of incoming calls
User Principles:
Compatibility :
IF the *task* is to communicate with other persons in the workgroup
AND *users* already have knowledge of team working in an open plan office
AND the *service environment* supports graphical display of objects
AND the *performance* should minimise users' affective costs, e.g. frustration
THEN *Apply principle of compatibility:*
The user should be made aware of all members of the workgroup, their presence/absence and their relationships
BECAUSE *(rationale)* This will allow users to exploit existing knowledge about office work, and the work environment with respect to the availability of persons and the actions which may be performed on these persons.
Enabling Tasks for the User Description
Start/end call
Obtain/adjust selfview
Change service
Control quality of the communication
Call alert
Allow hold, resume and forward call
References: Enabling Tasks and Goal Actions, which are included in the GIOs

Table 1: An example User Description for the Enabling State *Comments Communicable*

Step 2: Specify the characteristics of the service relationship for each goal task

Once the IBC designer has described the user the next step is to describe the service relationships for each of the goal tasks identified in the scenario description. The application of principles will contribute to the designer's overall rationale for the design of these enabling states, and the principles themselves are described with respect to this rationale in the service relationship description. When specifying the service relationship the designer will also access non-integration principles concerning service attributes, user action interpretation and task object representations appropriate to this service relationship (see Figure 7). Table 2 gives an example of a service relationship description.

Enabling State 1 - Comments Communicable
Description: The user can communicate with a recipient directly via a face-to-face videophone.
Source: Communicate comments
Principles applied to this rationale:
Service Relationship: uninterpreted/unrepresented
Service Attributes:
Involved Entities: user, workstation, recipient
Communication Type: Human-Human Conversation
Channel characteristics:
Information Content: Head and shoulders view of the participants and their voices are transmitted
Service Components: Audio and video service components are required. The audio component must be mandatory, the video service component may be optional
Synchronization requirements: Lip synchronization is required for the combination of audio and video service components
Format Preservation: Not required for audio and video service components, compression acceptable
Quality of Audio Service Component: Low quality. i.e. comparable to standard telephone
Quality of Video Service Component: Standard video quality (4:3 aspect ratio, 625 lines, 120 000 pixel, 20-40 Mbit/s peak bit rate) comparable with NTSC, PAL or SECAM standard resolution TV quality
Quality of Service: Adequate quality of loudness, brightness, contrast.
Service Attribute Principles:
IF audio and moving images are provided simultaneously (videotelephony conversation)
THEN audio transfer rate is <64 Kbit/s and standard moving image quality is required.

IF audio only is provided
THEN quality is always voice (3.1 - 7 kHz)

Enabling Tasks for the Service Relationship
Establish synchronous transmission
Establish service compression
Establish low quality audio
Establish standard quality video
Determine quality of service

References: Enabling Tasks and Goal Actions, which are included in the GIOs

Table 2: An Example Service Relationship Description for the Enabling State *Comments Communicable*

Step 3 - Describe The Object for Each Goal Task

In common with the user description discussed above, the purpose of this step is to describe the **required** object characteristics for a given enabling state. If there is a discrepancy between the required characteristics described by the enabling states description and the **actual** object characteristics then a principle can be applied which suggest a solution to this problem. Example principles are given below. This step also describes the required enabling tasks that produce the enabling state. These enabling tasks are then described in Table 5 (see Section 3.6). Table 3 gives an example object description.

The Object Description

Object (Document)
The object has to be indicated
The object's availability (non-availability) has to be indicated
Object attributes should be changeable
Object version should be changeable
Object Principles:
Controllability :
IF the task is to edit existing document objects
AND users can specify attributes of existing objects, and already have knowledge of team working in an open plan office
AND the service environment supports graphical display of objects
AND the performance should minimise users' affective costs, e.g. frustration, annoyance
THEN Apply principle of controllability:
only allow users to change essential attributes (e.g. privacy) of objects which they have created or have been given control of by another author
BECAUSE (rationale) editing of objects for users engaged in collaborative tasks should be a negotiated or managed process.
Enabling Tasks for the Object
Indicate object
Indicate availability of object e.g. busy dialogue
Change object attributes, e.g.privacy
Version control
References: Enabling Tasks and Goal Actions, which are included in the GIOs

Table 3: An Example Object Description for the Enabling State *Contents Editable*

Step 4: The Interleaving of goal tasks and the Integrated Enabling States Description

Whilst describing an enabling states designers will need to consider what types of technology are available and appropriate for the goal tasks identified. This information is available from the application level description (i.e. from the customers perspective). This information becomes particularly pertinent when there is potential interaction between goal tasks, and hence between services. The service environments for the two goal tasks of communicate comments and edit contents are:

- a real time, point to point, videotelephony service
- a multimedia document object database and multimedia editing tools

With the two goal tasks of communicate comments and edit contents there are two interleaving relationships important to our simplified example:

- Users want to discuss the editing of a video object whilst carrying out these editing changes in real time (i.e. interleaving between editing and communication tasks).
- Both users want to edit the same video object (i.e. interleaving between editing tasks).

For each of these two possible goal task relationships there are service integration principles that influence how the IBC designer describes their enabling states. Two examples of these integration principles are given below. The information for the service environment descriptions included in these principles is drawn from the Reference for User Services description described in the analysis of usage context.

IF the task involves human to human communication and human to machine conversation performed congruently
AND there are at least two users
AND the service environment supports audio and video communication and includes a multimedia editors
AND the performance depends on both tasks being carried out simultaneously
THEN Apply principles of salience:
ensure that the primary service component does not interfere with the secondary service component (e.g. mute a non-primary video service component or mute a non-primary audio component).

BECAUSE (rationale) machine should assist users in focusing on primary tasks and allow use other input/output modalities for secondary tasks (e.g. when editing a video document users can utilise a free audio channel for conversational tasks).

IF the *task* involves human to machine conversation and human to machine conversation (e.g. joint editing)

AND there are at least two *users*

AND the *service environment* includes a multimedia object database and multimedia editors

AND the *performance* should be quick and error free

THEN *Apply principle of controllability:*
allow only one user to edit the object, and allow the editing user to specify whether his/her actions will be visible on the workstation of the second user

BECAUSE *(rationale)* users engaged in collaborative tasks should be able to control what is being presented to other users to ensure privacy

Integration principles, then, provide usability recommendations to the designer which will influence the design of the enabling states for the two goal tasks of communication and editing which have been identified as potentially interleaving. Next the designer must design an integrated enabling state to describe the interleaved tasks.

If two or more subtasks are to be allowed to interleave then the integration of their enabling states must be designed. This means that the two enabling states must be combined and a third *integrated* enabling state description produced. This description will allow any modifications to the individual enabling states descriptions to be highlighted and any influences on software design to be considered. Table 4 gives an example of a integrated enabling states description.

Integrated Enabling State: Communicate Using Videotelephony and Edit Document
Description: The user can communicate with a recipient directly via a face to face videophone call and retrieve and jointly edit multimedia documents simultaneously
Sources: Comments Communicable and Contents Editable (Enabling States 1 and 2)
Description of the User
See Enabling States 1 and 2
Integrated Service Relationship :
Interactions and Modifications to Individual Service Relationships: There are two service relationships in this enabling states description. Multimedia objects are presented simultaneously with the presentation of video images of the recipient. As simultaneous, multiple video outputs will distract the users from their primary tasks one of the video outputs is set to pause. Simultaneous audio conversation is permitted.
Simultaneous joint editing is not permitted, but co-workers can hand over control of the editing tool. Hand-over should be negotiated and controlled by user currently using the tool. The visibility of user actions to the co-worker is allowed, but should be controlled by user currently using the tool.
Description of the Objects
There are two objects: People and Document objects. See Enabling States 1 and 2.
Enabling Tasks for the Integrated Service Relationship
Inform users of machine actions (e.g. video freeze)
Allow user override of machine actions if desired
Make editing actions visible/invisible to co-worker
Hand over editing to co-worker
References: Enabling Tasks and Goal Actions, which are included in the GIOs

Table 4: An Integrated Enabling State for *Communicate Using Videotelephony* and *Edit Document*

3.6 Stage 4: The Enabling Task Analysis

In describing the users, the service relationship and the objects the designer will have identified a number of enabling tasks which have to be carried out in order for users to reach the appropriate enabling states for

their goal tasks. Stage 4 of the GUIDANCE design method describes these enabling tasks and applies principles which help the designer meet the second usability criterion and reduce to a minimum the costs to the user in reaching the appropriate enabling states for their goal tasks. The principles applied at this stage are concerned with user costs and function allocation (see Section 2.4).

As with goal tasks, there may be an interleaving between enabling tasks, and the designer may refer to integration principles for assistance in the specification of the respective GIOs. For example, in this simple case history there is a potential interleaving between the enabling tasks of 'adjust self view' and 'place object at convenient place on screen'. Cognitive costs (e.g. workload) to the user because of this potential interleaving can be alleviated by allocating to the machine the task of placing the object conveniently so that there are no overlaps between the videophone window and the window containing the object. Enabling tasks integration principles assist the designer in making decisions of this type.

Table 5 gives an example of an enabling tasks description identified during the enabling states analysis are given below. 'Identify recipient' is an example of an enabling task allocated only to the machine, and contains a function allocation principle which makes this recommendation.

Enabling Task : Identify Recipient
Description: This enabling task relates to the user description of the Enabling State *Comments Communicable* and supports automatic dialling of recipients for face to face communication.
Class Relations:None
Usage Aspects:
Allocation: machine
Allocation Principles:
Simplicity:
IF the task is to identify the recipient of a call
AND users are limited in number and have knowledge of other users and their roles
AND the service environment has knowledge concerning the availability of recipients and their locations
AND the performance should minimise users' cognitive and physical costs
THEN Apply principle of simplicity:
design the user interface to support the rapid selection of the limited set of user numbers (addresses) without the use of a dialling keypad
BECAUSE (rationale) the restricted number of communicants makes number dialling unnecessary
Design Aspects
Functional Requirements
The user should be able to be perform a call by clicking on a person icon and the RTCS icon in any order.
Operational Requirements
Machine dialogues should confirm status of call and service attributes
References: GIOs Person Icons, RTCS Icon, Call Confirmation Dialogue

Table 5: An Example of an Enabling Tasks Description for 'Identify Recipient'

4.0 Software Design Using the Enabling States Approach

The completion of the design stages so far results in a complete description of the tasks which must be performed by the work system, including those tasks which are not directly related to the overall goal and which may vary from design to design. All design decisions that result from the user requirements have now been made. The next step is to begin the specification of the work system software.

The final layer in the GUIDANCE Design Method produces specifications of the appearance of the Generic Interface Objects (GIOs) required to satisfy the operational requirements described in the previous layer, together with the actions that they respond to, and their behaviour. This is done largely by collecting together the information specified in the earlier tables. Where this information is not sufficient to identify one particular design option or feature of a GIO, then either that feature is not directly relevant to the usability of the system or no definitive advice can be derived from the user requirements and principles. In

this case the GIO specification table will reflect the design decisions made by the software implementation team.

In the specification method GIOs which are to be used in reaching a particular enabling state are clustered into a *View* for that state, together with the objects which must be present to provide the service relationship defined in that enabling state. All interface objects that are likely to be used together (or in close sequence) and the consequent interaction behaviours are gathered together.

Figure 9 shows the how the design principles that were used in the specification of enabling states and enabling tasks influence the design of GIOs, and shows the relevant parameters of the GIO specification table.

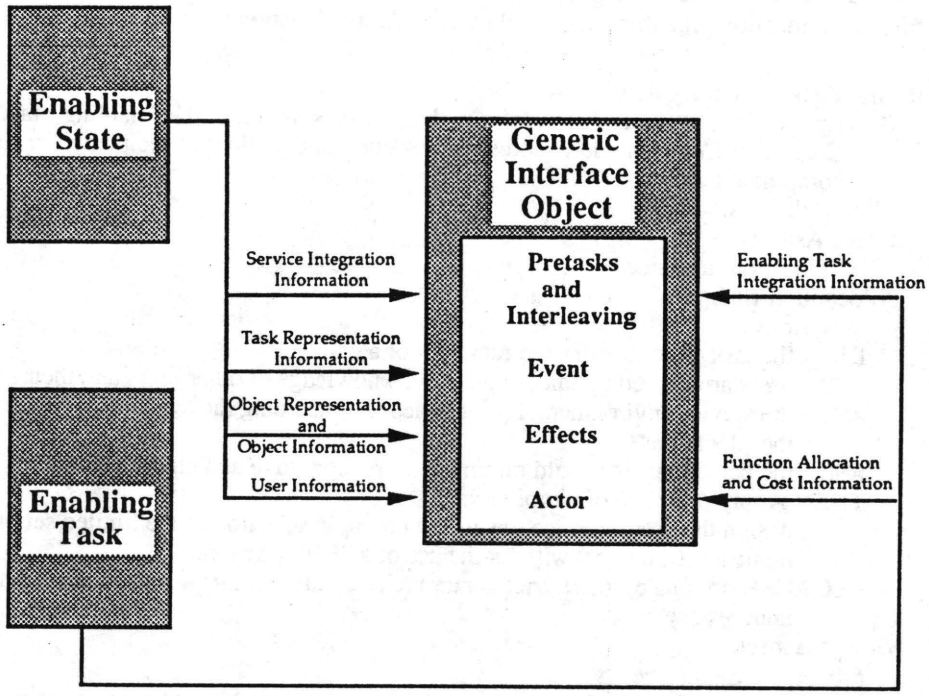


Figure 9: The flow of information derived from the application of principles into the design of GIOs

GIOs which are used within different services may interact if these services are used simultaneously, and service integration principles, therefore, provide information about task interleaving and the usability of different services which may affect GIO design. Similarly, enabling task integration principles serve the same function with respect to the enabling tasks the user has to perform on a given set of GIOs when using simultaneous services.

Task action interpretation principles help the designer in representing task actions on a given GIO and are applied to the 'event' slot in the GIO specification table. For example, principles might recommend different cursor shapes to indicate appropriate task actions. Task object representation principles and object principles influence the design of the 'effect' slot which describes how objects are represented. This may concern, for example, icon and dialogue design. User principles, function allocation principles and cost principles apply to the 'actor' slot in the specification table and make recommendations concerning which communicating entity should perform a particular action and why. The software implementation team may use software libraries to ensure consistency between GIO components such as widgets.

In summary, the use of principles throughout the design process encourages designers to consider issues of service integration and service usability by pointing out pitfalls that a designer might fall into and guides them towards options that have empirical evidence as to their usability.

5.0 Discussion: Principles Result in Integrated Services

The classification of design principles within the enabling states design method provides an integrated approach to IBC design which encourages designers to think about the design of systems from the users' perspective and provides support to IBC designers in solving the problem of IBC service integration and service usability.

The close linking of the principles and the design method offers the designer assistance in identifying the appropriate principles for a given stage in the design process, and provides a schema which can support the development of principles. Future work will integrate usability guidelines from other RACE projects and our own empirical work into the schema. Thus, principles can be grouped according to their role within the IBC development process, with resulting improvements in applicability and availability of design information. By structuring the design sequence so that relevant information is made available at the appropriate point, so that related information is kept together within tables, and so that tables follow a common schema, the task of design is made more manageable, and the likelihood that the designers will be able to make use of the information that is available to them is increased. Communication between individual designers and separate design teams is enhanced by the simple formalism provided by the specification tables, and so groups working on individual services can have more confidence that their designs will remain integrated. Finally, the classification schema encourages designers to use the principles as a checklist or rationale for the design decisions they have made.

Acknowledgements

This work was carried out as part of the R1067 GUIDANCE project supported by the Commission of the European Communities under the RACE programme. The project partners are: Standard Elektrik Lorenz AG, Germany; University College London, UK; British Telecom, UK; Roke Manor Research, UK; and Telia Research A.B., Sweden. We would like to thank our colleagues who commented on earlier drafts of the paper.

References

Byerley P., & Bruins, R,. Conceptual Framework for Usage of Telecommunication Services. In: P. Byerley and S. Connell. (Eds). *Integrated Broadband Communications:Views from RACE - Usage Aspects*, North-Holland Studies in Telecommunication Volume 18. Elsevier.

Denley, I. , Whitefield, A., Byerley, P., Voigt, U.B., Hermann, S., and May, J. (1992). Design Principles for Improving Service Integration for End-Users in Broadband Communication Systems. In: *Proceedings HCI '92 Conference.* University of York. 15-18th September 1992.

Esgate, A., Whitefield, A. and Life, A. (1990). Developing Usability Integration Principles for the Design of IBCN Systems. In: *ECCE-5, Proceedings of the Fifth European Conference on Cognitive Ergonomics*, Urbino, Italy, September 3-6th, 1990.

Guidance Position Paper, Issue B, 1992. Commission of European Communities. Brussels.

Terrins-Rudge, D., Adamson, S.J., Ionescu, R., Peterson, M. and Hedman, L. (1992). Designing for Designers' Needs. In: P. Byerley and S. Connell. (Eds)*Integrated Broadband Communications:Views from RACE - Usage Aspects* North-Holland Studies in Telecommunication Volume 18. Elsevier.

Whitefield A.D., Byerley, P., Denley, I., Esgate, A., and May, J. (1992). Integration of Services for human end-users (1): Design principles, enabling states analysis and a design method. In: P. Byerley and S. Connell. (Eds). *Integrated Broadband Communications:Views from RACE - Usage Aspects* North-Holland Studies in Telecommunication Volume 18. Elsevier.

COOPERATIVE PROBLEM SOLVING, AS REFLECTED IN AN EXPLORATION OF SOME MECHANICAL DESIGN ENGINEERING PROJECTS

Yvonne Wærn and Karl-Gustaf Wærn

Department of Psychology, Stockholm University
Sweden

Abstract

In this paper, which summarizes a preliminary study in an ongoing research project, we look at cooperation as demonstrated in the process of performing mechanical design engineering projects. This is a work situation in which people, often with knowledge and competence from separate technological fields, are required to cooperate. We give a background for how such cooperation could be described and we present a frame of reference for this purpose. The theoretical bases are derived from general problem solving theory and from general coordination theory. The frame of reference is used in describing empirical data obtained in an interview study of five projects in mechanical design engineering. Finally, we present some points for discussion, tentatively pertaining to computer supported cooperative work.

1. Introduction and aim of the paper

The increasing interest in designing computer support for cooperative work has led to a corresponding profusion of interest in cooperative work as such. Cooperation is, as everyone knows, essential in performing most of what can be termed professional work. At the same time, it may be observed that cooperation sometimes may be difficult to achieve. So, the question arises as to how people who are engaged in a common work project manage the given - or, as it were, emerging - needs for cognitive cooperation.

In this study (which forms a more or less preliminary part of an intended, longer research project) we look at cooperation as demonstrated in the process of performing mechanical design engineering projects. The aim of the study was to establish a frame of reference, or taxonomy, that could be used for the description and analysis of design engineering as a group problem solving process, possibly with some - future - perspective on using information technology as a cooperative tool (although computer assistance is not a main theme in this study).

The paper contains a description of the theoretical considerations, the frame of reference and also some summaries of findings in a small empirical investigation.

2. Theoretical bases and the frame of reference

The theoretical bases take their point of departure from descriptions of design engineering, from a general problem solving theory and from a developing theory of coordination.

2.1. Design engineering as a cognitive process

The design engineering process has been studied by several researchers, mostly from technological, more or less normative, aspects. Some of these are mainly interested in the organisational or economical aspects, others (and it is these that interest us in this context) are concerned with the question "What is the job of the design engineer?" (e.g. Pahl & Beitz, 1984, and Suh, 1988). Although the researchers differ from each other as to terms, models and specific points of view, consensus prevails as to the basic properties of the design process, which perhaps could be summarized in the following general points.

1. Design engineering is a process involving problem solving. It proceeds from receiving initial information (requirement specification) to the final realisation of a, more or less complex, technical object that is possible to manufacture and sell.

2. The process proceeds through several stages, or phases, e.g. an oscillation between "analysis" and "synthesis" (Suh, 1988). As such, the process is often cyclical, i.e. it involves iterations, in that a certain moment or stage may have to be repeated until a satisfactory solution is reached.

3. The process is also one of "knowledge turnover": The design engineer has to apply his professional knowledge to the problem at hand, but the resulting product provides, in its turn, a source for developing new knowledge.

4. More often than not the design is a product of several design engineers - with various kinds of knowledge and experience - working together in larger or smaller groups.

These very simple and general observations, which, in fact, can be readily made by anyone who is even superficially acquainted with engineering, thus form the outset of our approach. (It should be noted that our purpose with the following considerations is not to add another theoretical description of the design engineering process per se!)

2.2 Problem solving theory

A purely descriptive approach to the study of the design engineering process is presented by David Ullman and his associates (Ullman, Dietterich & Stauffer, 1988). What makes this approach interesting to us is that it is very closely related to (and inspired by) cognitive problem solving theory. Ullman has studied design engineers' problem solving by means of "think-aloud"-technique while they were solving experimental design problems. Ullman´s set of analysis concepts was based on Newell & Simon's account of Human Problem Solving (Newell & Simon, 1972). Briefly, the latter theory assumes that problem solving can be described as a search in a problem space. This means that, first, the problem has to be understood, which equals "creating a problem space". Then, alternative ways of moving around in the problem space (i.e. performing different operations) are tested, which is equivalent to "searching the problem space". The search presents difficulties when the problem space cannot be surveyed. In such a case, different operations have to be performed, without knowing their effects, the results of these have to be assessed and further operations have to be chosen on basis of the effects. Ullman's analysis attempts at describing the mechanical design process as such (at the individual level) in problem solving terms.

In order to assess and in some way emphasize the design problem solving as a sequential process over time, we found it, for our purpose, potentially useful to add a few "steps" to the two step "create/search" model (Table 1).

Table 1. A Model of Individual Problem Solving

Phase	Processes
Orientation towards relevant restrictions	Envisage goal, Envisage possibilities information Prerequisites for finding problem spaces
Creation of problem space	Describe initial situation Envisage operations which can change states Describe current goal
Search in problem space	Testing of alternative operations/methods solutions Judgment of effects Decision on operation
Re-creation of problem space	When one problem space does not seem to lead towards the goal, other problem spaces are sought. New descriptions of situation, goal and operations are thereby attempted.
Finishing	Compare result to goal Test result

2.3 Coordination theory

It should be noted that the description in Table 1 applies to individual problem solving. For our purpose we have, in addition, to consider how cooperation during problem solving can be described. Here is where "coordination theory" enters into the picture.

Coordination theory is concerned with prerequisites for, modes for, and results of cooperative work in general. We have here used concepts from the (proposed) outlines of coordination theory recently developed by Thomas Malone (Malone & Crowston, 1990, 1991). These authors suggest that cooperation can be described in terms of four levels:

1: Coordination. At this level, the actors select a goal and decide who will perform which kind of activity (definition of common goal; task and resource allocation).

2: Decision making. At this level the members of the group decide on the common goals to be achieved, the alternatives being considered, the evaluations of these alternatives and the choices that are made.

3: Communication. Here the actors have to establish a common language and transport their messages to each other in this language.

4: Common Object. In order to establish the common language the actors have to perceive common objects, such as physical objects in a shared situation, or - as it were - information in a shared database.

The essential trait of this model is that the "levels" are inclusive: the performance at any one level requires that adequate instances of performance at the "lower" levels are at hand (analogous to abstraction levels in other systems, such as protocol layers for network communications). This means, thus, that coordination requires group decision making, that group decision making requires communication, and that communication requires common objects.

Social science researchers have often found that people experience difficulties in cooperating, with consequent poor performance of the group as a whole. The performance of a group is sometimes not as efficient as is the sum of the individual participants' performance (see e.g. Forsyth, 1990). The causes of such a deterioration of performance can be manifold. Some are related to group dynamic processes, such as group pressure and social loafing. Other causes may be related to such difficulties in cognitive processes that could be accounted for by referring to coordination theory. Some considerations:

At the most complex level, coordination, group participants may find it difficult to coordinate their own individual cognitive goals into a common group goal. The result is coordination problems in allocating tasks and agreeing upon how to use scarce resources (in time and, perhaps, money). At the next "lower" level, common decision making, studies in the field of small-group social psychology have shown that people have difficulties in agreeing upon alternatives and arriving at a joint decision. What purely cognitive factors are involved here is, however, not easily determined (motivational and group dynamic factors problably play a very important part in group decision making). At the communication level, there may be difficulties in establishing the necessary common language. Other difficulties related to communication may concern current "information overflow". In almost any kind of workplace today, there is a continuous flow of papers and discussions. It could be that the participants of a working group do not know how to sort out the information that is relevant to their work; similarly, it could be that when they, in their turn, have to inform others, their information "drowns". At the common object level, finally, difficulties may arise in such cases as when the object is vaguely defined at the outset. In extreme cases, the "common object" can even be lacking from the beginning, so there will be nothing to communicate on.

2.4 A frame of reference for studying cooperative problem solving in design engineering

Using the "dimensions" accounted for above, we created a provisional assessment model in order to obtain a reasonable and manageable structure for the analysis of - the quite voluminous - information collected in the interview study to be described below. This was done by cross-classifying the problem-solving phases and the coordination levels, as represented in Table 2. In this table, we have added an individual level in order to cover the whole design process.

Table 2. A frame of reference for cooperative problem solving

Cooperation level		Problem	solving	phase	
	Orientation	Creation of problem space	Searching in problem space	Re-creation of problem space	Finish
Coordination	Coordinate comprehension of: Goals Restrictions Possibilities	Define over-all goal Find methods Allocate tasks	Allocate tasks Coordinate search	Revise concept for group	Find that total goal is achieved
Common decision making	Decide alternative goals Decide valid restrictions/ possibilities	Decide on problem space/ design concept/	Decide method/ evaluate outcome/decide next step	Decide whether or not to re-create concept Decide what might be changed	Decide that a problem is solved satisfactorily
Communication	Discuss understanding of functional goal/restrictions/possibilities	Discuss problem space/ design concept Seek/exchange information on physical goal/restrictions/possibilities	Discuss design details	Discuss alternative concepts	Discuss if product meets requirements
Perception of common object	Description of the functional goal	Description of the design concept´s main traits	Description of alternative solutions, methods physical object	Description of alternative concepts/previous designs	Perception of the end result/ object
.....	.....	.....	.....	.....	.....
Individual	Obtain basic requirement information	Find design concept for task/subtask	Test different detail solutions	Change individual problem space/ concepts	Find that individual task goal is reached

3. The interview study

Five mechanical design engineering projects in different fields were studied. Each project involved two to five design engineers, mostly in the mechanical field but also in electronics. The projects varied as to scope and complexity, but were all finished or nearly so. The projects were chosen so that a core group of people was engaged in the project, and so that it was possible to conceive the project as a whole (albeit it might be a part of a still bigger whole). The objects to be designed were: a hull for a missile sight device, a tank mine, an impregnation barrel for high-voltage cable, a vegetable cutting machine for catering service, and an automatic device for tightening and sealing acetylene gas flasks.

All participants in the projects' "core" design engineering groups were interviewed by means of semi-structured interviews, where they were asked to describe the course of the project.

The interviews were taped and written down in extenso, from which excerpts (around 230) were made and classified into the "cells" of Table 2. The analysis showed that this frame of reference could be used to describe most of the contents of the interviews, although a few "cells" of the cross-classification remained empty (but not contraindicated).

In this paper we will give an account of the results which pertain to the frame of reference suggested. Other types of results will be covered in another paper. Due to ethical reasons, we cannot identify which comments and experiences that refer to which project. We have chosen the sub-headings according to the "problem solving dimension". For space reasons, the five "cooperation levels" are referred to in continuous text, using italics: *coordination*, *decision*, *communication*, *common object*, *individual level*.

3.1 Orientation

This is the "briefing", where design project participants get the necessary basic information for starting their work. Here the requirement specification plays an important role (although it differed very much in scope and detail between the projects). As to *coordination* in this step, it seems to serve the purpose of getting the project members aware of the functional prerequisites for the whole project. The *decisions* in orientation are - in our material, which entirely refers to customer-ordered design projects - mostly taken by the customer's representative. Also *communication* takes to a great extent place with the customer. The *common object* seems to be the "req-spec". *Individually* we have some instances where a designer felt a need for obtaining additional information on his own.

3.2 Creating the problem space

Here is where a general basic concept for the design solution is created. In our projects, it seemed quite common that one single person (the project leader, the head design engineer, or even the customer representative) played a central role. In *coordination*, the theme was generally one of assigning sub-problems to the group members according to their competence and their interests. The project leader was, of course, central, but also group members had their saying. At the *decision level* no explicit indications were found; it seems plausible that coordination and decision making could merge in this phase.

Communication in problem space creation was evidently essential; rather strong expressions are sometimes used by the interviewees: "brainstorming", "hellofa fiddling around", etc. The *common object* is probably mostly still the "req-spec", or, perhaps, some already existing earlier designs. Our information shows quite clearly that even if the common concept is clear, it leaves a lot of problem space creating left for the *individual* design engineer. "You know what the thing should do, what it has to perform, but you don´t know what it can look like" is a prevailing comment.

3.3 Searching in the problem space

During this phase the thinking work proceeds - for the group as well as for the individual design engineers - with the aim of detailing the design object(s). As to *coordination*, the main task seems to concern a) observing that time limitations and deadlines are held, and b) seeing that organisational and other external circumstances are met with. *Decisions* seem, in our material, mainly to concern priorities for the suggested solutions, i.e. checking off procedures against results. The decisions are made partly by the project leader, partly by means of group consensus, depending on the importance level. Some decisions are made by the customer (suggestions from the project group may have to be approved by him, even in detail). There is no doubt that *communication* is very intense in the problem space search, most so within the group, but also between the group and external agents, such as representatives for manufacturing departments. Shapes of details, materials, physical interfaces between objects and so on are put under debate. Most of the communication is informal (which the interviewees generally considered the most rewarding), but also more formal meetings occur. In what could be termed search, few *common objects* are mentioned, except for some cases in which two design engineers work with parts that have to fit together mechanically. *Individual problem space search* is not frequently accounted for, as the interviews were not directed towards individual problem solving.

3.4 Re-creation of problem space

In problem solving it is inevitable to encounter "dead-ends". Sometimes it is possible to proceed within the same problem space, through getting back to an earlier cognitive step. Sometimes, however, the whole problem space has to be reconstructed, i.e. "re-created". This could be expected to occur in complex tasks such as design engineering. According to our material, such re-creation of the whole design project is rare at the *coordination* level (preliminary project studies and pre-prototyping are activities intended for preventing just that). But needs for re-creation involving coordination may arise due to changes in external requirements or to new information coming up during the work. Also here, time and cost seem to be important parameters. Our material gives scarce information on *common decisions* in this respect; decisions are perhaps mostly taken by project leaders or executives at higher levels. *Communication* may be intense in re-creation cases, but it is in our material difficult to distinguish between "group" and "individual" backtracks. The methods for solutions seem to be two: meetings as needed (sometimes referred to as "storm-meetings" or "disaster meetings") and seeking advice outside the group. No excerpts contain anything that could be classified solely at the *common object* level (quite understandably). The most frequent excerpts concerning problem space re-creation are those that can be referred to the *individual* level. "Doing the thing over again", "rethink", "go back to the beginning" are typical wordings. But some designers also emphasize the necessity of preventing re-creation needs by careful planning at the outset, trying to foresee what can happen. Time plays a certain role here; type quotation : " It´s important that you get enough time in the beginning to think it through and try to imagine all possible problems that may pop up. For later, it may be almost impossible to change yourself. If once you have chosen a concept and worked on it that much, it´s difficult to change, you´re locked up."

3.5 Finishing the problem solving

As to *coordination*, our material only indicates that it is confined to the project leader (or design engineering executive) judging that some project members have done their job and assigning them to other tasks. *Decisions* are - for our projects - mostly taken outside the group, by the customer or by the company board. Project group *communication* in the finishing phase is not mentioned in the excerpts. The *common object* is the designed product, or prototype: "When you see the thing, and see that it works, then you know it´s finished." As to the *individual* level it seems clear that the design engineering process consists not of one but of several problem spaces, since everyone has several tasks to perform during the project. Criteria for the judgement of when a part problem is solved differ according to the design engineer´s task and technological field. It is, however, rather striking that several interviewees hint that one, technologically, in fact never is finished with a solution; there are always details that could be made better. But time and circumstances control: "there´s always a deadline", "when time is out and there are just a few small and unimportant details left that somebody else could do as well".

4. Some points for discussion

As has been pointed out above, our study was a preliminary one, and the results can, of course, not be taken as the basis of any attempts for generalisation. (For one thing, the "one-point" interviews give insufficient information on temporal sequences of the design process.) It seems, however, that our frame of reference was suitable for organizing data into a picture where strengths as well as weaknesses of cooperation during design problem solving are indicated. So, for instance, the requirements specifications may be crucial, (essential as they should be, as they constitute the initial "common object") and cause trouble, or time delay, if they are too vague and proving not helpful for finding design concepts or for testing the final design. On the other hand, too detailed specifications may be experienced as constraining to the design group creativity.

That communication, within the group, or with outside agents of different kinds, is intense and important in problem space creation and search is clearly demonstrated (as could be expected). The exchange of knowledge with other people may very well be the essence of design engineering projects. Communication, however, is many-faceted, and formalized communication involving many people may be experienced as inefficient by the project group participants.

Our investigation has not been directed towards computer based cooperation tools. It is, though, interesting to note that no interviewees mentioned the use of CAD as a cooperative medium, although most of them used CAD for their individual work. (One mentioned the advantage of mailing CAD diskettes to the customer.) Further exploration is needed in order to see the cause of disregard of this potential for a common object, and for communication.

For the formal decisions which have to take place, formal decision making support might be tested, i.a. since it was found that traditional formal meetings sometimes were considered inefficient.

Communication media were not relevant in these situations, since almost all designers worked physically close to each other. However, some comments about the difficulty of getting information in time from other departments (such as manufacturing or tool design)

indicate that physical distance (as well as differences in expertise) might affect the possibility to cooperate.

Computerized systems for group coordination do not seem to be useful in projects such as the ones studied here, where only a few participants are involved, and where there usually is a closely and immediately available project leader to take the responsibility of coordination and managing the project.

Our future work will be concerned with more details regarding communication. Also, more projects will be investigated in order to find factors that can affect the outcome of cooperative problem solving.

Acknowledgement

This research has been supported by grants from the Swedish Work Environment Fund.

References

Forsyth, D.R. (1990). Group Dynamics. Pacific Grove: Brooks/Cole

Malone, T.W. and Crowston, K. (1990). What is Coordination Theory and How Can it Help Design Cooperative Work Systems? In: Proceedings from the CSCW '90 Conference, pp 357-370.

Malone, T.W. and Crowston, K. (1991). Toward an interdisciplinary theory of coordination. Technical Report, Center for Coordination Science, Sloan School of Management, MIT, No 120.

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

Pahl, G. and Beitz, W. (1984). Engineering Design. The Design Council, London.

Suh, N.S. (1988). The Principles of Design. Oxford University Press, Cambridge, Mass.

Ullman, D.G., Dietterich, T.G. and Stauffer,L.A. (1988). A model of the mechanical design process based on empirical data. AI EDAM, 2, pp 33-52.

The Act of Negotiating During Design

Beth Adelson[*], *Troy Jordan*[+]

[*]Department of Psychology, Rutgers University
USA

[+]Sloan Center for Coordination Science, MIT
USA

1 Introduction

Increasingly within organizations the nature of the problems to be solved require the work of teams of individuals with different training, vision and mandates[1]. These intra-group differences lead to identifiable and recurrent sorts of conflicts. These include conflicts over: goals; allocation of resources; roles; and judgements as to what constitutes an optimal solution to the problem at hand[2].

In this paper we argue that conflicts like these, which arise in the context of cooperative work, can successfully and appropriately be resolved through collaborative negotiation. In support of this argument we will first set out a prescriptive theory of collaborative negotiation. Next we will present NegotiationLens, a tool which allows parties involved in conflict resolution to work through the process described by our theory. We will then present two simple cases in which the tool was successfully used. We will conclude by presenting a case study of a large design project. The conflicts which arose in this project were analogous to the conflicts in the simple negotiations and so we will use the case study to argue for the utility of NegotiationLens in complex situations across problem domains.

2 A Theory of Collaborative Negotiation

In this section we present a prescriptive theory of negotiation which we assert can be useful in large collaborative projects like the design project described in Section 4. That is, the theory is intended to move groups, like the collegial but diverse groups typically involved in large projects, away from an impasse and back into a collaborative relationship. This is accomplished by having the groups make explicit their needs and resources and then allowing them to *jointly* construct mutually acceptable solutions. This has the effect of solidifying and improving existing working relationships.

The underpinnings of our theory derive from theoretical and empirical work on negotiation which has accumulated over the last decade (Susskind & Cruikshank, 87; Brockner & Rubin, 85; Sycara, 85 & in press; Simpson, 85; Kolodner, Simpson & Sycara, 85; Pruitt & Rubin, 86; Raiffa, 82; Fisher & Uri, 81; Fisher & Brown, 88; Kolb, 83; Adelson, 91 & 91a; Adelson & Jordan, in press). We begin with a description of the theory. We then describe the tool which grows out of the theory and provide two examples of its use. Because the examples are analogous to the design-domain conflicts described in the case study presented in Section 4, these examples will be used in that section as models for potential resolutions.

1. Making Needs Explicit
   In this part of the negotiation the parties are asked to state what they need in order to reach a successful resolution of the situation. Part of this process includes: i. Making implicit desires explicit. This allows both sides to better understand their own needs and the needs of the other party. ii. Providing explanations as to the importance of

[1]This work was made possible by a Henry Rutgers Research Fellowship from Rutgers University; and by the generous resources provided by Tom Malone at MIT's Center for Coordination Science. We also thank MIT's Muse Consortium for their collegial support.

[2]For a more extensive treatment of intra-group conflicts see Adelson & Jordan in press; Malone & Crowston, 1991; Crowston, 1990.

each need. This allows each party to feel that the other has well-motivated rather than simply arbitrary needs. iii. Developing objective criteria concerning the legitimacy of each need. This will aid both parties in deciding which positions or parts of their positions are reasonable as they work towards a resolution.

2. Making Resources Explicit
Here the parties state what they can offer each other in their collaborative endeavor. This stage makes clear the boundaries of the situation. Additionally, it can make each party feel that the other is making a good-faith effort.

3. Matching Interests to Resources
This is a process by which the parties look for opportunities for mutual gain. But, the discovery of these opportunities has in practice been markedly difficult. However, in the course of developing our software we have developed a way of increasing the likelihood of noticing these opportunities. That is, in our version of the theory the parties systematically compare a given need against the currently listed resources. This process is illustrated in Section 3.1, example 1.

4. Developing Joint Solutions
Here the parties are encouraged to: i. Initially develop a variety of solutions which might accommodate the needs of both sides equally; and ii. Iterate through this process by evaluating emerging solutions until a mutually agreeable one is found. (See point 6 and Section 3, point 5.)

5. Developing Alternatives
In addition to developing possible negotiated solutions the parties are asked to individually examine their alternatives to working together. This serves several functions. When good alternatives to working together do exist, knowing these alternatives can increase the parties confidence in their own resources, allowing them to approach the negotiation with a mind-set which has been found, perhaps counter-intuitively, to foster flexibility (Adelson & Jordan, in press). However, it can also allow the parties to quickly decide that the current collaboration should be abandoned. As mentioned above, the interesting point here is that this kind of determination, when made early on often preserves the collaborative relationship, allowing future joint efforts to succeed.

   In the case where good alternatives are not found, it can increase the parties commitment to the negotiation process, thereby motivating the parties to construct a joint solution.

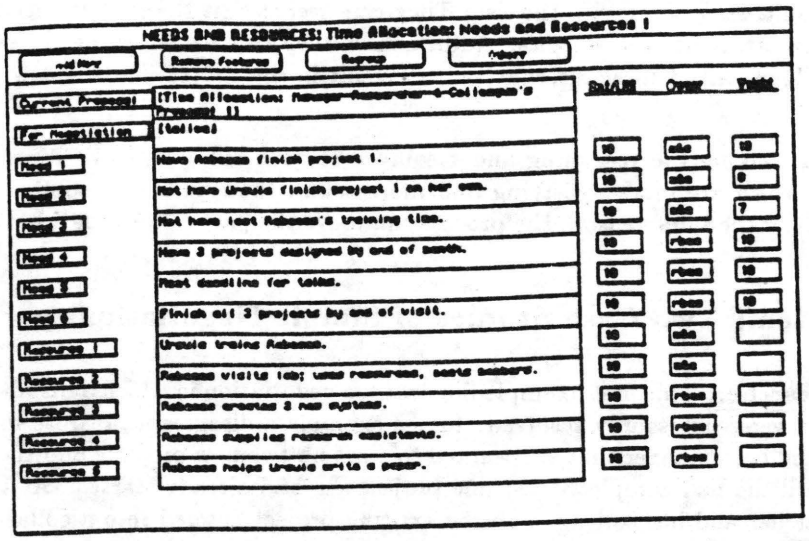
6. Respecting the Other Side
A central part of this theoretical framework is that it asks the parties to commit to respecting each other. Part of this respect entails having the parties to make their needs, resources and rationales explicit. It also takes the form of having the parties treat the needs of the other side as seriously as if they were their own and so manifests itself in the development of mutually beneficial alternatives. Additionally it means the parties not only commit to the negotiation as a whole but also to the agreements which result.

   The example of the visiting researcher, described in Section 3.1 is a good illustration of parties acting with commitment and respect.

3 NegotiationLens: The System Embodying the Theory

We begin by giving the reader an overview of the experience of users when they engage in conflict resolution with NegotiationLens. At the level of the facilities provided for users: NegotiationLens provides a sort of spread sheet for keeping track of the state of the negotiation. It gives the parties appropriate workspaces at appropriate times. Futher, it allows users to make clear dependencies among related pieces information. The tool purposely does not make any decisions for the parties since one motivation of the work is to have the parties develop and internalize negotiation skill.

At a deeper level, in the process of using the tool, the two (or more) parties *jointly* build a solution by considering the needs as well as the resources which they bring to the table. Additionally, they are encouraged to move into collaborative positions because they are allowed to back off from any initial unilaterally proposed solutions as they are prompted through the stages listed here. In this way the system truly does embody the theoretical framework described in the last section. Steps 1 and 2 above map onto step 3 here. Step 4 above maps on to step 4 here and step 5 above maps on to step 5 here.



The Visiting Researcher Case Study (Section 3.1)

Figure 1: Needs and Resources with Decision-Making Information Shown

1. Problem Statement: Both parties begin by separately creating a statement of the problem as they see it.

2. Initial Solution: Both parties separately propose an initial solution which is satisfying to their side. It has been found that this step ultimately allows the parties to move out of inflexible starting positions (Brockner & Rubin, 85; Pruitt & Rubin, 86; Susskind & Cruikshank, 87). A party strongly committed to an initial position frequently needs to be able to state a solution that accommodates that position in order to feel it has been heard and considered.

3. Underlying Needs and Resources: Based on their 'Problem Statement', *rather* than on their 'Initial Solution', the parties now jointly construct a list of what they believe needs to be done and what resources they are willing to contribute in order to obtain the optimal design. Additionally each party can weight the importance of each of the needs they have listed. (The use of the weights is discussed in conjunction with point 5 below.)

4. Collaborating: Matching Needs to Resources The parties then jointly construct a new solution by matching the resources against the needs. As mentioned above, systematic matching is critical both in discovering a successful solution and in increasing a sense of collaboration between the parties. The systematic matching process is facilitated by the system's grouping of needs separately from the resources (But see footnote 3).

   It is the matching process that uncovers the previously non-obvious ways in which the parties can help each other (see Section 3.1).

5. Iterating:
   If the parties feel dissatisfied with the newly developed solution, NegotiationLens provides them with a means for understanding the source(s) of their dissatisfaction. The parties can now look back at how important they believed each need was to them at the time it was listed. (The two right most columns in Figure 1 show the Needs and Resources list along with who entered each need and how important it seemed at the time.) The parties can also now enter a number in the "`satisfied/utilized`" column (Figure 1, third column from right) to indicate how well each need is being satisfied by the current proposal and how well each resource is being taken advantage of. Additionally, selecting the `regroup` option on the menu bar in Figure 1 allows the parties to request that the needs and resources list be sorted by weight; by weight for each owner; by satisfaction/utilization; or by satisfaction/utilization for each owner.

   Several things can happen as a result of sorting and inspecting the weights and satisfaction values. The parties can decide that the current proposal is not satisfactory because the initial list of needs and resources was inaccurate or

incomplete. As a result, they can revise the list. They can decide that the list was accurate but that the weights were not and so they can revise the weights. Additionally, they can see which needs are not being met and look for resources to fill them. All of these alternatives can result in the generation of a new proposed solution [3].

NegotiationLens results in making the reasoning and resources of both sides explicit. As a result, it can lead to the development of solutions which are more satisfying and more sound than unilateral solutions based on unarticulated criteria. Additionally, and for the same reason, the process can move the parties towards a better long term relationship.

3.1 Using the System: Two Case Studies of Simple Negotiations

1. **The Visiting Researcher:** This first example illustrates a negotiation in which a dispute concerning *goal conflicts* and *time constraints* were successfully resolved. In the example a disagreement arose in a research group when a visiting researcher told her manager that she wanted to start up two new projects and the manager replied that she should not do so until she had completed the one project she had already started. At this point, the manager of the group, the researcher and her colleague on the existing project entered into a collaborative negotiation. Each side began by separately expressing the problem as they saw it. Next the two parties went on to offer an initial unilateral solution to the problem.

   Following this the parties used NegotiationLens to make the reasoning underlying their solutions explicit by creating a list of what each wished to get from the situation and what each was willing to offer. They also entered a weight for each need.

   At this point, because the parties had a needs and resource list, they were able to create a joint solution in which the researcher was given the go ahead to accomplish all that she wanted and the lab got a commitment to have three systems built, rather than just the one which the manager was expecting. (As an aside, the commitment was met.)

   In the final stage the parties considered the goodness of the solution by entering a value indicating the extent to which each need was satisfied and each resource was utilized (Figure 1, leftmost column). Considering the joint solution and the satisfaction values entered by the parties it is not surprising that the negotiation resulted in a solution which both parties felt was mutually agreeable. Beyond that each party reached a better understanding of the needs, strengths and concerns of the other, this resulted in a more relaxed group dynamic and allowed them to avoid future conflicts around these sorts of issues. Additionally, it strengthened the relationship between the researcher and her colleague in that they agreed to (and did) jointly write a paper on the first project upon its completion.

2. **The New Faculty Member:** The second example is one in which the negotiation began with a *goal conflict* and resulted in a mutual gain once one party reconsidered the legitimacy of part of his initial solution. In this negotiation a young researcher (Dennis) was trying to negotiate the terms of a first faculty appointment with the help of a more senior colleague (Karen). In this example NegotiationLens was used only by the two colleagues to help the junior colleague define an optimal solution. The second party, Dennis' new department head, Isaac, was not directly involved in the use of the tool, although he was affected by the rethinking that resulted from its use. Initially Dennis wanted Isaac to allow him to buy out of teaching with some research funding he had been offered. More specifically, he wanted Isaac to use the buy out money to bring in one of Dennis' friends to teach his courses. He was particularly eager to have this friend as an intellectual companion in his new job.

   Dennis and Karen used the tool to create a problem statement, an initial solution and a needs and resource list both for Dennis and, to the extent possible, for Isaac. Dennis and Karen then reviewed the list in an effort to construct a proposal which would be acceptable to Isaac, since he had initially exhibited resistance to the idea of Dennis' buying out of teaching. Isaac believed that having the faculty teach the students provided the students with the best education.

[3] As parties develop their negotiation skills they can use the regrouping features in some fairly sophisticated ways in order to make revisions to the list and its values. For example, if the parties are not satisfied with a solution, but are not sure why, they can first sort the needs by weight and then enter the satisfaction values, allowing them to more easily see what important needs are not being met. As a second example, if one party feels that the current solution is more favorable to the other side it can sort the list by owner and then within that by satisfaction values. The parties can then easily see if the solution really is inequitable and try and satisfy both sides more fully.

In reviewing the list of needs it came out that Dennis' reason for wanting to buy out was that he wanted to do well at his new job and that underlying his thinking was the criterion of doing as much research (and therefore as little teaching) as possible. However, it also became clear in considering the department head's position that doing well included being willing (if not down right eager) to teach. This suggested that Dennis should change his criterion as to what constituted doing well at the new job, and as a result, change his stance on the buy out. Having done this Dennis and Karen then reviewed the extent to which the resources in the situation were being utilized. They noticed that if Dennis did not buy out he could use his research money to bring in his friend as a visiting professor (and possibly in the long term change the department head's attitude concerning the potential contribution of visiting faculty.)

This process, in which Dennis reviewed the legitimacy of the criteria underlying his initial position led to a solution which benefited both parties.

4 Case Study: Generalizing NegotiationLens to Large Design Projects

In this section we will attempt to demonstrate the general usefulness of NegotiationLens on two counts. What we are asserting here is tht NegotiationLens has the potential to resolve larger-scale conflicts and is applicable in a range of domains which includes design. In making this argument we will refer to aspects of the negotiations described above and show how, they can serve as models for resolving the conflicts described here[4].

Overview: Several years ago a major computer company gave high priority to a project whose goal was to create a workstation which would provide an integrated environment in the form of a graphical user interface (GUI) for running a wide range of business applications. Towards this end, upper-level management assembled a set of several teams of leading employees.

Unfortunately, two years later after continual setbacks the project was cancelled without the release of the workstation.

Below we describe some of the inter- and intra-group conflicts which appear analogous to the 2 cases described in the previous section.

Selected Groups: Responsibilities and Conflicts[5]

1. **Group: User Interface (UI)**
   The UI team was charged with ensuring that the diverse set of applications would have a unified look and feel, giving the system a global coherence. The team, diverse in its talents, included several software engineers and one user interface designer.

   One of the intra-group[6] conflicts arose because the differing backgrounds of the group members led them to have different implicit criteria concerning what constituted a good problem solution. That is, the system was slated to contain a set of applications which were similar but not identical and which in addition had names that were sufficiently similar that even members of the development group often got confused and launched the wrong application. Because of their confusability the group member with the user interface background proposed that this set of applications should be grouped under a single menu. His rationale was that putting the set on one menu would allow users to focus on the differences between the members of the set and as a result confuse them less often.

   The proposal met with strong but not clearly explained resistance. Repeated and lengthy discussions within the group did not resolve the issue. Finally one of the newer software engineers confided to the UI designer that the resistance came from the software engineers' rationale that an interface should reflect its implementation.

   The example of the junior faculty member who reviewed his criteria for job success provides us with an analog for looking at this dispute. Had the UI group engaged in a NegotiationLens-like process the software engineers' criterion of wanting interfaces to map on to architectures might have become explicit. The engineers could then

[4] Aspects of this case have been reported in less detail in Groenbaek, Grudin, Bodker, & Bannon (1991) and in Grudin (1991).

[5] Not all of the project's design decisions generated conflicts and not all of the conflicts were intractable.

[6] This sort of lengthy technical dispute also occurred between UI and the human factors and industrial design groups. The nature of the disputes were sufficiently similar that we will not elaborate upon them here.

have decided whether they wanted to retain this constraint in this situation. That is, the rationale does have some legitimacy. It makes systems easier to maintain and modify. However, had the rationale and its legitimacy been examined in this context where it was also decreasing usability a considered decision could have been reached more quickly and with less acrimony.

2. **Group: Performance Analysis (PA)**
   The usual responsibility of the Performance Analysis group was to test newly built systems in order to ensure that they could meet minimum performance requirements. For this project it was decided that, contrary to the usual arrangement, the PA group would be involved at the outset in order to avoid finding hard to deal with after-the-fact problems (such as finding that systems which had been targeted to support and be marketed with 24 terminals could in practice only support, and be marketed with 18). The performance analysis group had over time developed many tools for assessing system prototypes and their ccontribution was potentially valuable to the UI group. However, the performance analysts were accustomed to making assessments once they had a prototype to work with. In this situation, where they were being asked to produce an evaluation in the absence of a prototype they decided to come up with a list of minimum response times for basic operations such as cut, paste, copy, and delete.

   When PA presented their proposal for performance requirements to the UI group the proposal was greeted with resistance which was sufficient to cause the performance analysis group to withdraw from the project. Both groups ultimately lost out in this situation. A successful contribution to the high visibility workstation project would have benefited PA. Additionally, had PA been able to uncover an inadequacy in the performance of the workstation before it was cast in code it would have been helpful to UI. Looking back to the examples presented in Section 3.1 it seems as though NegotiationLens could have helped the two sides to uncover the source of the rift and to have come to a mutually beneficial solution. That is, the resistance of the UI group came from two sources: The performance analysis group had not made explicit how they had come by the performance criteria in their proposal; the criteria seemed arbitrary. Further, a role conflict existed. UI felt that the advice was presumptuous, they believed that as good engineers they were continually trying to optimize their implementations and that the PA group had not really spoken to that hard problem, of which they were well aware. What we see here is a conflict in which each side had something that would have benefited the other but neither side was able to make that clear. As a result the two sides withdrew and lost the opportunity for mutual gain. If we view the visiting researcher and new faculty examples as partial analogs to this situation we can use elements of both negotiations to construct a scenario with a mutually beneficial outcome.

   Had the UI group been encouraged to make a needs and resource list along with legitimizing statements, it could have become clear that they believed they were from the outset optimizing what they were implementing. Had the PA group also been encouraged to make a resource list UI could have been made aware of the tools PA had for doing evaluations on prototypes. Then, when looking for opportunities for mutual gain both sides could have made an explicit decision as to how they could have made the best use of each others' resources. Under this scenario it seems that the two sides might have come to an agreement that PA could provide useful input once a prototype had been built (something which the UI group could have done). Had such a plan been implemented the work of both groups might have progressed more rapidly and/or effectively.

3. **Marketing (MK):**
   The central concerns of marketing were to maintain and expand the company's market share. Marketing attempted to maintain market share by ensuring that new products were backwardly compatible with existing products. They attempted to expand market share by requesting functionalities which competing companies featured in their advertising. This meant that marketing would frequently demand that UI change functional and/or interface elements of the system. Often this would occur at a late date, since that is when the product tended to be evaluated by marketing.

   These disputes are inherently difficult to resolve. There are real differences between the goals of the two parties and both are legitimate. Although NegotiationLens may not be able to get the parties to find a mutually agreebale solution here it can at least ease the tension accompanying inter-group conflicts as we saw in the visiting researcher example.

   The process fostered by the use of NegotiationLens eases tensions by encouraging a working through of differences within a framework of respect and commitment. That is, within this framework the parties make explicit the criteria underlying their positions and provide explanations for the weight which they are giving to their needs. This means that although the parties may still disagree at the end of the process, they have seen that the disagreement is not

a result of one side discounting the other's rationale, nor from one side being arbitrarily stubborn. This allows the disagreeing parties to understand their disagreements, see why they are legitimate, and feel that they are being treated with consideration and respect.

Looking at the case study presented in this section it seems that NegotiationLens may have been able to resolve a number of the conflicts which arose within and between the groups working on the advanced workstation project. We claim this is the case because it moves parties into a collaborative stance by having them make their implicit reasoning explicit, allowing the creation of jointly developed problem solutions.

5 Summary, Implications and Future Work

In this paper we have argued that conflicts over: goals; allocation of resources; roles or turf; and judgements as to what constitutes an optimal solution to the problem at hand repeatedly arise in the context of both large and small cooperative work projects in a variety of domains. Further we have claimed that these conflicts can successfully and appropriately be resolved through collaborative negotiation. In support of this argument we have outlined a prescriptive theory of collaborative negotiation and have described NegotiationLens, a tool which has allowed parties involved in conflict resolution to work through the process described by our theory.

Of course there is a cost to negotiating disputes as they arise. In many cases the cost will not be worth the benefit. But in other cases, the ones presented here, we believe that it will be clear that negotiation will result in a savings. For example, if a group whose contribution had at the outset been considered valuable (like performance analysis) is withdrawing from the collaboration, entering into a negotiation is worthwhile. Similarly, if a dispute concerning the implementation of a central feature of a system is continuing for a period of time which is equal to the time it would have taken to implement the feature it again seems that a several hour negotiation will produce a savings.

A related issue arises concerning reducing the cost of using negotiation tools in supporting complex group problem-solving. If the negotiation tool is integrated with other tools used in the daily work environment it is likely that the overhead involved in their use will be minimized (Grudin, 88; Conklin & Yakemovic, in press). As we mentioned in Section 3, NegotiationLens is built on top of the Object Lens system which provides users with an object oriented database, electronic mail (Lai, Malone & Yu, 89; Lee & Malone, 88 & 90), meeting scheduling (Resnick & Jordan, 90), and collaborative writing facilities (Adelson & Jordan, 91; Adelson, 91 & 91a). Further, the system was designed to be used both by experienced and unsophisticated computer users and to allow users to develop new applications to support cooperative work as needed. Our hope, therefore, is that NegotiationLens, because it is a part of the larger Object Lens environment will be a low cost tool for aiding group problem-solving, however these claims are yet to be put to evaluative tests. And this points the way towards future work. What we would want to do is compare conflicts in which no negotiation process was suggested against ones in which various negotiation techniques were given to the parties. Further, we would want to look at each of these negotiation techniques when they were and were not supported by a computer tool.

A last, but important test of this work concerns multi-group negotiations. Many serious conflicts involve only bi-lateral negotiation (union versus management; divorce and custody cases; two-party treaties) and there is a need to study these situations in detail. However, those who are seriously committed to the issue of negotiation also need to deal with multi-party conflicts (global environmental debates; middle east conflicts, etc,). It is not yet clear how the study of two-party conflicts can generalize to our understanding of situations involving many factions. Some results may generalize, some may not. And this too is an important topic for future research.

6 References

Adelson, B. Educational tools for what you wanted to do anyway. *Proceedings of the Fourteenth Annual Meeting of the Cognitive Science Society*, 1991.

Adelson, B. A collaborative negotiation tool. *SIGCHI Bulletin.* October, 1991a.

Adelson, B. and Jordan, T. The Need for Negotiation in Cooperative Work. E. Barrett (ed.). *The Social Creation of Knowledge.* Cambridge: MIT Press. In press.

Brockner, J. and Rubin, J. Entrapment in Escalating Conflicts NY: Springer-Verlag. 1985.

Carroll, J. *The Nurnberg funnel: Designing Minimalist instruction for practical computer skill.* MIT Press: Cambridge, MA. 1990.

Carroll, J. M., & Rosson, M. B. Deliberated evolution: Stalking the View Matcher in design space. Human-Computer Interaction, Volume 6 (1991), Numbers 3 & 4

Conklin, E. J., & Yakemovic, K. B. A process-oriented approach to design rationale. Human-Computer Interaction, Volume 6 (1991), Numbers 3 & 4

Conklin, J. and Begeman, M. gIBIS: A hypertext tool for exploratory policy discussion. In Tatar, D. (ed.) *Proceedings of the Second Conference on Computer-Supported Cooperative Work.* ACM press. 1988.

Crowston, K. *Towards a Coordination Cookbook: Recipes for Multi-Agent Action* Doctoral Dissertation, MIT Sloan School. 1990.

Crowston, K. Malone, T. and Lin, F. *Cognitive science and organizational design.* HCI, 3, 59-85.

Fischer, G., Lemke, A. C., McCall, R., & Morch, A. I. Making argumentation serve design. Human-Computer Interaction, Volume 6 (1991), Numbers 3 & 4

Fisher, R. and Uri, W. Getting to Yes. NY: Penguin. 1981.

Fisher, R. and Brown, S. Getting Together. NY: Penguin. 1988.

Greif, I. Computer supported cooperative work. I. Greif (ed.). Morgan Kaufmann: San Mateo, CA. 1988.

Grudin, J. Why CSCW applications fail. In Tatar, D. (ed.) *Proceedings of the Second Conference on Computer-Supported Cooperative Work.* ACM press. 1988.

Groenbaek, K. Grudin, J. Bodker, S. and Bannon, L. Cooperative System Design. In *Participatory Design.* Schuler & Namioka (eds.) Erlbaum: Hillsdale, NJ. 1991.

Grudin, J. Systematic sources of suboptimal interface design in large product development organization. HCI. June, 1991.

Kolb, D. *The Mediators.* MIT Press: Cambridge, MA. 1983.

Kolodner, Simpson and Sycara. A process model of case-based reasoning in problem-solving. IJCAI 85.in problem-solving. IJCAI 85.

Lai, K., Malone, T., Yu, K. "Object Lens: A 'Spreadsheet' for Cooperative Work" ACM Transaction on Office Information Systems, 6(4) pp. 332-353. 1989.

Lee, J., & Lai, K-Y. What's in design rationale? Human-Computer Interaction, Volume 6 (1991), Numbers 3 & 4

Lee, J. Sibyl: A qualitative decision management system. In P. Winston (ed.). *AI at MIT* Vol. 1, MIT Press: Cambridge, MA.

Lee, J. and Malone, T. How can groups communicate when they use different languages? In R. Allen (ed.) *Proceedings of the ACM Conference on Office Information Systems.* Palo Alto, CA. 1988.

Lee, J. and Malone, T. Partially shared views. *ACM Transactions on Information Systems.* 1990.

Lewis, C., Rieman, J., & Bell, B. Problem-centered design for expressiveness and facility in a graphical programming system. Human-Computer Interaction, Volume 6 (1991), Numbers 3 & 4

MacLean, A., Young, R. M., Bellotti, V. M. E., & Moran, T. P. Questions, options and criteria: Elements of design space

analysis. Human-Computer Interaction, Volume 6 (1991), Numbers 3 & 4

Malone, T. and Crowston, K. *Toward and Interdisciplinary Theory of Coordination.* MIT Center for Coordination Science Tech. Report CCS TR 120. 1991.

Pruitt, D. and Rubin, J. *Social Conflict: Escalation, stalemate and settlement.* Random House: NY. 1986.

Resnick, P., Jordan, T. Scheduler'sAide: an Application in Object Lens to Support Meeting Scheduling. Internal Memo, Center For Coordination Science, Massachusetts Institute of Technology. 1990.

Tatar, D. (ed.) *Proceedings of the Second Conference on Computer-Supported Cooperative Work.* ACM press. 1988.

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., and Suchman, L. Beyond the Chalkboard. *CACM*, 1987.

Simpson, R. A Computer Model of CBR in Problem-Solving. GA Tech PhD thesis, 1985.

Susskind, L. and Cruikshank, J. *Breaking the Impasse: Consensual Approaches to Resolving Public Disputes.* Basic Books: NY. 1987

Sycara, K. Resolving Adversarial Conflicts. GA Tech PhD thesis, 1987.

Sycara, K. Utility Theory in Conflict Resolution. In press.

Winograd, T. and Flores, F. *Understanding computers and cognition.* Ablex: Norwood, NJ. 1986

The organization as a structure of negotiated and non negotiated binds[1]

Cristiano Castelfranchi, Rosaria Conte and Amedeo Cesta

IP-CNR
Institute of Psychology of the National Research Council

Viale Marx 15, I-00137 Rome, Italy
e-mail: {pscs, amedeo}@irmkant.bitnet

Abstract

This paper addresses some drawbacks of the communicative view of social relations, in particular the language-cooperation link. Some limits of this theory are shown, which seem to reduce its analytical and applicative power. We detail to some extent some points that should be added to the current paradigms in order to overcome the given problems. Different kinds of pre-established external structures will be shown which limit, de facto, *realistic agents'* behavior. In particular we consider an agent trying to cooperate with others in a social setting, and describe how the agent receives a number of pre-existing constraints, independent of her will. Such constraints are mainly provided by: the objective structure of power and dependence relationships among the agents involved; past problem-solving experience, either shared with others or already codified; existing norms, which are aimed at regulating the agents' behaviors in a given context.

1. A premise

To understand the many changes generated from the spread utilization of computer both in social life and in work processes, an understanding of *human cooperation activity* (more generally of social activity) and an investigation on the *real nature of organizations* may be required. With respect to these two crucial points diverse disciplines (namely Organizations Theory, Computer Supported Cooperative Work, Distributed Artificial Intelligence, and Conversation Theory) share a quite idealistic paradigm.
Let us show few examples of such a paradigm:

- according to Winograd [18] *"from a language/action perspective.. people act through language.... Conversations for action are the central coordinating structure for*

[1] This work has been partially supported by CNR under "Progetto Finalizzato FATMA", and under "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo", grant n.104385/69/9107197 to IP-CNR. The authors are members of the "Project for the Simulation of Social Behavior" at IP-CNR.

human organization. We work together by making commitments so that we can successfully anticipate the actions of others and coordinate them with our own."

- Gasser [13] views an organization as *"particular set of settled and unsettled questions about beliefs and actions through which agents view other agents. Organizational change means opening and/or settling some different set of questions in a different way, giving individual agents new problems to solve and ... a different base of assumptions about the beliefs and actions of other agents'"*. As Chaub-Draa et al. [7] says according to this view *"an organization should not be conceived as a structural relationship among a collection of agents or a set of externally-defined ... the best way to define an organization is to view the concept of organization as embedded in the belief, intentions and commitments of agents themselves. In other word, an organization is defined as a set of agents with mutual commitments, global commitments, mutual beliefs and eventually, joint intentions when these agent act together to achieve a given goal"*. Both the points of view equate an organization to a team --a group of agents who agree to carry out a joint activity. Actually a huge difference exists between a spontaneous group of agents and an institution.
- Bond [3] uses a notion of commitment taken from the sociological studies of organizations (Becker [1]). In his view, an agent participates in a given organization setting by commitment; commitments are generated by negotiation and are used to constrain individual actions. Using this formal apparatus Bond defines an organization as *"a set of agents with consistent mutual commitment"*. In this way he consider just one aspect of the individual role in an organization (and in a team too). One missing aspect concerns the fact that in agent 1's commitments with respect to agent 2 a key role is played by the *expectations* of agent 2 on agent 1 and by the particular *role power* of agent 1. For example agents do not have a number of resources either through commitment or by contracting with each other, because they may be already endowed with them. Moreover, agents are given some other resources by commitments (obligations, permits) of other agents whom they are not able to negotiate with: in fact the other agents may provide these resources out of their authority.

The model the examples refer to satisfies the following assumptions:

1) agents consciously and intentionally build up their relationships, the organization (and even their society), and these consist in the sharing of the agents' minds;

2) agents are viewed as ruled by cooperative goals and common interests, and, if they come into conflicts (to be settled for the common welfare), this just depends on their lack of knowledge, or on their different points of view, tasks and roles. There is apparently no room for an agent's intention to take advantage of a relationship or of an organization for personal interests;

3) communication is considered to be the "glue" of an organization and the nature itself of an interaction: the models of language and conversation are turned into the models of social interaction and organization.

The nature of human-computer interaction, as well as the nature of working with and by means of a computer, can but emphasize the delusion that communication is the basis of society and organizations. Computers and HCI may hide such *non communicative* relations of organizations as power, dependence, interest, or normative relations. These become less visible, are "immaterial" (like money when it is passed via computer).

The interest of such a model for the cognitive scientist lays in the relevance given to agents' mental representations and to the autonomy of agents themselves. However such assumptions are insufficient: according to our analysis, in fact, the explicit representation of concepts in the agents' minds do not imply neglecting the role played by externally imposed limitations to their activities. This paper will address some drawbacks of the communicative view of social relations, in particular the language-cooperation link. Furthermore, some limits of this theory are intended to be shown, which seem to reduce its analytical and applicative power. We also detail to some extent some points which should be added to the current paradigm in order to overcome the given simplifications, by showing different kinds of pre-established external structures which limit, de facto, *realistic agents'* behavior.

2. Free market and spontaneous cooperation

Two are the main arguments against a conversational theory of cooperation:

1) within conversational theories, a mild view of linguistic interaction is dominant. As ethnomethodological studies have exhaustively shown, formal aspects of conversation (turn-taking, monitoring the listener's attention and understanding, etc.) induce adoption and cooperation in any sort of linguistic interchange -- even the most aggressive one [4]. There is no verbal exchange without adoption. However, cooperative routines in conversation are not sufficient conditions for claiming that conversation is adoptive, if the latter statement implies some decision-making concerning whether or not to adopt the other conversant's goals. Those routines are used by the participants as a conversational tool. Their function is precisely to allow for conversation to be carried on. However, the conversants do not need to have goals corresponding to these functions. It is sufficient that they "know how" to carry on a conversation, that they apply some routines and the job is done. No cooperation, stricto sensu, is necessarily implied.
Of course, if linguistic interaction is viewed as the general case of interaction (rather than the opposite), and if conversation is viewed as necessarily cooperative, a mild, non realistic view of (organizational) interaction follows.

2) Emphasis on negotiation: cooperation within organizations is fully described by the whole pattern of binds holding within each organization plus their "histories". However, not all such links are spontaneously created and negotiated by the agents involved: the structure of cooperative work is not always so dynamic and "contractualistic". Nor is it true that all commitments can be handled as equivalent, except for their deadlines if any. Quite on the contrary, when a social agent cooperates with others, especially if this happens within an institutional organization, she undergoes a number of pre-existing constraints independent of her will, mainly provided by the following aspects:

- the *objective structure of power and dependence relationships* among the agents involved;
- *past problem-solving experience*, either shared with others or already codified;
- *existing norms*, which are aimed at regulating the agents' behaviors in that context, setting up which commitments might take place and which have to.

Let us see all three aspects, starting from the last one, to dwell a little bit more on the first one (the most fundamental) and on the relation that it bears with the models of negotiation and cooperation.

3. Norms, "conversation" and commitment

Agents within organizations are not allowed to freely commit to any sort of activity. Not all agents are allowed to perform all requests; not all requests are allowed to be asked of all agents; not all agents are held to accept all requests, and so on and so forth. This bears a number of consequences, also on applications:

1) There is a question of rationality of the conversational network and of the decisions of the single agents: agents might perform either useless or wrong requests; their commitments might be improper; they might carry on conversations for possibilities or for orientation to acquire information which should be already available to the system and not drawn from agents' personal experience (see, for instance, the ruling procedures).

2) Furthermore, there is a problem of monitoring and conformity, or at least of tutoring/counseling provided to the negotiating agents. Suppose agent 1 ask agent 2 to perform a given procedure but is unknown that the procedure is invalid or useless unless a formal permission is given by agent 3. Agent 1 believes her job is done, when in fact it is not, especially if agent 2 performs the action required, thus acting uselessly or improperly. Agents need help with regard to their commitments, and possible to be reminded of necessary sub-procedures.

3) Finally, not all commitments are equivalent, nor can be handled in the same way. If agent 2 is committed to perform a given action in front of agent 1, it should be distinguished whether:

- this commitment is merely personal (and if so, is it out of friendship or due to negotiation? For in the latter case, an obligation of reciprocation is impinging on agent 1);
- it is due (and if so, is it out of reciprocation or for institutional tasks?).

Furthermore, the social cognitive structure of commitments essentially consists of letting the other to be entitled to "demand satisfaction". Therefore, it is essential of commitment the "responsibility" assumed by the committed agent [15] [17]. Now, while in personal relationships, this is quite obvious, in role commitments institutional hierarchies come into play. Indeed, it is possible that while agent 2 is committed with agent 1, who is responsible of a missed fulfillment is agent 4 (agent 2's chief) whose task is checking agent 2. Without even knowing it, and thanks to his role, agent 4 is committed with agent 1. Is the organizational support able to remind agent 4 of this indirect commitment? Is it able to inform agent 1 about the identity of the agent really responsible of the commitment, even though he did not participate in the initial conversation? or, for example, the knowledge about the commitment between agent 1 and agent 2 is just contained in their "private" knowledge [2]?

4. Memory based planning and cooperation

A further constraint to the "freedom" of agents' commitments lies in the individual or collective memory of solutions to past problems.
In the last few years, studies on planning have thoroughly explored both the planning based on analogical reasoning about previous situations (case based) and the instantiation and adjusting of abstract and failed plans (adaptive) [14]. In addition, one

should remember the studies about scriptistic knowledge, which provides a baseline to follow in customary activities. It is our belief that both types of knowledge (that causal and temporal of planning and that situational and customary of scripts) should be re-unified [9]. For sure, when the agent must solve a problem, either personal or social, she usually neither resort to general solutions nor she starts every time from zero. Rather she tries to apply plans and procedures already known. This is true for cooperation as well.
First, shared knowledge of plans and scripts is essential for agents to understand one another [9], to understand requests and coordinate the activity without specifying everything. Elsewhere [9], it has been shown that rather frequent types of topic changes are allowed in conversations without seriously damaging comprehension and the carrying on of the participants' interaction.
Secondly, collectively found solutions will be channeled by this knowledge (with negative consequences as well, such as ignoring new and maybe better solutions). This is especially true since most scripts are multi agent structures usually cooperative. Indeed, in many circumstances, interaction is allowed by integration of single agents' plans which results in a multi agent overall planning structure (MAPSs). MAPSs [10] are but plans which mention more than one agent; more precisely, they are n-action plans (for $n > 1$) in which not all actions may be accomplished by one single agent.

There are several types of MAPSs. One of the dimensions along which MAPSs vary is the cooperation/exchange one:

- cooperative MAPSs are converging into one and the same common end. For instance, a multi agent plan for a robbery is a cooperative plan, since it is aimed at a single common end but includes a variety of specialized tasks which must be accomplished by distinct agents;
- exchange MAPSs include more than one end. For instance, the subplan BUY-NEWSPAPER is complementary to the subplan SELL-NEWSPAPER in a global MAPS which is aimed at two complementary ends. Interestingly enough, MAPSs often fall in the middle of the continuum between exchange and cooperation. There are exchange MAPSs in which agents act to achieve a common end (f.i., the MAPS for doctor-patient interaction). What is more, each type may be included in the other: a MAPS for robbing a bank is a means for each agent's getting its share of the cake; on the other hand, a cooperative MAPS might include some exchange among the participants: one might ask for another's help to accomplish its share of the plan in the common interest.

A further dimension of variation is the orchestrated/functional one:

- Orchestrated: MAPS is conceived of by a "third" agent's mind [5]. Consider the band of thieves: their cooperation might be orchestrated by a criminal organization, which worked out the whole plan with its specialized tasks and roles and enrolled the executors. The latter might be unaware of one another and even of the final end -- each performing its task as a result of the negotiation with the third agent. Here, therefore, cooperation among the executors of the plan does not imply any form of negotiation and commitment among them.
- Functional: MAPSs might be external to any mind, so to speak, and exclusively due to their selective (reproductive) effects [5]. Plans of this sort are usually customary cooperative plans, although ritual aggression -- like the fights for getting access to females among several species, or the ritual insults from one side to the other occurring during the wedding ceremony in some African villages -- might be seen as examples of functional multi agent plans.

MAPSs are in sum a memory of collective plans, with tasks and roles specified, enriched with knowledge about the requirements of the role players, sequence and justifications of different agents' actions. This deeply modifies negotiation and requests, transforming them into a simple allocation-acceptance of roles.
MAPSs recur in most aspects of everyday life. From the simplest to the most complex forms of interaction, memory of past solutions comes to the agents' help. For all, let us consider the example of BUY-NEWSPAPER. Several interesting aspects of the ordinary activity of buying newspapers reveal not only that it consists of applying an existing solution to a problem already encountered, but also (and especially) that the solution applied is a collective one. As we shall see, (part of its) consequences are not calculated by the single agents although they favour both any single interactional episode and their iteration:

1. Public "advertisements" of the roles: Customers do not need to search agents whom to give money in order to receive newspapers in return. Indeed, an important sub-aspect of problem-solving directed to exchange is a complex procedure for identifying the agents endowed with the resources and willing to enter a relation of exchange with the would-be customer. Usually, newspapers sellers offer their "goods" in acknowledgeable booths designed on purpose.

2. Sellers are held to "adopt" the customers' goals of buying newspapers. A further fundamental aspect of social problem-solving is how to have one's goal adopted by a "useful" agent once one such type of agent has been identified [6]. This point to a special aspect of MAPSs, namely their including a representation of the norms regulating the behavior of agents who bids for a given task. Therefore, once accepted the role of selling newspapers, an agent loses some of its control about whom to enter a relation of exchange with. This is no longer a matter of negotiation: sellers are "held" to sell papers and magazines to any customer no matter how little they like him, provided of course the latter gives back money. Interestingly enough, the most fundamental norm regulating exchange is one which prescribes reciprocation: once obtained their outcomes, there is no utilitarian reason for agents to keep to the promise of reciprocating their partners. Consequently, a norm is needed to ensure a practice in absence of which the continuity of social interaction would be seriously impaired.

3. The type, entity and time of reciprocation is socially set up. As claimed by social scientists, one of fundamental features of economic exchange is precisely the simultaneity and quantifiability of reciprocation. In the example considered, in addition, the type and amount of reciprocation is set up beforehand, independent of the interactional episode, and is not subject to negotiation.

5. Objective structure of power and dependence relations

The fundamental structure that has a strong impact on agents, their conversations and commitments, even beyond any institutional organization is the structure of their dependence relations. Such a structure is objective in the sense that it exists independent of the agents beliefs and wants. It largely determines whether actions conversations and cooperations will get through, thus determining the rationality or irrationality of communication as well.
Agents cooperate because they are lacking some ability, competence or resource required to achieve their goals or fulfill their tasks. In other words, they are lacking

some powers which other agents possess. Consequently, they are dependent on others to reach their goals [6] [8].
Dependence is not necessarily a social notion. A relation of dependence may be said to occur whenever: a) any object or event in the external world may increase, if used, the probability that a given state of the world be realized, and b) that world state is represented as a goal by at least one agent. In such a case, we say that agent to be dependent on the enabling object or event. The latter will then be called a "resource". Resources enter the structures of the actions. An action is a relation that can hold among agent(s), goal(s), and resource(s). We then say that anything that is involved in the action, except agent x, is a resource r of that action a. The use of a resource r necessarily implies that the action requiring it be done.

A set of resources is required for any act to take place. In our notion, cubes, tables and hands are resources in the block world. In the social world, others may be used as resources (not only in exploitation but also in prosocial action: in help, in a quite abstract sense, the recipient is a resource of the action "give help").

Agents are usually dependent on the existence of resources. We will call this type of dependence a resource dependence, to distinguish it from social dependence. Agent x is said to be dependent on resource r when the latter is required for x to achieve his goal that p. Thus, for instance, x is resource dependent on a hammer for having a nail driven into a wall.

In our definition, social dependence is a resource dependence in which the resource slot is filled in by a social agent. Agent x is said to be dependent on agent y whenever:

- a resource (namely an action a) is required for x to achieve one of his goals p, and
- x is not able to do a, while
- y is able to do so.

in other words, x depends on y with regard to an act useful for realizing a state p when p is a goal of x's and x is unable to realize p while y is able to do so. Being able to do any action a is here meant as x's having a in his action repertoire, and either its condition q is true or x has another action which leads to q.
In this context, y's action is a resource for x's achieving his goal.

Social dependence, as well as resource dependence, is an objective relationship, in that it holds independently of the agents' awareness of it: x may depend on y even though they both ignore the fact. However, many relevant consequences may derive from x's and y's (either unilaterally or mutually) becoming aware of it: to mention just the most salient ones, x may try to influence y to pursue p, while y may choose whether to adopt x's goal or not [6] [8].
Moreover, not only a dependence relationship may be known; it may also be wanted, in that either x or y may actively "work" on maintaining or strengthening the relationship. And, not only a dependence relationship may be wanted once established. It may even be created by the agents, by producing those objective conditions that define a dependence relationship (a certain goal in x's mind; the lack of a certain power condition, etc.). So, for instance, y may influence x, and induce him to have p as a goal of his own; since p cannot be achieved by x without y's help, y has created a dependence of x on her by means of an influencing strategy; otherwise, supposing that x already has p as a goal of his and is also endowed with the power conditions useful for achieving it, y may deprive x of some of them (say, by stripping him of a certain resource), thus making x become dependent on her relative to p.

Within organizations, the nature of powers and capacities is two-fold. On the one hand, in addressing himself to y, x refers to the actions that belong to the agent's action repertoire, to her know how. For instance, if agent 2 cannot speak English I cannot ask him to write a letter in that language. On the other hand, the agent's competence means something quite special within the organizational context: it refers to what an agent is held to do, her role-tasks. From this point of view, it is unrelieved whether agent 4 (the chief) is able or not to write a letter in English since he is not held to write letters. This type of power and capacity has a normative base, and leads us back to the necessary consideration of the role of norms within a model of organizational cooperation.

6. Dependence and conversation

From any given dependence net [6] important predictions concerning the agents' communications and commitment are derived. A dependence net precedes conversation and commitment and simply implies their goals-tasks and their capacities. In other words, not all structures of communication and commitment are equally rational and efficient with regard to the underlying dependence network. If, say, agent 1 is dependent on a resource that agent 2 controls, it will be rational to ask agent 2 to let the resource to be used by agent 1. The type of request will obviously depend both on the nature of the resource in question (whether this is information, material goods, etc.) and the nature of the control exercised by agent 2: whether she is simply using the resource, possessing it, etc.).
If agent 1 does not know that the resource is under someone else's control, or does not know the latter's identity, he will undertake useless conversations. If agent 1 does not know how and why to obtain the resource from agent 2, his chances of negotiating with agent 2 about that resource are rather poor: this is the case when agent 2 is not spontaneously "benevolent" [11] and agent 1 is not informed about some possible dependence of agent 2 on him.
True negotiation does not consist, as in contract nets [12], of requests or proposals that the other agent automatically accepts unless she is involved in some other job or has got the opposite goal. In true negotiation, agent 2 must have some specific reason to do what she is required of doing. Even in organizations, as seen above, most commitments are not necessarily due, but rather optional. Even when "due" they might be optional and not necessary procedures; sometimes, they are simply "favours". True negotiation consists of knowing implicitly or explicitly how to persuade agent 2 to cooperate.

7. Conclusions

The analysis of the dependence, problem-solving and normative constraints underlying conversations and commitments among cooperative agents has been argued to shed new light on the study of organizational cooperation. It serves to point out a perspective necessary to the understanding of cooperative relations and organizations: this has already been shown by Winograd, when he states that the conversational approach is not sufficient and that complementary views are possible, like role analysis [18]. Indeed, the above constraints show that the conversational view is somewhat misleading: agents do not spontaneously negotiate all their commitments. Usually, they found themselves to comply with norms, to accept tasks, they even happen to be

indirectly, and therefore unknowingly, committed with someone because they are responsible of someone else's doings! Conversational theory disguises and mixes up different types of commitment and cooperation in a cooperativistic and contractualistic view. As a consequence, predictions concerning what the agents will do, and what would be rational for them to do are not allowed. Even from the applicative point of view, the theory runs into two risks: the explosion of communications among cooperative agents with consequent huge "coordination costs" [16]; on the other, the lacking of essential information for creating commitments and adequate solutions to the problems encountered in an organizational context.
This view of organizations as multi-layered structures, with binds of different types and levels constraining the conversational and commitment "freedom" of the agents, bears many consequences also on the role of the computer as a support both to work and to interactions. So, it has consequences on the kind of information that the computer has to convey to the user. In other terms, the computer should help the members of the organization or its clients, to know, understand and remind who depends on whom; who is able to cooperate on a certain task; who is responsible for a certain act or commitment; what is the local situation of credits and debts of each agent; etc. In this case the computer would really make the organization more understandable and give a substantial support in organizational cooperation.

Acknowledgements

We thank Maria Miceli for her thoughtful suggestions and comments.

References

[1] Becker, H.S., Notes on the concept of commitment, *American Journal of Sociology*, 66, pp.32-40, 1960.

[2] Bignoli, C., and De Michelis, G., Coordination through Multiple Media, *Proceedings of ECCE-5*, Urbino, Sept. 3-6, 1990.

[3] Bond, A.H., A computational model for organization of cooperating intelligent agents, *Proceedings of the Conference on Office Information Systems*, Cambridge, MA, pp.21-30, 1990.

[4] Castelfranchi, R., No More Cooperation, Please! In Search of the Social Structure of Verbal Interaction, in A. Ortony, J.Slack,O.and Stock (Eds.), *A.I. and Cognitive Science Perspectives on Communication*, Heidelberg, Germany: Springer, 1992.

[5] Castelfranchi, C., and Conte, R., Emergent functionality among intelligent systems: Cooperation within and without minds, *AI and Society*, 6:78-93, 1992.

[6] Castelfranchi, C., Miceli, M., and Cesta, A., Dependence relations among autonomous agents, *Preprints of the Third European Workshop on Modeling Autonomous Agents in a Multi Agent World*, August 5-7, 1991, Kaiserslautern, Germany (to appear on: Y.Demazeau, E.Werner (Eds.), Decentralized A.I. - 3, Elsevier (North Holland), 1992).

[7] Chaib-Draa, B., Moulin, B., Mandiau, R., and Millot, P., Trends in Distributed Artificial Intelligence, *Artificial Intelligence Review*, 6, pp.35-66, 1992.

[8] Conte, R., and Castelfranchi, C., Mind is not enough. Precognitive bases of social interaction, *Proceedings of Simulating Societies Symposium*, Univ. of Surrey, Guildford, April, 1992.

[9] Conte, R. Castelfranchi, C., and Cesta, A., Natural Topic-Change in Plan-Recognition, *Proceedings of EPIA-89 (Portuguese Conference of Artificial Intelligence)*, vol.2, Lisboa, Portugal, Sept. 1989.

[10] Conte, R., Castelfranchi, C., Cesta, A., and Miceli, M., Pragmatic Structures for Social Interaction, in M.C.Jackson, P.Keys, S.Cropper (Eds.), *Operational Research and Social Sciences*, Plenum, London, 1989.

[11] Cohen, P.R., and Levesque, H.J., Rational Interaction as the Basis for Communication, in P.R.Cohen, J.Morgan and M.E.Pollack (Eds.), *Intentions in Communication*, MIT Press, Cambridge MA, 1990.

[12] Davis, R., and Smith, P.G., Negotiation as a Metaphor for Distributed Problem Solving, *Artificial Intelligence*, 20 (1), 63-109, 1983.

[13] Gasser, L., The integration of computing and routine work, *ACM Transaction on Office Information Systems*, 4 (3), pp.205-225, 1986.

[14] Hammond, K.J., *Case-Based Planning*, Academic Press, N.Y., 1989.

[15] Jennings, N., On Being Responsible, *Preprints of the Third European Workshop on Modeling Autonomous Agents in a Multi Agent World*, August 5-7, 1991, Kaiserslautern, Germany (to appear on: Y.Demazeau, E.Werner (Eds.), Decentralized A.I. - 3, Elsevier (North Holland), 1992).

[16] Malone, T.W., Organizational Structure and Information Technology: Elements of a Formal Theory, MIT, Sloan School of Management, T.R.,1985.

[17] Singh, M.P., Social and Psychological Commitments in Multiagent Systems, *Preprints of the AAAI Fall Symposium on Knowledge and Action at Social and Organizational Level*, Asilomar, CA, November 15-17, 1991.

[18] Winograd, T., A Language/Action Perspective on the Design of Cooperative Work, *Human-Computer Interaction*, vol. 3, 1, 3-30, 1987.

part 4
USER INTERFACE EVALUATION

When Visual Programs are Harder to Read than Textual Programs

T. R. G. Green and M. Petre#*

* MRC Applied Psychology Unit
15 Chaucer Road, Cambridge CB2 2EF, U.K.

Institute for Educational Technology
Open University, Milton Keynes MK7 5AA, U.K.

Abstract

Claims for the virtues of visual programming languages have generally been strong, simple-minded statements that visual programs are inherently better than textual ones. They have paid scant attention to previous empirical literature showing difficulties in comprehending visual programs. This paper reports comparisons between the comprehensibility of textual and visual programs, drawing on the methods developed by Green (1977) for comparing detailed comprehensibility of conditional structures. The visual language studied was LabView, a circuit-diagram-like language which can express conditionals either as 'forwards' structures (condition implies action, with nesting) or as 'backwards' structures (action is governed by conditions, with boolean operators in place of nesting). Green (1977) found that forwards structures gave relatively better access to 'sequential' information, and Gilmore and Green (1984) showed 'backwards' structures gave relatively better access to 'circumstantial' information. These differences were supported in the present study for both text and graphics presentations. Overall, however, the visual programs were harder to comprehend than the textual ones, a strong effect which was found for every single subject, even though the subjects were either experienced LabView users or else experienced users of circuit diagrams. Our explanation is that the structure of the graphics in the visual programs is, paradoxically, harder to scan than in the text version.

1. Introduction

Large numbers of visual programming languages (VPLs) have been invented. Their proponents claim that these languages are easier to understand than textual languages (TLs). Sometimes the claim is based on ill-digested pseudo-psychology about TLs 'not utilizing the full power of the brain' (Myers, 1990, p.100); sometimes it rests on the unfortunately-named so-called 'software science', extended now into the visual domain by Glinert (1990) in an otherwise excellent paper. At other times VPLs are described as 'de-emphasizing issues of syntax and providing a higher level of abstraction' (Myers, 1990, p. 100), a more cautious claim. Comments from professional electronics designers tend to support this last view (Petre and Green, 1992), although the electronics domain is rather different from programming.

Improved legibility of VPLs is an important issue, not only because TLs are notoriously hard to read, but also because of the potential for making use of increased legibility to enable 'display-based problem solving' (Larkin, 1988; Howes and Payne, 1990). Green, Bellamy, and Parker (1987) proposed a cyclical coding model in which programs were developed a chunk at a time -- a chunk possibly but not inevitably corresponding to a 'programming plan' (Rist, 1986; Davies, 1990) -- and in which, as each chunk was prepared mentally and became ready to insert into the material that had been written so far, the programmer had to re-establish the intentions and structure of the existing material. This 'parsing-gnisrap' model was supported by detailed observations (Green et al., 1987; Davies, 1989). According to such a model, increasing the comprehensibility will certainly facilitate the programming process. As Fischer et al. (1991)

have shown, such 'talking back' from the environment is also typical of interaction with very complex environments as well as the much simpler ones studied in the laboratory. Although this may seem a glimpse of the obvious, programming languages seem to have been mainly devised for ease of generation rather than for comprehension (in so far as cognitive aspects have been considered at all).

Unfortunately, previous research on comprehensibility of VPLs has been unencouraging (Brooke and Duncan, 1980; Curtis et al., 1989). Nevertheless the developers of VPLs clearly believe in their products: could previous research be wrong or inappropriate? Perhaps so; today's VPLs are usually dataflow languages rather than the flowchart-like languages studied by Curtis et al and others; moreover, today there are commercial languages with real users, in contrast to the subjects of previous studies, who had used TLs for programming and visual notations for documentation; and finally, the dataflow model has allowed some languages to be based on familiar metaphors. Typical among these is the LabView language (Santori, 1990; Hils, 1992), the language that we used in the studies reported here. LabView is closely based on the metaphor of electronic block wiring diagrams.

In this paper we therefore examine the claims that VPLs are highly comprehensible, taking LabView as our type of the VPL. The paper builds on a preliminary paper by Green, Petre and Bellamy (1991) presenting outline data from a sample of LabView users; for various reasons, that paper presents only outline results. The present paper (a) reports results from a further sample of electronics designers, thoroughly familiar with the underlying metaphor of LabView, (b) presents full analyses, (c) relates the findings to the 'match-mismatch' hypothesis (Gilmore and Green, 1984) and the 'cognitive fit' hypothesis (Vessey, 1991), (d) presents a simple model of information-gathering from VPLs and TLs which is sufficient to account for the results, (e) relates that model to the models of graphical information-gathering by Lohse (1991) and of display-based problem-solving by Larkin and Simon (1987), and (f) concludes that future claims of advantages for notation structures and information displays should be more closely related to models of the cognitive processes of information extraction.

2. The Background

Previous work on the design of programming languages has frequently made the claim that no particular notation is universally best; rather, each notational structure highlights some kinds of information at the expense of obscuring other types. For conditional program structures within a procedural paradigm Green (1977) distinguished between 'circumstantial' (or 'taxon') and 'sequence' information, showing that nested conditionals favoured sequence information, ("Given this input, what happens?"), a claim later strongly supported by the work of Curtis et al. Conversely, Gilmore and Green (1984) showed that a more declarative programming language gave improved access to circumstantial information ("Given this result, what do we know about the input?"). Gilmore and Green postulated that across the spectrum of information structures, performance was at its best when the structure of the information sought matched the structure of the notation, and that a mismatch would lead to poor performance (the 'match-mismatch' hypothesis).

In a parallel but independent line of work, Vessey (1991) investigated the use of tables and graphs for conveying information and for problem-solving. She distinguished between the external problem representation (in our terms, that would be the available data) and the representation of the problem solving task (in our terms, that would be the problem to be solved). When the types of information emphasized in these two representations match, she asserts, "the problem solver uses processes (and therefore formulates a mental representation) that also emphasize the same type of information. Consequently, the processes the problem solver uses to both act on the representation and to complete the task will match, and the problem-solving process will be facilitated." (p. 221).

3. The Languages

Green and Gilmore used simple TLs based (at some distance) on existing full-scale languages. One of these, Nest-INE, was a sequential notation using nested conditionals with the assistance of extra cues, which had earlier been shown to assist both novices and professionals (Sime, Green and Guest 1977: Green 1977). Their other notation, which we shall refer to as And/Or, was based on a production system model, in which the structure was clearly circumstantial. Both of these languages were employed in the present experiment (Figures 1 and 2). The prevailing characteristic is that Nest-INE supports working *forwards*, from the input to the output, whereas And/Or supports working *backwards*, from the output to the input.

For the corresponding VPLs, it so happens that the LabView language is capable of expressing conditionals either as a sequential structure or as a circumstantial structure. The 'Boxes' notation (Figure 3) is unusual in being interactive -- at any one moment, the display only shows one arm of an if-then-else conditional, and the reader has to click on the True/False button to toggle the two arms of the conditional. The result is a sequential structure: given the selector condition, the appropriate part of the case structure can be quickly found. In contrast, given the output, the reader has to search through a variety of cases to find the appropriate case that will generate such output -- although, once found, reading off the input conditions is relatively easy.

The 'Gates' notation, also available in LabView (Figure 4), is seemingly a circumstantial structure. Given an output, it is relatively straightforward to work backwards through to the input side, recording the conditions as one goes; but working forwards from input to output is much harder, requiring the reader to branch backwards at each AND-gate.

4. The hypotheses

Both Brooke and Duncan (1980) and Curtis et al. (1989) found that the advantages of flowcharts over text, if any, were at the detailed level rather than at the overview level. We therefore chose to investigate detailed comprehension of conditionals. In part 1 we investigated question answering, following the lines adopted by Green (1977) and Curtis et al. (1989), in this new context of dataflow VPLs with experienced users. The question-answering task corresponds to "What does this program do?" (forwards) or "What made it do that?" (backwards). In Part 2 we investigated same-different comparisons between programs, which have not previously been studied. When both programs are in the same notation, same-different judgements correspond to such real-world questions as "How do these programs differ?"; when they are in different notations, the real-world equivalent might be "Does this program agree with this specification?".

If the supporters of VPLs are correct, then visual representations will be uniformly superior. The contrary hypothesis is that in Part 1, sequential languages will facilitate forwards questions, and that circumstantial languages will facilitate backwards questions, with no strong assertion about the difference between the graphical and textual modes. In Part 2, our hypothesis is that a mismatch between program structures (i.e. one forwards, and one backwards notation)would slow up performance.

5. Procedure

Examples of stimulus programs are shown in Figures 1-4, covering Text or Graphics crossed with Sequential or Circumstantial. The two graphical notations were derived from LabView, using exact screen images copied into SuperCard. For the Boxes notation (Figure 3), the interactive aspects of LabView were emulated in SuperCard. In half the programs there was at least one outcome that could be reached by more than one route: these are 'multi-path' stimuli. Other programs were 'single-path'.

In Part 1 of the experiment, each subject was shown a stimulus program and after a short interval was then shown either input data (for a *forwards* question) or an output result (for a *backwards* question). Responses were made by mousing on radio buttons. For a forwards question, a single mouse click was sufficient to state the action of the program. Backwards questions for single-path programs were answered by setting 6 radio buttons to appropriate values, True, False, or Irrelevant; for multiple-path programs, 12 radio buttons were set. Examples of these tasks are shown in Figure 5.

In Part 2 of the study two programs were presented side by side, and the subject responded either Same or Different, again by mousing a button. Due to poor design, the subjects in Group 1 (see below) only received comparisons in which one notation was a TL and the other a VPL. The mistake was repaired for Group 2, and all possible comparisons were made.

Two groups of subjects were recruited. Group 1 (N=5) were occasional programmers who had used LabView for their work or at least 6 months. Overall programming experience ranged from 5 to 15 years and covered a variety of languages, Basic and assembly language being the commonest. Group 2 (N=6) were very experienced programmers and designers of advanced digital electronics apparatus with no previous experience of LabView.

6. Results

6.1 Part 1: Forwards and Backwards questions

Errors were few and unsystematic. *Time scores* were transformed to logarithmic units and analysed by ANOVA with three repeated-measures factors: Modality (text vs. graphics), Structure (sequential vs. circumstantial) and Direction of questioning (forwards vs. backwards single vs. backwards multiple). Since the procedure for Part 1 was identical in the two versions of the experiment, but the subject populations differed, we treated the combined results as having an additional between-groups factor, Group (Group 1 vs. Group 2). The ANOVA revealed no interactions involving the Group factor with a significance level below 10% except Direction*Structure*Group, which reached 0.08. There was not even a main effect for Group (i.e. neither group of subjects responded significantly faster overall). It appears likely, therefore, that the results we report will hold for a wide range of subject populations.

The overall pattern of results is illustrated in Figure 6, expressed in the logarithmic units used for statistical analysis. *Match-mismatch:* Forwards questions were answered faster in notations with Sequential structure, Backwards questions were answered faster in notations with Circumstantial structure, as predicted by the 'match-mismatch' or 'cognitive fit' hypothesis. This interaction, Direction*Structure, was highly significant ($F(2, 18) = 21.77$, $p = 0.0001$). *Graphics v. Text:* Overall, graphics was slower than text. The main effect for Mode was very highly significant ($F(1,9) = 208.6$, $p < 0.0001$). Moreover, graphics was slower than text *in all conditions*: there were no high-order interaction effects, and in particular Mode*Direction*Structure was far from significance ($F(2,18) = 1.43$, $p = 0.64$). The overall geometric mean for Text responses was 35.2 seconds, as against 68.1 seconds for Graphics responses. *Size of effect:* Hayes (1981) gives a procedure for estimating ω^2, the size of an effect in terms of total variance accounted for. The three largest effect sizes were Mode, Direction, and Direction*Structure, for which ω^2 was respectively 25.0 %, 24.7 %, and 5.7%; so about 56% of the total variance was accounted for by these three. *Uniformity of effect:* For each individual subject, we compared the 8 times for text notations to the 8 times for graphics notations. The mean time for graphics conditions was greater than the mean time for text for every single subject.

6.2 Part 2: Same-Different Judgements

Two analyses were made, one of all the responses, the other only of correct responses. Because each comparison employed two notations, the response time was affected not only by the intrinsic difficulty of each notation but also by the individual combination of notations. We were particularly interested in testing for a *match/mismatch effect.* Contrast analyses were used to test the hypotheses that judgements are faster (or slower) when the modalities differ (i.e. one Graphics, one Text), or else when the structures differ (i.e. one Sequential, one Circumstantial). Neither was significant.

We also tested for an *intrinsic difficulty effect*, using linear contrasts for number of Graphics notations presented on each trial (0, 1, or 2) and number of Sequential notations (also 0, 1, or 2). Although these tests had lower power, both contrasts were significant. For Mode, $F(1, 10) = 43.9$, $p < 0.0001$, showing that the more Graphics components, the longer the response time (Figure 7); for Structure, $F(1,10) = 5.6$, $p = 0.040$, showing a weaker effect that the more Circumstantial components, the longer the response time (Figure 8). It would appear that intrinsic effects of Mode and Structure were much stronger than match/mismatch effects, and that the intrinsic difficulty of the graphics mode was the strongest effect of all. Error data were not revealing.

7. Discussion

7.1 Visual languages versus Text languages

Our first conclusion obviously concerns the plausibility of dataflow VPLs. Given these results, it can hardly be claimed that VPLs are consistently superior to TLs! The data show that the graphical notations are in fact consistently *worse* than the textual notations. Our tasks were realistic tasks, testing the level of detailed comprehension that had previously been shown to be the best point of flowchart-based VPLs. The language is reasonably successful commercially. The size of the problems we used was not unrealistic (the LabView manual itself gives examples with a comparable number of control components). Our subjects included two levels of programming experience, of familiarity with LabView, and of familiarity with the wiring-diagram metaphor. In view of all this, the poor showing of the LabView VPL shows that dataflow languages are poor at communicating this type of structure.

The problems of graphical representations showed most clearly in Part 2, the same-different comparisons, where the Gates structure stood out as hard to work with. The sight of subjects crawling over the screen with mouse or with fingers, talking aloud to keep their working memory updated, was remarkable. This structure was very difficult for our subjects, *even for Group 2, who were very experienced with it.* As a support for reasoning processes it clearly leaves a great deal to be desired.

7.2 'Match-mismatch' / 'cognitive fit'

The results also show, as expected, support for the 'match-mismatch' or 'cognitive fit' hypotheses, thereby confirming the results obtained by Gilmore and Green (1984), Curtis et al. (1989), and Vessey (1991); but the effect of the match-mismatch is less than had been expected. Similarly Sinha and Vessey (1991), comparing Lisp and Pascal as vehicles for learning recursion and iteration, also found that 'cognitive fit' only told half the story, and that there were differences between programming languages that were not explicable in those terms. In the present experiment, the reason may be caused by the problems of tracing program behaviour and gathering information in the Graphics notations.

7.3 Information structures and models of information gathering

The profound difficulties of the Graphics modes in our study seemingly contradict recent developments in display-based problem solving (Payne and Howes, 1990; Larkin, 1988; Larkin and Simon, 1987) and sophisticated models of information-gathering in tables versus graphs (Lohse, 1991).

The reason is not far to seek. The models just cited have all obtained their results by showing that the graphical presentation of data made for improved information-gathering. For instance, Lohse's extremely thorough model predicts time required to answer questions about graphs and tables by allowing for eye-scan times, time to discriminate a symbol or a shape, time to match symbols against working memory, etc. The graphical presentations allow faster scanning and faster discriminations, so the model predicts faster total times. Similar, although less well-developed, arguments lie behind the Larkin and Simon account of how diagrams improve simple problem-solving.

In contrast, the information-gathering process in a VPL is sometimes extremely complex. We illustrate a plausible (but idealised) instance in Figure 9. It would not be difficult to cast this process into the form of a computational model, but the exercise is unnecessary to make our point: *not all graphical structures are equivalent.* Program structures contain 'knots' which force working memory load on the reader. Essentially, information gathering in the Gates notation is equivalent to traversing a maze. Interestingly enough, the text structures used in this study do not contain similar knots; they make a better use of spatial topography than the graphical notations! (Figure 10)

A further conclusion, therefore, is that the study of cognitive processes involved in understanding graphs and tables should not be limited to either the match-mismatch effect, or the faster speeds of scanning and processing graphical symbols. The information structure of the graph must also be considered. In cases like these where the graphical structure contains 'knots' but the textual version does not, the supposed advantages of graphics over text will prove illusory. Instead, performance will be dominated by working memory limitations.

References

Brooke, J. B. and Duncan, K. D. (1980) Experimental studies of flowchart use at different stages of program debugging. *Ergonomics*, 23, 1057-1091.

Curtis, B., Sheppard, S., Kruesi-Bailey, E., Bailey, J. and Boehm-Davis, D. (1989) Experimental evaluation of software documentation formats. *J. Systems and Software*, 9 (2), 167-207.

Davies, S. P. (1989) Skill levels and strategic differences in plan comprehension and implementation in programming. In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers V.* Cambridge University Press.

Davies, S. P. (1990) The nature and development of programming plans. *Int. J. Man-Machine Studies* 32 461-481.

Fischer, G., Lemke, A. C., McCall, R., and Morch, A. I. (1991) Making argumentation serve design. To appear in *Human-Computer Interaction*, 6 (3 & 4), 393-419.

Gilmore, D. J. and Green, T. R. G. (1984) Comprehension and recall of miniature programs. *Int. J. Man-Machine Studies* 21, 31-48.

Glinert, E.P. (1990). Nontextual programming environments. In Chang, S.-K. (Ed.) *Principles of Visual Programming Systems.* Prentice-Hall.

Green, T. R. G. (1977) Conditional program statements and their comprehensibility to professional programmers. *J. Occupational Psychology*, 50, 93-109.

Green, T. R. G., Bellamy, R. K. E. and Parker, J. M. (1987) Parsing-gnisrap: a model of device use. In G. M. Olson, S. Sheppard and E. Soloway (Eds.) *Empirical Studies of Programmers: Second Workshop.* Ablex.

Green, T. R. G., Petre, M. and Bellamy, R. K. E. (1991) Comprehensibility of visual and textual programs: a test of 'Superlativism' against the 'match-mismatch' conjecture. In J. Koenemann-Belliveau, T. Moher, and S. Robertson (Eds.), *Empirical Studies of Programmers: Fourth Workshop.* Norwood, NJ: Ablex. Pp. 121-146.

Hils, D. D. (1992) Data flow visual programming languages. *J. Visual Languages and Computing*, in press.

Howes, A. and Payne, S. J. (1990) Display-based competence: towards user models for menu-driven interfaces. *Int. J. Man-Machine Studies*, 33, 637-655.

Larkin, J. H. (1988) Display-based problem solving. In D. Klahr and K. Kotovsky (Eds) *Complex Information Processing: the Impact of Herbert A. Simon.* Hillsdale, NJ: Erlbaum.

Larkin, J. H. and Simon, H. A. (1987) Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science*, 11, 65-100.

Lohse, J. (1991) A cognitive model for the perception and understanding of graphs. *Proc. CHI 91 ACM Conf. on Human Factors in Computing Systems,* pp 137-144. New York: ACM

Myers, B. (1990). Taxonomies of visual programming and program visualization. *J. Visual Languages and Computing*, 1, 97-123.

Rist, R. S. (1986) Plans in programming: definition, demonstration, and development. In E. Soloway and S. Iyengar (Eds.), *Empirical studies of programmers.* Norwood, NJ: Ablex.

Petre, M. and Green, T.R.G. (1992) Requirements of graphical notations for professional users: electronics CAD systems as a case study. *Le Travail Humain* , 55(1), 47-70

Santori, M. (1990) An instrument that isn't really. *IEEE Spectrum*, August. Pp 36-39.

Sime, M. E., Green, T. R. G. , and Guest, D. J. (1977) Scope marking in computer conditionals – a psychological evaluation. *Int. J. Man-Machine Studies*, 9, 107-118.

Sinha, A. and Vessey, I. (1991) Cognitive fit: an empirical study of recursion and iteration. Unpublished MS, Dept. of Accounting and MIS, Pennsylvania State Univ.

Vessey, I. (1991) Cognitive fit: a theory-based analysis of the graphs versus tables literature. *Decision Sciences*, 22, 219-240.

Note

LabView is a trademark of National Instruments Inc. SuperCard is a trademark of Silicon Beach Software Inc.

We are very grateful to Rachel Bellamy, who helped set up the experiment, and to David Hendry, who improved the quality of this paper.

```
if high :
   if wide :
      if deep : weep
      not deep :
         if tall : weep
         not tall : cluck
         end tall
      end deep
   not wide :
      if long :
         if thick : gasp
         not thick : roar
         end thick
      not long :
         if thick : sigh
         not thick : gasp
         end thick
      end long
   end wide
not high :
   if tall : burp
   not tall : hiccup
   end tall
end high
```

Figure 1: an example of the Nest-INE notation (Text, Sequential).

```
howl :    if honest & tidy & (lazy | sluggish )
laugh :   if honest & tidy & ¬ lazy & ¬ sluggish
whisper : if honest & ¬ tidy & ( nasty & greedy | ¬ nasty & ¬ greedy )
bellow :  if honest & ¬ tidy & nasty & ¬ greedy
groan :   if honest & ¬ tidy & ¬ nasty & greedy
mutter :  if ¬ honest & sluggish
shout :   if ¬ honest & ¬ sluggish
```

Figure 2: an example of the And/Or notation (Text, Circumstantial).
The conditional structure shown is logically equivalent to the structure shown in Figure 1, with suitable change of labels (e.g. Bellow = Roar).

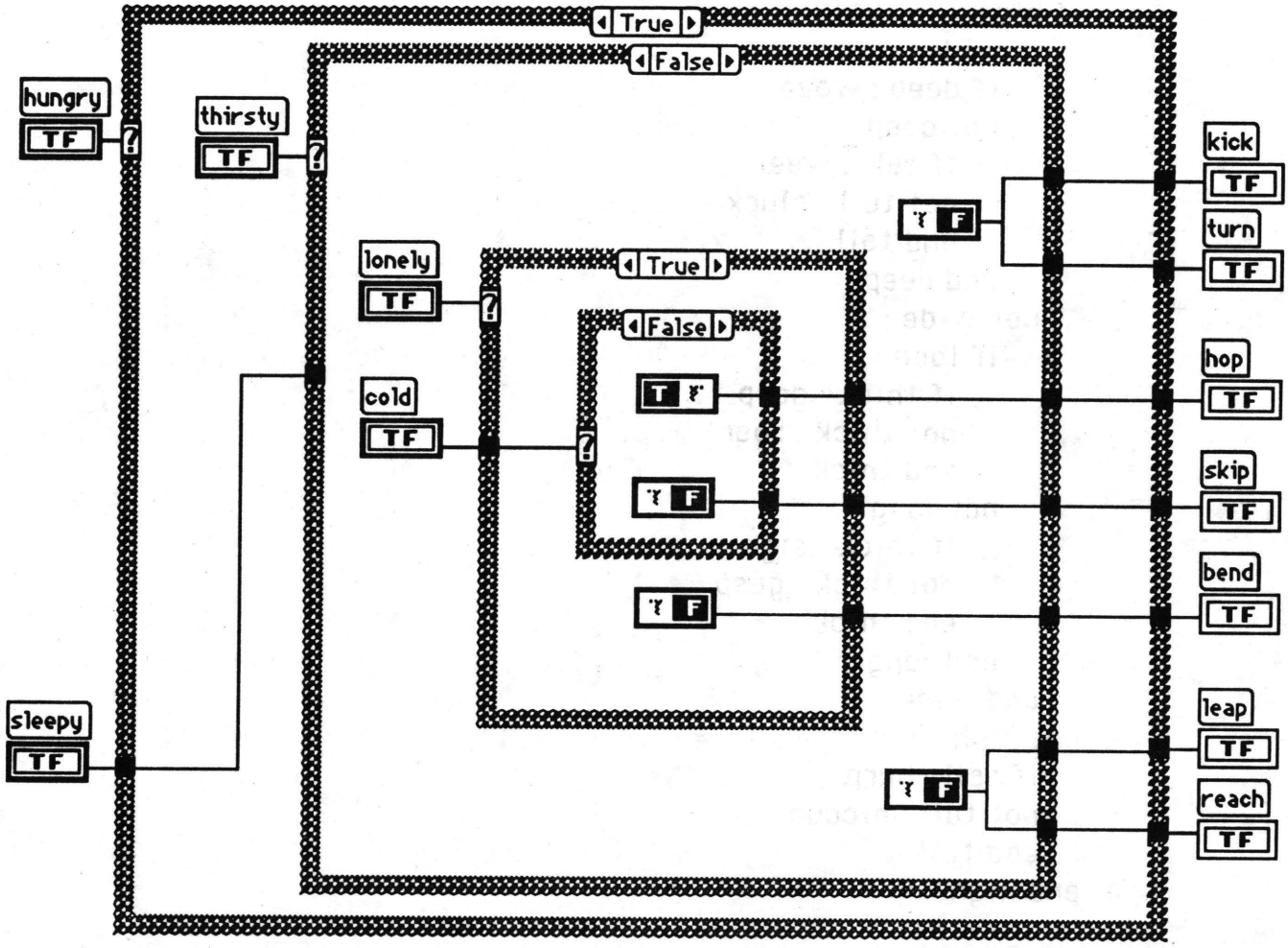


Figure 3: example of the Boxes notation (Graphical, Sequential).
This notation is interactive. Mouse clicks on the true/false buttons toggle between the true and false arms of the conditionals. Only one arm is visible at one time.

The figure shows the state of the display for Hungry = true, Thirsty = false, Lonely = true, Cold = false. The value of Sleepy is irrelevant in these conditions, although it is relevant in other circumstances (cf. Figure 1, where if High is false the values of Wide, Deep etc. are irrelevant). The output is shown by sending a boolean value to each possible output. In this figure, the output is Hop.

The conditional structure shown is is both logically and structurally equivalent to the structure shown in Figure 1, with suitable change of labels (e.g. Hop = Roar).

The notation is exactly as used in LabView, although the interactive component was achieved by writing a SuperCard emulation of LabView's behaviour.

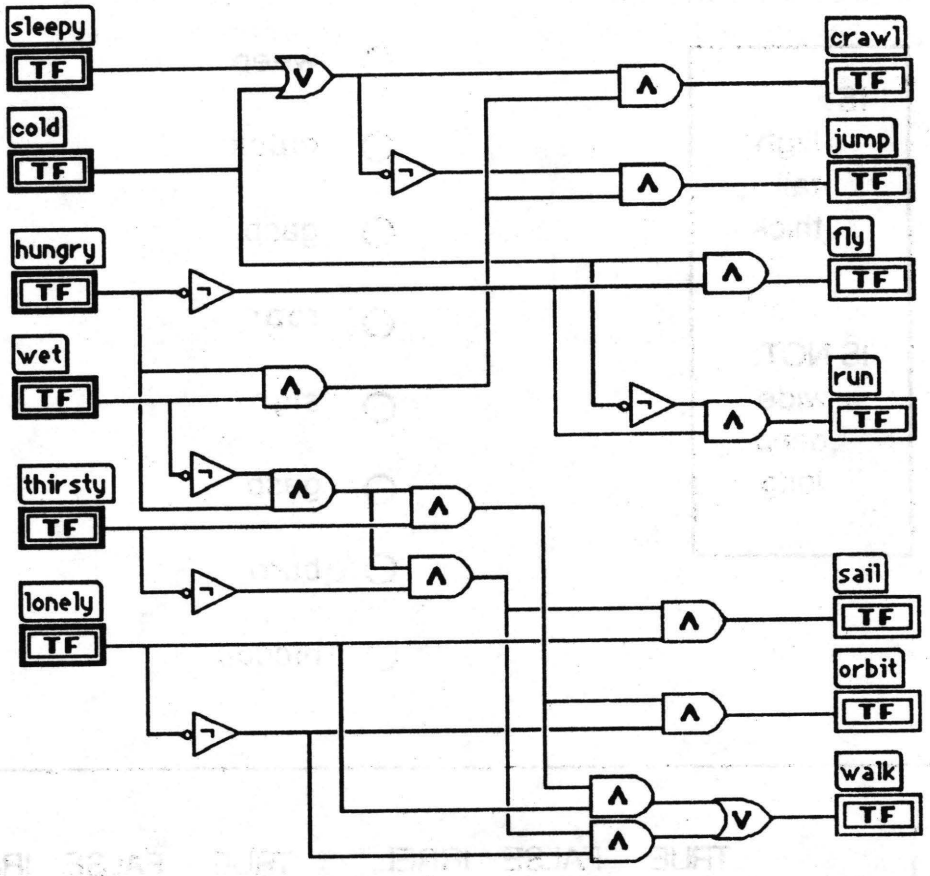


Figure 4: example of the Gates notation (graphical, Circumstantial). As in Figure 3, input comes from the left (Sleepy, Cold, etc) and proceeds to the right. The gates perform the operations of AND (shown as $\wedge$), OR (shown as $\vee$) and NOT (shown as $\neg$).

The conditional structure shown is both logically and structurally equivalent to the structure shown in Figure 2, with suitable change of labels (e.g. Orbit = Bellow), and is logically equivalent to the structure in Figure 1, with suitable change of labels (e.g. Orbit = Roar).

The notation is exactly as used in LabView.

There is considerable freedom in the layout of Gates structures. Our layouts were designed by a professional circuit designer following accepted practice.

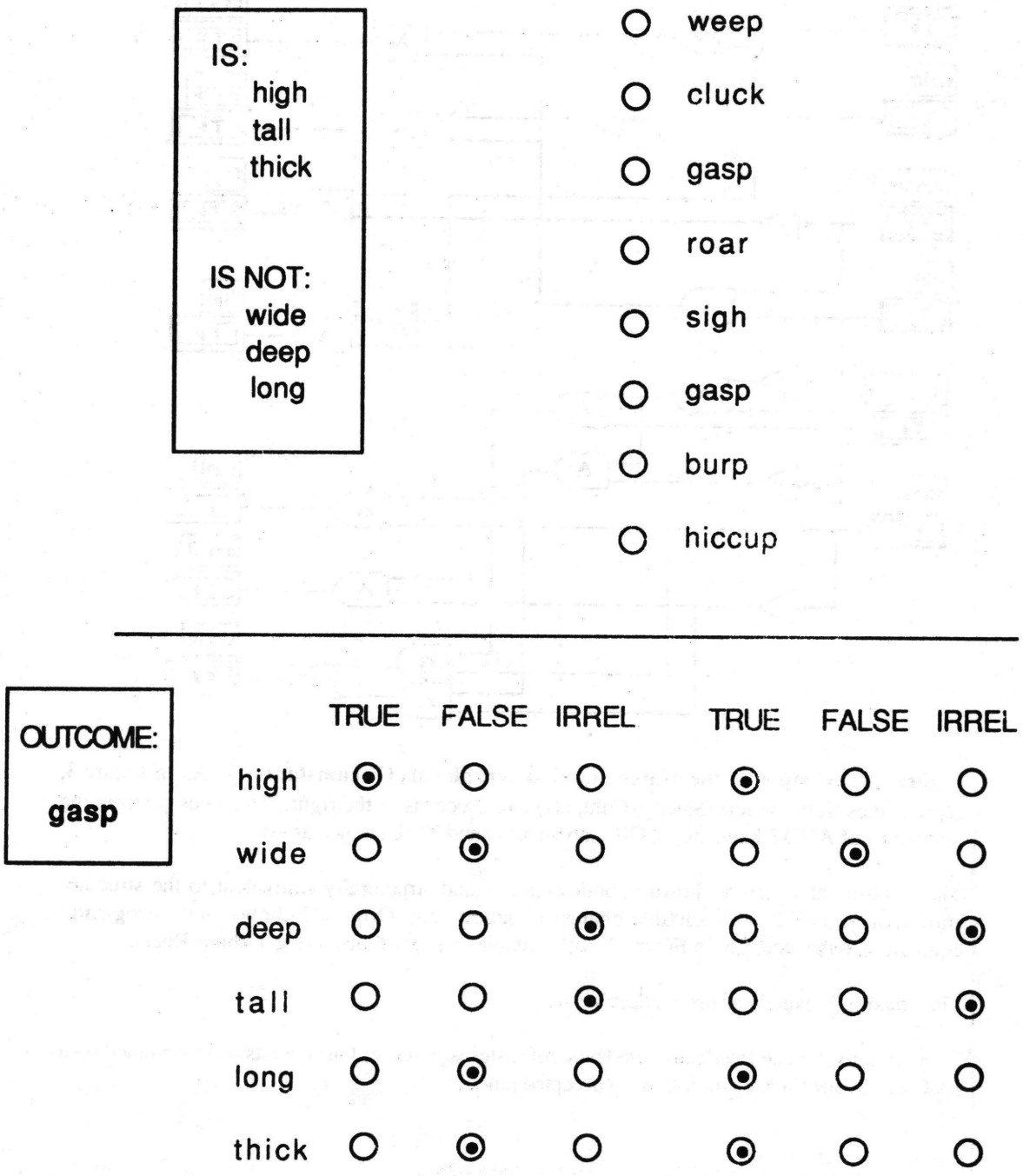
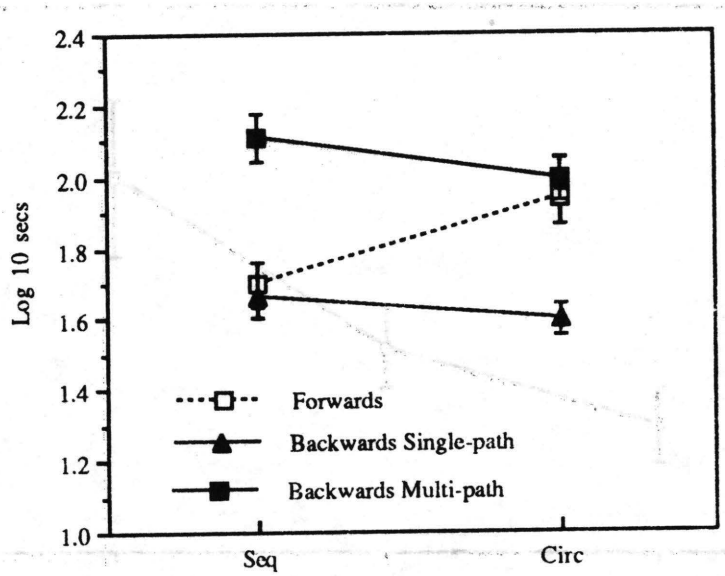


Figure 5: Problem presentation for a forwards question (above) and a backwards question (below). The same presentation was used for each type of notation, appearing below the program on the screen. The circles represent Macintosh 'radio buttons', which were clicked with the mouse to turn them on and off. The backwards questions contained two sets of answer buttons (as shown here) for the multi-path programs, but only a single set for the single path programs. The buttons were initialised to 'IRREL'; this figure shows a correct answer to the question, given the program shown in Figure 1.

(a) The two Graphics notations (Boxes and Gates):



(b) The two Text notations (Nest-INE and And/Or):

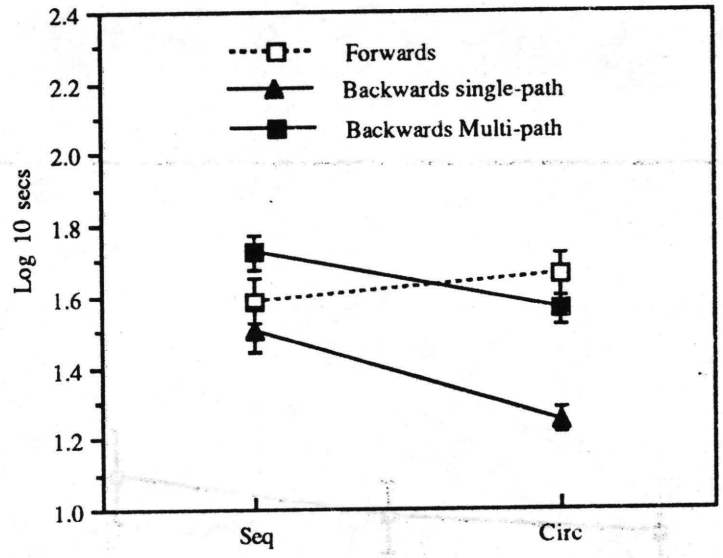


Figure 6: Response times in Part 1, pooled over both groups, showing the Direction * Structure interaction: for Forwards questions, notations with Sequential structure (i.e. Nest-INE and Boxes) are faster than Circumstantial, but for Backwards questions Circumstantial structures are faster. Note that times for the Graphics notations are slower.

Above: response times to programs expressed in Sequential Graphics ('Boxes') and Circumstantial Graphics ('Gates'). **Below:** response times to programs expressed in Sequential Text ('Nest-INE') and Circumstantial Text ('And/Or').

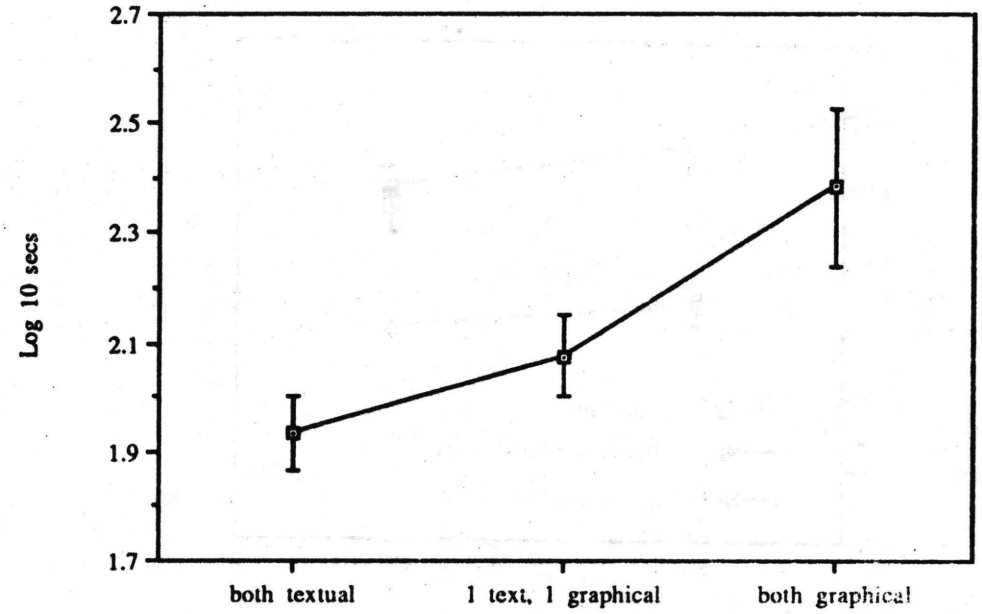


Figure 7: Intrinsic effect of Mode in same-different judgements. Comparing two textual notations was fastest; comparing two graphical notations was slowest. The difference is surprisingly great.

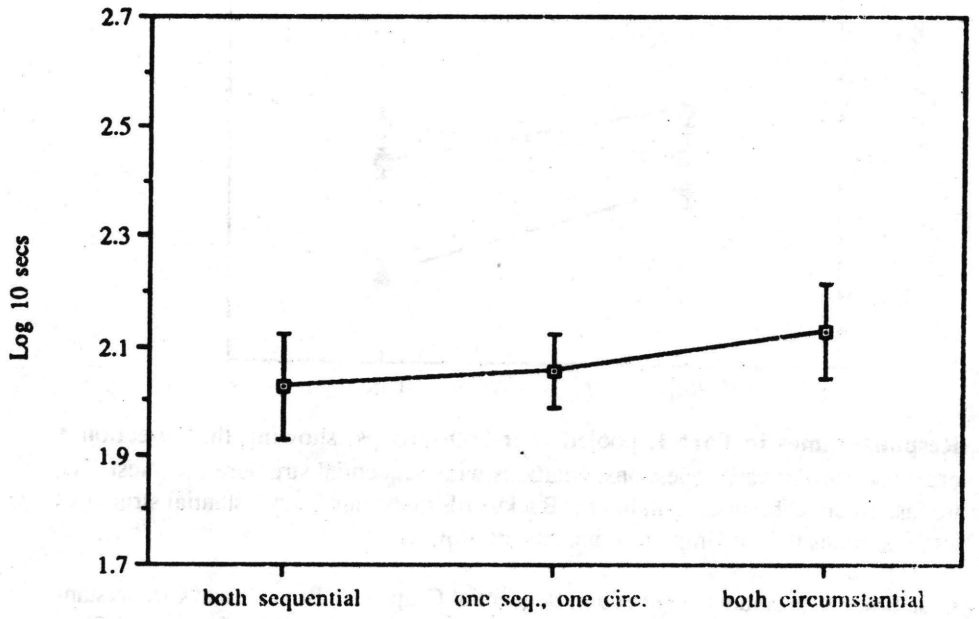
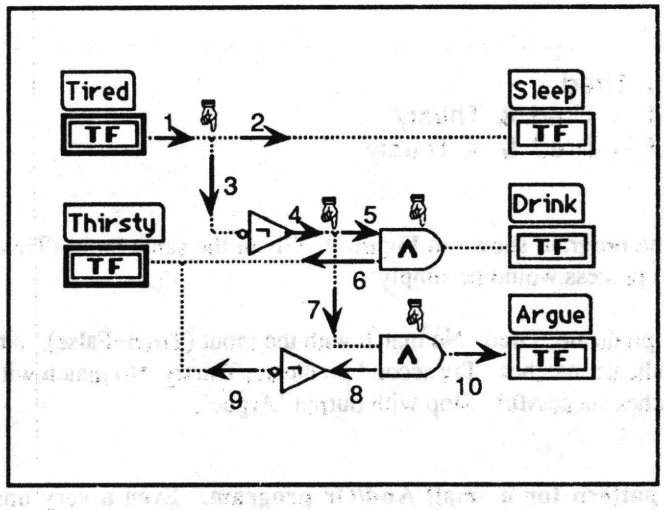


Figure 8: Intrinsic effect of Structure in same-different judgements. The comparisons in which both notations were Sequential were slightly faster than those in which one or both were Circumstantial.



Consider answering a forwards question for a small Gates program (see figure). Suppose the input is Tired=False, Thirsty=False. A plausible account of the cognitive process (agreeing with detailed observations of subject strategies made in the experiment) would be:

Take first truth-value, Tired=False. Scan to find Tired in program. Propagate False along the wire. Place finger at choice point and consider each direction in turn.

- Propagate horizontally first. On reaching next component, Sleep, abandon this branch, since we haven't sent a True to an output yet.

- Now propagate downwards. On reaching the invertor, start to propagate True. On reaching junction, place finger at choice point and consider each direction in turn.

  - Propagate horizontally first. On reaching and-gate, place finger on it and start to search backwards. Search reaches input Thirsty. Look up truth-value, which is False. Return to most recent finger; abandon this branch since the conjunction is false.

  - Return to previous finger. Start to propagate downwards. On reaching and-gate place finger on it and start to search backwards. On reaching invertor, remember to invert next signal found. Search reaches Thirsty which is set to False so invertor is set to True. Return to most recent finger and continue propagating True. Search reaches Argue. Stop with output 'Argue'.

Figure 9: Typical search pattern for a small Gates program, following typical strategies observed in our subjects. The task is to discover the output, given the input Tired=False, Thirsty=False. Numbers in the diagram show order of examining segments; arrows show direction of travel; pointing hands show junctions where fingers (mental or physical) must be placed. Even in this extremely simple case the process is not trivial.

Sleep: if Tired
Drink: if ¬ Tired & Thirsty
Argue: if ¬ Tired & ¬ Thirsty

This is the And/Or version of the program shown in Figure 9. Given the same input (Tired=False, Thirsty=False) a plausible account of the solution process would be simply:

Try first line. Try first predicate, Tired. No match with the input (Tired=False). Abandon this line.
Try next line. First predicate matches. Try second predicate, Thirsty. No match with input. Abandon.
Try next line. All matches successful. Stop with output 'Argue'.

Figure 10: Search pattern for a small And/Or program. Even a very unsophisticated algorithm, such as 'try each line in turn', is very easy to apply to this notation, with a minimum of working memory required.

User- vs. System-Initiated Data Entry into Expert Systems: a Behavioral Perspective

C.M.M. Hurts

Leiden University, The Netherlands

ABSTRACT

This study reports an experiment that studied two different interfaces for entering data into a legal expert system in relation to system transparency. The interfaces were assigned to two groups of beginning expert system users and differed in terms of the amount of structure provided to the user, inviting him either to adopt a goal-driven style of data entry with little entry freedom, or a data-driven style of entry with more entry freedom. **Transparency** was behaviorally defined as the ability of users to **learn** from the system's problem solving capability. It was hypothesized that the effect of interface type on transparency would depend on the phase of the experiment, with users learning more with the system-driven interface at the beginning and with users learning more with the user-driven interface at the end of an experimental session. The data showed that, although both groups learned over the duration of the experiment, there was only a slight advantage for the system-driven interface group at the end of the experiment. The meaning and practical use of these findings is discussed, also in relation to the accuracy of entering data with each of the two interfaces.

1. Background

Classical expert systems control when and what the user must communicate. For example, medical diagnosis systems expect users to provide data about symptoms in a way that is determined by the system. These passive user roles are based on the belief that system expertise is complete and/or does not have to be communicated to the user. Of course, this assumption may not be (and usually is not) true for one of the following reasons (see also Eason, 1991; Roth et al., 1987; Lehner & Kralj, 1988):

a. The expertise can be formalized only partially and provides correct solutions only for routine cases. Human input is required to supplement the expert system in less routine cases. Unfortunately, the expert system itself is not always able to recognize which cases are routine and which ones are not.

b. In *synthetic* expert system tasks (such as most planning tasks) the user may not be able to state the precise characteristics of the desired solution until he has seen (a first attempt at) the solution. Satisfactory solutions can therefore only be achieved by co-operative and interactive refinement of the solution by both user and system. This often requires the system to match its problem solving style to the user's.

c. Users are novices, wanting to (actively) learn from the expert system by doing, that is, by using the expert system as a coach for learning to solve problems.

In order to overcome these limitations, the traditional types of interaction with expert systems have been expanded to include, amongst others, explanation facilities, knowledge base editors, and (intelligent) tutoring systems. These forms of interaction are sometimes called **meta-communication** (Waern, 1989). The resulting visibility of the expert system's internal structure and functioning to the user may also be called **system transparency** (Maass, 1983).

In this study the issue of system transparency was approached from an active (human) learning perspective and from a data-entry perspective. These perspectives are explained below.

1. It was assumed that, when teaching or coaching is designed to stimulate **learning by doing**, design guidelines for effective *learning* situations are (at least partly) also design guidelines for effective *using* situations (where users may want to compare the system's knowledge to their own). Learning by doing is a teaching strategy with a long tradition and seems particulary suitable for rule and concept learning (Anzai & Simon, 1979). In the present study the additional claim is made that transparent interfaces from the point of view of active learning often are also transparent interfaces from the point of view of using the expert system.

2. In the present study attention was focussed on two different modes of **data entry**, that is, means by which users can enter case descriptions into a legal expert system. The aim was to compare these modes in terms of their ability to match the problem solving style naturally preferred by the user. It was assumed that the greater this match, the greater the ability of the system's (expert) answers to reinforce knowledge structures being built by the (learning) user.

2. Problem statement

In this study an expert system was used that reasoned backward from the goal (legal means, such as appeal, available to defendants in criminal court) to the "problem facts" (i.e., characteristics of the defendant and his trial, also called the "case model"). **System-driven data entry** is here defined as the style of data entry that amounts to the system asking the user questions (see Figure 1). Here the sequence and nature of the questions is embedded in and is determined by the inference engine of the expert system. In this way, each question asked reflects the current data base of the system (what facts are

known) and, implicitly, the reason (goal) for the question (through the wording of the question or the timing of the question). Questions not relevant to the problem at hand are not asked. In the present study incorrectly answered questions (given a number of predefined, "objective", case descriptions, handed out on paper) were fed back to the user, inviting him to try the question once more (see also next section).

On the other hand, with **user-driven data entry** the sequencing aspects of the data entry task are left entirely to the user. In this study it involves the user filling in a computer-form containing all descriptive items that ever might be relevant for cases of the type discussed (see Figure 2). More precisely, the user only needs to mark one or several categories (e.g., "offence" or "misdemeanor") of one or several dimensions (e.g., "type of sentence"). Which categories and which dimensions are marked is up to the user with the exception of some combinations of categories that are (logically) impossible (e.g., a defendant can not be accused of an offence and a misdemeanor at the same time). In the latter case the user is warned by the system and asked to correct his entry. If the entry is correct, the expert system is activated, using some or all of the data that were entered before. By inviting the user to mark descriptive items, this mode of data entry stimulates the user to "recognize" the relevant case data by just looking at the form.

In order to prevent users from entering incorrect data, input was checked against the "objective" case descriptions and, in case of errors, they were invited to correct them. This also applies to "don't know" type of errors (system-driven entry) or omissions (user-driven entry). In the case of user-driven data entry, some selected categories may not be relevant for the problem solving process given the previous input, even though these categories may apply to the case at hand. In the present study this type of input was accepted by the expert system as correct.

It was realized that there are more types of data entry than the ones mentioned above. However, most of these are either mixes of user-driven and system-driven styles or utilize (semi)natural language input by the user. The present study focussed on (more or less) "pure" system-driven and user-driven styles. Also, language interfaces, despite their appearance, do not really change the nature of the data entry task. For these reasons these alternative styles were not considered separately in this study.

With respect to **system transparency** (measured by the degree to which users can learn from the expert system's answers), these two modes of data entry were hypothesized to have the following effects.

1. **With user-driven data entry users learn better from the expert system's answers at the end of an experimental session than at the beginning.** User-driven data entry matches a data-driven style of problem solving. According to the literature, this style of problem solving is typically found with expert problem solvers, but not with novices (Anderson, 1983; Patel & Groen, 1986). Moreover, because of greater entry freedom, a user-driven style makes it possible for users to deliberately select more descriptive items than deemed necessary for solving the problem. This enables him to enter in a hypothesis-testing mode of problem

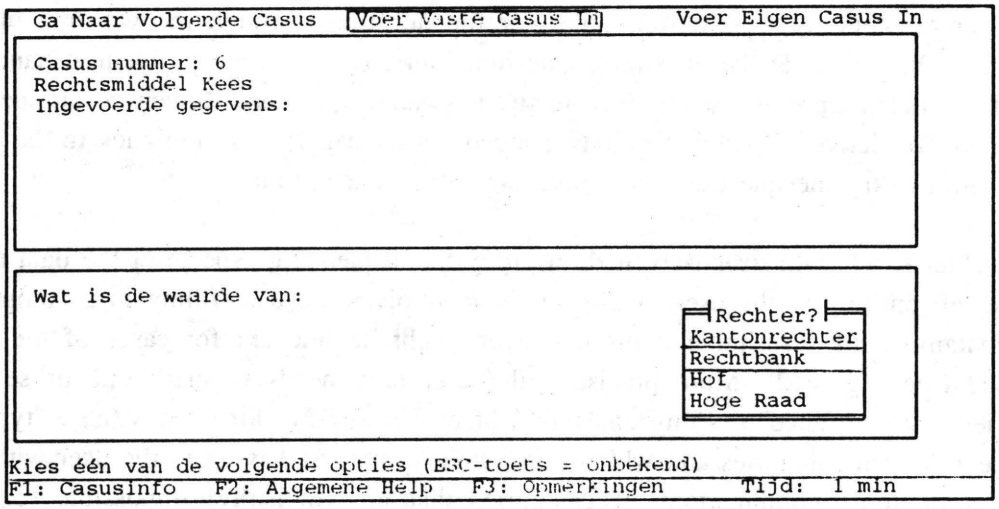


Figure 1. Typical screen of system-driven interface, showing an expert system question and a menu of user answers.

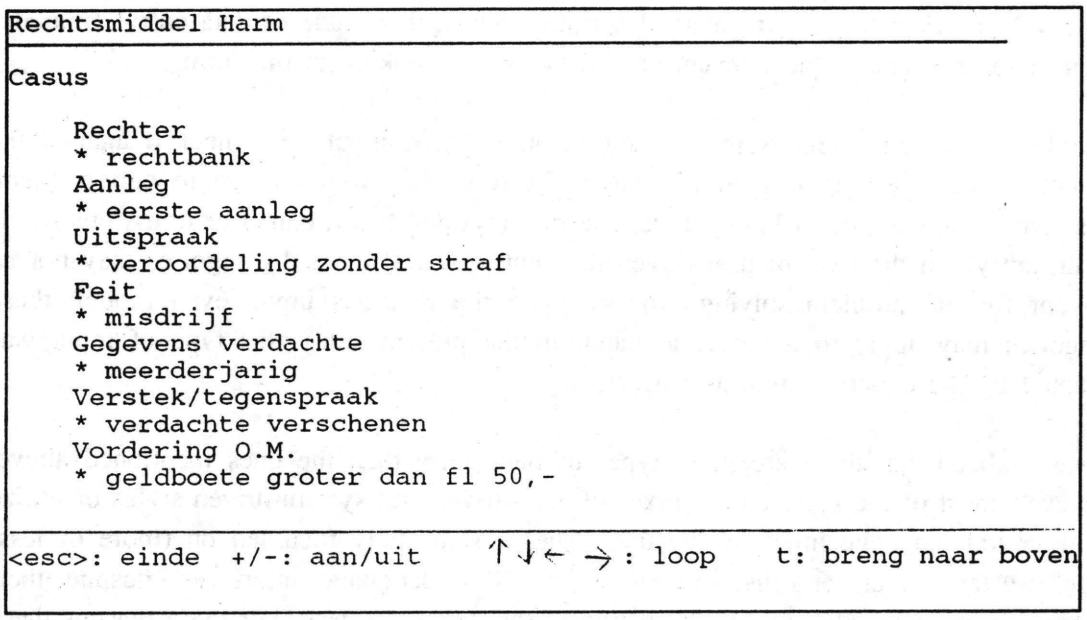


Figure 2. Summary screen for data entry with user-driven interface, showing selected item types and item type values (the latter preceded by an asteriks).

solving (with the user interrogating the system instead of the other way around), a behavioral style also typically found with experienced users. Of course, greater entry freedom carries in it the risk of the user inadvertently providing data not strictly required by the expert system. However, this was believed to influence users at the beginning, but not at the end of an experimental session.

(It should be noted that the entry freedom was limited due to erroneous input being corrected by the system; for this reason users could not deliberately change the data, that is, change the nature of the case they were having the system solve.)

2. **With a system-driven mode of data entry users learn better from the expert system's answers at the beginning of an experimental session than at the end.** System-driven data entry matches a goal-driven style of problem solving. According to the literature, this style of problem solving is typically found with novice problem solvers, but not with experts (Anderson, 1983; Patel & Groen, 1986). Moreover, this style keeps the user from entering data that are irrelevant for the problem solving process (given the previous input). This is believed to help novice users even more. At the same time, this style makes it more difficult to enter in a hypothesis-testing mode, because users cannot enter arbitrary data or in arbitrary order (not even if the data apply to the case). Novice users are believed not to suffer from the latter limitation.

Separate from the ability of users to learn from the expert system's answers, there is a question of how easy (how fast, how many errors) it is to enter data in either mode of entry. The structured nature of the system-driven mode (only relevant questions asked) leads us to expect that data entry in this mode is easier than in the user-driven mode, despite the fact that the word "user-driven" suggests otherwise. However, this effect can be expected to disappear as more experience is gained in the data entry task (allowing users in the user-driven mode to become efficient data enterers themselves). Therefore as a third hypothesis we state:

3. **Data entry is more accurate and faster with the system-driven interface than with the user-driven interface.**

The importance of studying data entry accuracy under the two interfaces is twofold:

1. More accurate data entry is valuable in itself: even though the present study did not focus on the primary task of entering data, for most expert systems in the real world this is an important (if not the only) task .

2. Entry accuracy may be correlated with the degree to which the user can learn from the expert system's answer. This would indicate that there may be other factors (factors unrelated to the mode of data entry, e.g., individual differences) underlying the ability to learn the expert system's knowledge.

3. Method

Thirty-two law school students were asked to solve twenty-seven predefined legal cases taken from the domain of criminal law. The task consisted of assigning one of several types of appeal to the defendant appearing in the case description. In 15 of these cases (the so-called "learning" cases), case data had to be entered to the expert system first. Subsequently, the student was asked to predict the system's answer, after which the expert system told the student its solution. On the learning cases the user was given feedback on

both data entry performance and the quality of the prediction, so that continuous learning was made possible. After data entry errors the student was given a maximum of two other attempts to correct these errors. In any case, the entry had to be correct (or was corrected by the system) before the prediction task could start.

Data entry performance and prediction performance were logged automatically for later analysis. Of the twenty-seven cases, the remaining 12 cases were so-called "test" cases. The only difference with the learning cases was that data did not have to be entered explicitly into the system for these cases. In this way, differences between the two groups in prediction performance could be attributed solely to different amounts of "learning" on the learning cases, and not to spurious features inherent in the data entry task itself, such as the extent to which the correct solution is suggested by the format of data entry. The test cases were grouped in three clusters of 4, interleaving with three other clusters of 5 learning cases as indicated in Figure 3.

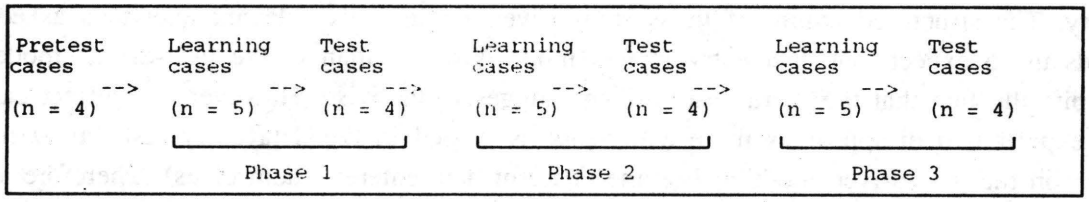


Figure 3. Sequence of experimental events.

Each combination of 5 learning cases and 4 test cases corresponds to a "phase" of the experiment. By comparing results for these three phases, differences due to experience could later be assessed.

All students were second or third year law school student. They all had followed an introductory course in criminal law. In order to ensure that none was an expert at the subject matter of legal means, none was allowed to have criminal law as a major (specialization). They were divided into two groups, one working solely with a system-driven interface, the other working solely with a user-driven interface. Before the beginning of the experiment, a pretest was administered consisting of 4 cases on which no data had to be entered explicitly and on which no feedback was given. Any differences between the two groups in pretest performance would have to be accounted for later on when analyzing the main results.

Dependent variables were:

1. Data entry errors on the learning cases (number of erroneous attempts at one complete set of entries for a case, with a maximum of three attempts per case);

2. Time taken to complete data entry on the learning cases in minutes (approximated by the total amount of time taken to complete the learning cases), and

3. Prediction errors on the test cases (number of erroneous attempts to predict the expert system's solution, with a maximum of three attempts per case).

4. Results

Data were analyzed for all dependent variables through a split-plot design with one between-subjects variable (system-driven vs. user-driven, also called GROUP) and one within-subjects variable (first, second or third cluster of learning and/or test cases, also called PHASE). The analyses for unadjusted scores are summarized in Table 1 and 2.

Table 1. Descriptive statistics for prediction errors, broken down by GROUP and PHASE.

Condition	Unadjusted Mean	Adjusted Mean	Standard Deviation (Unadjusted)	N
Phase 1				
System-driven	5.31	5.52	1.03	13
User-driven	5.37	5.16	1.17	19
Phase 2				
System-driven	6.92	7.13	1.12	13
User-driven	7.11	6.90	1.63	19
Phase 3				
System-driven	4.23	4.44	0.44	13
User-driven	4.84	4.63	1.26	19

Table 2. Summary table of analysis of covariance for prediction errors (ANOVA figures - without covariates - indicated by "Unadj.").

Source	SS	MS	Df	F	P > F
Between subjects					
Group (A)	0.03	0.03	1	0.02	0.88
Unadj.	1.88	1.88	1	1.06	0.31
Subj.with.A	41.07	1.47	28		
Unadj.	53.11	1.77	30		
Within subjects					
Phase (B)	98.71	49.35	2	38.15	0.00
B*A	1.29	0.65	2	0.50	0.61
B*Subj.with.A	77.62	1.29	60		
Covariates	12.04	6.02	2	4.10	0.03

Table 1 shows that prediction performance was better for the system-driven group, especially in the third phase of the experiment. However, the effect of GROUP was not significant, $P > 0.05$, as can be seen in Table 2. As stated before, it was expected that the user-driven group would outperform the system-driven group at the end of the experiment and that the inverse situation would apply to the beginning of the experiment. However, not only was a main effect of GROUP not observed, but neither was an interaction between GROUP and PHASE ($P > 0.05$), as can also be seen in Table 2.

Table 2 also shows a main effect of PHASE on prediction errors, $F(2,60) = 38.15$, $P = 0.00$. Inspection of the means in Table 1 indicates that this effect amounts to a significant drop in error rate, especially going from the second to the third phase of the experiment. This, of course, reflects the effect of practice, as was expected. In combination with the lack of a significant effect of GROUP, this seems to indicate that beginning expert system users are able to learn from the expert system's answers, **irrespective** of the mode of data entry.

The data presented so far are not adjusted for covariates. Two potential covariates are prediction scores on the pretest (also measured by number of prediction errors) and data entry errors (measured on the learning cases only). Analysis of covariance allows one to adjust the prediction scores on the test cases for correlations with the pretest scores and/or the data entry errors. Moreover, such analysis adjusts for the possibility of the two interface groups initially not being equal with respect to these two covariates. The results of this analysis are also contained in Tables 1 and 2.

After adjustment, the mean prediction errors for the two interface groups are even less different than they already were ($P > 0.05$). Clearly, there is no direct (adjusted) effect of GROUP on prediction errors. However, Table 2 shows a significant correlation between the combination of covariates and prediction errors, $F(2,28) = 4.10$, $P = 0.03$. Closer analysis shows that this effect is largely due to a high correlation between data entry errors on the learning cases and prediction errors on the test cases (not shown in the tables). Apparently, the more data entry errors users make, the greater the probability of them making prediction errors.

Although this correlation is in itself not surprising (see also the discussion in the next section), the effect it has on the main effect of GROUP is. At least it suggests that the system-driven interface has lower data entry errors associated with it than the user-driven interface, otherwise the correlation would have been expected to raise the effect of GROUP, rather than lower it. In order to test this assumption, data entry errors and durations of the two interface groups were also compared (learning cases only). The descriptive statistics are contained in Table 3.

Table 3. Descriptive statistics for data entry errors and durations (in minutes), broken down by GROUP.

Condition	Mean	Standard Deviation	N
Errors			
System-driven	5.69	2.81	13
User-driven	10.26	8.29	19
Duration			
System-driven	35.91	7.75	13
User-driven	53.11	13.61	19

As can be seen, the system-driven group made fewer data entry errors than the user-driven group, as was expected. This effect turned out to be marginally significant, $F(1,30) = 3.64$, $P = 0.07$. Apparently, entering data with the user-driven interface was more

difficult, as is also shown by the differences in total data entry durations, $F(1,30) = 16.90$, $P = 0.00$. This effect, in combination with the significant correlation between data entry errors and prediction errors, helps to explain why the main effect of GROUP on prediction errors was suppressed after adjustment for the covariates.

5. Conclusions and Discussion

In this study the impact of the format by which users enter data into an expert system was assessed in relation to the ease by which they can learn from the domain knowledge of the expert system. It was assumed that interfaces that allow effective learning of the system's expertise often allow also effective supervision or modification of this expertise, thereby partly satisfying the requirements of **transparent** interfaces.

Two types of data entry interface were compared in an experiment in which students entered a case that was described on paper, after which they were asked to predict the expert system's solution. The first type is called a **system-driven** interface and asks the user questions in a format and sequence that follows from the expert system's goal structure, thereby focussing the user's attention on the *process* of problem solving as conducted by the system. The other type is called a **user-driven** interface and allows the user to enter data by selecting descriptive categories that are assumed to apply to the case at hand **before** the system starts solving the case. Here the user is free to select any categories and in any order.

It was expected that at the beginning of the experiment users learn more with the system-driven interface and that at the end of the experiment users learn more with the user-driven interface.

The results show that type of data entry interface did not have a major impact on students' ability to learn from the system's expertise; any effects found were in favor of the system-driven interface and then only in the latter phase of the experiment. Although students' knowledge of the expert system's rule base improved significantly as the experiment progressed, students did not learn differently with the two interfaces (the interaction between GROUP and PHASE was not significant), contrary to what was expected.

Possibly, effects would have been more pronounced if the experiment would have continued longer. In other words, perhaps the experiment lasted too short to allow differential learning to take place for the two interfaces. It should be noted that the average experiment lasted only a little over an hour. This may not have been enough for the effect of the system-driven interface to be observed (which was assumed to match the goal-driven style of problem solving often found with novices). The expected advantage of the user-driven interface was discussed in terms of the data-driven style of problem solving often found with experts. Clearly, even if one hour of experimentation is enough to allow any effect of interface type on novices to be observed, it certainly does not make one an expert. The effects of longer experiment durations (or, possibly, of longitudinal designs) on the results remain an issue for future research.

As another explanation, the goal structure of the expert system that was supposed to underly the system-driven interface may not have been visible enough to the user. Similarly, not all advantages of the user-driven interface, that were expected at the end of the experiment, may have been present. For instance, the greater entry freedom under the user-driven mode, that was supposed to stimulate hypothesis-testing behavior, was limited due to the system having the user correct (or correcting itself) any entry errors *before* the expert system was activated. Hence, even though users could deliberately enter other data than "prescribed" by the predefined cases on paper, they could not assess the effect this had on the expert system's solution, as they were forced to stick with the predefined cases. Tentative though these explanations may be, they seem to indicate the lack of guidance or support in the data entry phase of the present experiment, aimed at clarifying to users how to (better) use the data entry interfaces.

Despite these somewhat negative and inconclusive findings, other results show that the two data entry interfaces *did* have a (marginally significant) effect on user behavior: users made fewer data entry errors and needed less time to enter data with the system-driven interface than with the user-driven interface. Assuming that data need to be entered accurately and quickly into expert systems, the system-driven style of data entry clearly should be preferred over the user-driven style.

As data entry errors on the learning cases were also correlated with prediction errors on the test cases, one might expect the results to show an indirect effect of interface type on prediction errors (i.e., more errors with the user-driven interface). However, there was no strong evidence for such effect, even though the mean error rate for the user-driven group was higher, especially at the end of the experiment. Apparently, with the user-driven interface users were able to somehow prevent the greater entry error rate to have a significant influence on their prediction errors (even though *within* each interface group this correlation was significant). As one explanation, it should be noted that all students received feedback on entry performance: perhaps students in the user-driven group were able to use this feedback very well, thereby preventing their prediction error rate from increasing too much.

Of course, most expert systems used in real life do not have the capability to provide users feedback on their entry performance; in that case accurate data entry is not only important in itself, but also a prerequisite for users' ability to learn from the expert system's domain knowledge (in this study measured by prediction errors).

References:

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

Anzai, Y. & Simon, H.A. (1979). The theory of learning by doing. Psychological Review, 86, 2, 124-140.

Eason, K.D. (1991). Ergonomic perspectives on advances in human-computer interaction. Ergonomics, 34, 6, 721-741.

Lehner, P.E. & Kralj, M.M. (1988). Cognitive Aspects of the User Interface. In J.A. Hendler (ed.), Expert Systems: the User Interface. Norwood, NJ: Ablex.

Maass, S. (1983). Why Systems Transparency? In T.R.G. Green, S.J. Payne, & G. VanDeVeer (eds.), The Psychology of Computer Use. London: Academic Press.

Patel, V.L. & Groen, G.J. (1986). Knowledge-Based Solution Strategies in Medical Reasoning, Cognitive Science, 10, 1.

Roth, E.M., Bennett, K.B., & Woods, D.D. (1987). Human Interaction with an "Intelligent" Machine. International Journal of Man-Machine Studies, 27, 479-525.

Waern, Y. (1989). Cognitive Aspects of Computer-Supported Tasks. Chichester: Wiley.

Lack of usability in a patient administrative system as a function of events in the system implementation process

Carl Martin Allwood and Tomas Kalén

Department of Psychology
University of Göteborg, Göteborg, Sweden

ABSTRACT

In the present paper we take a usability perspective and report a case study from a project implementing a patient administrative system at a big hospital. The concept of usability is defined to include the components of user-acceptance, user-friendliness and user competence. User-friendliness includes program availability, compatibility with the users' mental properties, help-resources and possibilities for individualization. Our results indicate that many deficiencies in the project and the resulting system could be traced back to deficient contract formulations and to the lack of usability oriented developmental techniques in the project. Delays in the project had direct consequences for how the user training was planned and carried out. The user training delivered included very little instructional materials and the tasks given in the training had little relation to the trainees' future work tasks. On the basis of our results it is suggested that contract forms should be developed which include specification of usability goals for the system to be developed and possibly also specification of usability oriented developmental techniques to be used in the project.

1. Introduction

Many of the application programs developed in recent years are not very successful as aids for users in carrying out their work tasks. This is in spite of the fact that much research has been carried out in the area of human-computer interaction (HCI) and in spite of the fact that methods have been evolved which can be used when attempting to develop programs and computer systems which will serve the user efficiently (Helander, 1988).

In the present research we have attempted to analyze a system development process taking place in social context. In this way we hope to better understand how various events in a system development process, to a large extent uninformed by modern research in HCI, may affect the usability of the resulting program. Such results may be used in order to take measures which in real life may result in improved systems.

First we present an analysis of the usability concept. Goodwin (1987) points out that in order for a program to have *effective* functionality it should not only have the appropriate functionality (i.e. contain the relevant program functions) but it should also be usable. However, what is meant by usability?

Due to the complexity of the environment in which usability is to be implemented, usability can be seen as an interactional property in the sense that it is an effect of the interaction between the user, the task, and the program. This implies that usability relates to the system in a broad sense, including the computer, the user and different help resources. Usability can be analyzed into three parts: user-friendliness, user-acceptance and user-competence (Allwood, 1991).

One important aspect of user-friendliness is that the program should be compatible with the user's mental properties and support the user to utilize the program in the best way. For example, in a specific situation the program's demand on the user's attention should not exceed the capacity of his/her short-term memory. One other example: The information presented on the screen should (especially for novices) be as effective as possible in acting as cues which lead to the activation of relevant conceptions in the user's memory. For instance, menu items presenting fonts to chose from could be written in the font type the label stands for, similarly, the color symbolism used in the program should be compatible with the users' expectations.

Other aspects of user-friendliness are availability of the computer system when the user needs it, and the existence of a helpful help-facility and of other help-resources, the user manual being an example of an important help-resource. Moreover, the system, as far as it is relevant, should be individualized. For example, users should, if desired, be able to get more explanations from the system and they should be able to interact with the system in the language they prefer or at least in a language they know.

User-acceptance means that the user should feel motivated to interact with the system. This can often be ensured through measures taken in the system development process, for example by involvement of the users in the development process. By user-competence is meant that the user should have the appropriate competence to interact fully with the system. If this is not the case the system will be underutilized no matter how much functionality and other good properties the system might have. Previous research indicates that the functionality offered by the system is commonly underutilized by the user (e.g. Rosson, 1985).

In an early study Allwood (1984) interviewed eleven computer consultants and three other persons with similar work tasks about their experience of and view on usability factors in system development projects. The informants were chosen to be representative of companies of different sizes from different parts of Sweden. Delays were reported to be common in system development projects. There was great variation between the respondents with respect to the responsibility they took for usability aspects of the development projects. Furthermore, most of the consultants appeared to have very limited awareness of research results relating to the importance of considering the users' prior knowledge of various kinds when developing the program.

With respect to user training on the program, the informants were of the opinion that the ambitions for the training often were too limited in the sense that it did not aim for understanding of the programs but simply for operative ability. There was a feeling that the training often was carried out under too short a time, and that the effect of this was force-feeding.

Below we describe the events in the implementation process of a patient administrative system at a Radiology Clinic in a university hospital in Göteborg, Sweden. We focus on events with relevance for the program's ensuring usability and especially on events with relevance for the user training given.

2. Method

2.1. Data gathering methods

In order to understand the events in the development process different information gathering methods were used:

-Participant observation. One of the present authors was present at many of the planning meetings in the project and was also present in the sessions were the training on the system was carried out. After each meeting the researcher kept a diary of what had happened at the meeting. The diary was also used to write down information which was provided to the researcher in-between the meetings.
-Interviews. The key actors in the system implementation process were interviewed on their views of events in the development process, in some cases on two or more occasions.
- Protocols. Protocols from the project meetings were collected and studied.
-Questionnaires. In connection with the actual education, questionnaires were distributed to the trainees, before and after the education. Moreover, a brief questionnaire was distributed to the trainees after each session of the actual education.
-Performance data. After the end of the education we collected performance data from eight members of the staff who participated in the education. The users attempted to solve naturalistic tasks with help the computer and tasks involving finding information in the user manual for the program.

2.2. The system used

The system introduced in the development project, is a patient administrative information system especially designed to fit the needs of a radiology clinic. The system has been developed by a commercial software producer in close collaboration with radiology specialists. Some of the ideas behind the program were developed in connection with an off-spring project from a large research project mostly concerned with achieving proper functionality in computer programs for health care institutions. The main idea behind the system is to facilitate and coordinate the various work routines occurring when a patient is treated at the clinic, e.g. appointment registration, medical record handling, referral handling, drug prescription, care planning and evaluation.

The system is menu-driven and built up from several independent "modules". The modules can be connected to each other in different combinations in order to fit the activity patterns of a specific clinic.

The performance data collected and an informal walk-through of the system, with a usability perspective in mind, showed a number of usability deficiencies in the program. For example, in the Patient Identification module it was necessary to enter all fields with patient information even if one only wanted to change and save information in one of the fields. Thus, it was not possible to exit directly after having changed the information in a field. Instead one had to enter all the remaining fields before exiting the module. All in all, 18 key presses were needed to enter the relevant field, to save the changes made and to terminate the operation.

An other example of lack of usability was found in connection with the system's helpfacility. Performance data from eight users, each solving four realistic work tasks, showed that the system's helpfacility was not available on 93% of the occasions when the user needed help. The users only utilized the helpfacility on five occasions and not on any of these occasions were they helped to proceed further with the task (Allwood and Kalén, 1992).

3. Results

Here we focus on how different events in the development process affected the usability of the system and especially the planning and delivery of training on the program.

3.1. Choice of and decision about program

The program was chosen by the health care authority of Göteborg and the nearby county of Bohuslän in 1988. It had also been sold to other counties by the same computer consultant (hereafter: the consultant). The Radiology Clinic we studied was not heard at this stage. The understanding of the Clinic leaders when accepting the program was that the system was more or less ready for delivery. However, this turned out not to be the case at all. A contract between the health care authority and the consultant was signed in December 1988.

3.2. The contract

The contract between the consultant and the health care authority stated conditions for the adaptation and use of the program. The time for delivery of the program was stated to be March/April 1989. With respect to the provision of documentation, one copy of the system operating manual and of the user documentation was promised at the delivery of the program "in existing condition". In an Appendix to the contract the phrase "in requisite condition" was used. Provisions were made in the contract for the correction of "original errors in the software". In an Appendix it was stated that the consultant should correct errors and deficiencies in the program which were in conflict with what was stated in the documentation of the program. Vague statements were also made about the accommodation of the software to the needs of the Clinic.

The staff at the Radiology Clinic involved with the project critizised the contract with the consultant from the start. A complaint in 1988 was that the contract was incomplete, it was not clear what demands could be made on the consultant. In 1989 and 1990 the complaints were that it was not clear how much the consultant was committed to do with respect to the

adjustment of the program to the context of the Clinic, and furthermore, it was not clear what financial demands could be made on the consultant, for example in the case of delayed delivery.

3.3. Start and development of the project

Initial contacts between the Clinic and one person from the consultant took place in order to accommodate the program to the Clinic. This work was planned to involve tailor-fitting the program to the needs of the Clinic. However, this cooperation was not entirely successful and some members of the Clinic experienced the staff member from the consultant as a hindrance to the work. To some extent this problem may have been caused by different interpretations of the contract by the consultant and the Clinic. The difference of opinion seem to have been centered around the question of *how much* tailor-fitting the consultant was committed to do.

The Clinic early in the project revised a specification requirements from a hospital near Göteborg for most of the modules in the program in accordance with the needs of the Clinic. The revision was ready in June 1989.

The project organization soon split up into two groups: a project leaders' group and a project work group. The project leaders' group took care of much of the overreaching planning in the project and the work group did most of the work involving specification of details and the planning for the user training. The leaders' group did not interfere much with the work of the work group nor did they inform themselves very much about the specific details of the work carried out by the work group.

3.4. Economy

At certain periods there was a shortage of financial resources in the project but most of the time the economic situation in the project has not been seen as a cause of problems.

3.5. Delays

The system start had to be postponed many times. At first the system start was planned at the end of December 1989 or in the beginning of 1990. This could not be realized due to the fact that the program was not delivered until November 1989 and one part, the Record- (or Archive-) module, not delivered until April 90, however still without any help texts. Moreover, the delivery of the physical computer was also somewhat delayed. The "final" computer was installed in March 1990 and a corrected version of the program was delivered in September 1990. However, even in September 1990 there were bugs in the Record-function and subfunctions were missing. One module of the system was started in the beginning of January 1991.

3.6. Acceptance

Although the expectations and acceptance of the staff involved with the program was good from the start the many delays with system delivery, with the system start and the incompleteness of the system, including the manual, created feelings of frustration in the project group and among the staff.

3.7. Relation between the Clinic and the consultant

Initially the relation between the consultant and the Clinic was good. However, when the delay in the agreed program delivery began in the middle of 1989 the relation started to become strained.

Some interviewees from the Clinic staff complained that important representatives of the consultant made inappropriate analogies to the functioning of a much smaller hospital. However, in the opinion of the staff member from the consultant responsible for contacts with the clinic, the same problems occur in connection with both small and large hospitals. The only difference is, according to the consultant's representative, that these problems may occur more frequently in larger hospitals.

In general, there was a growing feeling in the project that the consultant had not lived up to the contract signed or other agreements made. Time limits were not kept and the consultant had not been responsive enough to the suggestions for changes of the program given by the Clinic. The explanation given by Clinic members is that the consultant had been too eager to sell, one had sold more programs than one has staff to support. However, according to the consultant's representative this is a question of disagreements of what actually had been sold and bought - a complete system, or a software product to be successively developed. Furthermore, the representative argued that the consultant only had responsibility to introduce changes into the program which were identical for all the hospitals, small as well as large, involved in the contract.

3.8. Development of the documentation

According to the contract the documentation to the program was to be delivered with the program. However, although in November 1990 the program had been delivered, only an incomplete version of the user's manual (henceforth: the manual) had been delivered.

The manual as it stands is completely a one man's product. The manual writer in an interview stated that to the extent that he had read the research literature on the writing of manuals he had not found anything usable for his manual writing. Instead he based his manual writing on his earlier experiences and on close contacts with more experienced colleagues in other companies. His developmental approach is to look for examples of good manuals from other Swedish computer companies and then to emulate their example.

The manual writer also believed that the documentation should be evaluated with respect to its usability and efficiency (this was also part of the agreement with the Göteborg health care

authority). However, it was not specified in the interview how the evaluation should be carried out. Finally, he considered the present documentation to be a "basic framework" to be further developed and improved in co-operation with the users. None of these ambitions seems to have been realized, and the updating of the manual to fit the new version of the system was just about to be completed in February 1992.

In order to get an impression of the usability of the manual we carried out a user test on the manual. In this test eight of the staff members who had received training on the system were given different tasks. For example, they were given three tasks which involved finding some specified information in the manual. The information requested was for task 1 to find out how to book a patient on the time nearest available on urgency level 2, for task 2 how to indicate that the patient was visiting the clinic for the first time, and for task 3 how to *insert* a word in a sentence. The subjects were also asked to state their opinion of the manual.

The results on the information finding tasks showed that all subjects could find the information requested in task 1, but that only two and no subject could complete task 2 and 3, respectively. According to the users' comments on the manual most users found it unstructured and they also commented on the difficulty of finding the requested information. However, the manual text was found to give extensive information on how to carry out tasks. The users also asked for a more elaborated table of contents. No user requested a subject index although the manual did not contain such an index (and still does not do so in May 1992).

3.9. Planning and delivery of the user training

In May 1989 a radiology nurse joined the project group, initially for a period of six months, as responsible for user training and education. Although the nurse had no teaching education, she had some experience of user training on a patient administrative information system similar to the present program. The nurse's previous experience of computer training appears to have been the most important criterium for her employment.

At the time when the teacher started to plan the training (August 1989), one immediate problem was that there were no adequate training materials. The only information about the program available to her was the specification requirements for the program. She several times requested adequate training materials or at least "hardcopies" from the program deliverer. As described above, a manual regarded as usable was not available and it took a long time, in spite of repeated requests, before the requested screen-dumps were acquired. Therefore, much of the teacher's time, allocated for the planning of the user education, was devoted to other work tasks.

The training we studied was delivered in January 1990. Forty staff members were given up to seven sessions on all available modules of the system, e.g. not the record module. The staff involved in the training were nurses, secretaries, receptionists, and Records Office personnel. One training session lasted two and a half hours and each training session was devoted to a specific module. A final session attempted to integrate the use of the different modules. The staff was trained in small groups (4 persons in each group).

All in all, each of the first three of the seven sessions were run ten times and each of the remaining four sessions on the average five times. The trainees came by administrative units (sections) which by itself may have created some homogeneity. No attempts were made to create homogeneous groups within sections.

We found the training given to the staff deficient in various ways. Very little paper training material was used in the sessions and the teacher did not encourage the trainees to take notes. A further drawback of the training was that it was not understanding oriented. For example, the system's character of a database was never explained to the trainees. Finally, the tasks given to the trainees to work with were unrealistic in relation to the work tasks they were to perform with the system.

Due to the strained work situation a trainee sometimes had to leave in the middle of a session. No substitutes were provided when the staff were participating in a training session. A further problem was that although the users were encouraged to practice on the program between the sessions and computers were specifically installed for this purpose, the users reported in our questionnaire that very little in-between session training actually occurred (on the average 13 minutes). Since the users showed a positive attitude towards the training in their questionnaire responses we suspect that the main reason for why so little in-between session training did occur was lack of time. No economic support was given, for example by providing means for substitutes, in order to ensure that the users had the possibility to practice between sessions.

Allwood (1990) suggested that computer training should include giving the users training in how they can seek help and solve problems that might arise when they have started to use the computer in their work. As the manual was not available when the training was carried out, no training could be given in how to use the manual. (However it did not seem to us that the teacher would have initiated such training even if the manual had been available.) Furthermore, minimal information was given about the program's helpfacility and no other information relevant in connection with seeking help in problem situations was given.

After the training we conducted a performance study, using four realistic work tasks and involving eight of the trainees. The trainees showed poor results (Allwood & Kalén 1992) which indicates that the training was not very successful.

4. Discussion

In general there was a lack of awareness of the importance of usability aspects of the system in the project. For example, no usability goals were ever formulated. Likewise, the contract between the vendor and the health care authority did not include any formulations with respect to demands on usability aspects. This suggests that there might be a general need for a better understanding of the importance of usability aspects in connection with system development projects among many project leaders and among administrators responsible for contract writing.

Although the advertisement for the program stated that the program has been developed in "close collaboration with the users" our observations on the usability of the program suggest that no user testing of the program had been carried out when the program was sold to the health care authority of Göteborg. Furthermore, no usability testing was carried out when the

system was developed and installed at the Clinic, nor did the users' manual go through any usability testing. Our performance study on the program and our evaluation observations of users as they attempted to use the manual clearly indicates that usability testing would have improved the program and the manual from a usability point of view.

Since usability directly relates to the work situation for the staff, it seems clear that usability issues are of relevance for the trade unions. However, although usability might be regarded to be a potentially "hot" issue for the trade unions, their representatives in the project group were on the whole been quite passive. An attempt was made by the unions' representatives to get a representative into the work group (a separate group from the project leaders' group). When this attempt failed the trade unions' representatives mostly paid attention to issues concerned with system specification requirements but did not engage themselves with usability issues.

On the whole, at least in Sweden, it appears that the trade unions are quite well aware of the importance of functionality issues but they do not appear to have the same understanding about the importance of usability. This is also evidenced in the system development and research projects in Scandinavia where the trade unions have been involved, for example the UTOPIA project (Floyd et al., 1987). The orientation in these projects have been mostly towards achieving proper functionality in the programs. It is also interesting to note that the patient administrative program that we studied in the present investigation was developed partly within a research project (DASIS) were the researchers took a similar approach to the one taken in the UTOPIA project.

It seems clear that the consultant and the Clinic took different perspectives on the developmental project. The interest of the Clinic was naturally to receive a system which was tailor fitted to the needs of the clinic and which had good usability. However, the consultant appeared to see the development project in a larger perspective, i.e., as one developmental project among many similar projects over time. The strategy of the consultant appeared to be to develop the program over the run of many implementation projects without having the ambition to make perfect each specific project. The same strategy seems to have been taken with respect to the manual. The speed at which it was developed did not coincide with the needs of the Clinic project. The statement of the consultant's representative to the effect that the consultant only planned to introduce changes into the program which were valid for all the hospitals which had bought the program supports this interpretation.

The vagueness of the contract as to the costs for delays in delivery for the consultant may have contributed to the lack of concern on the consultant's part with respect to agreed delivery time. As we have described above, delays in delivery was one source of difficulty in connection with the planning and delivery of efficient user training. However, it should also be pointed out that the delays in the project also facilitated a thorough planning of the ergonomical aspects of the physical computer work places for the staff.

Apart from the lack of instructional materials, other features of the training given also contributed to its lack of success as evidenced by the results in our performance study. Examples of such features are that the training was not understanding oriented and that it was not clearly oriented towards the future work tasks of the users.

It should also be noted that measures taken in the training did not optimally prepare for the users' future work situation in the sense that very little material was handed out which the users could refer to when using the computer in their work. Likewise, lack of practice in finding information from the manual, the lack of in-between session practice with the program also suggest that the measures taken in the system development process did not efficiently prepare the users for future work with the program. The usability deficiencies in the manual as such and in the helpfacility can also be assumed to make the users' future work more difficult than necessary.

A possible deficiency in the project was that the leaders' group did not keep themselves well informed about the specific details in the results from the work of the working group. However, the beneficial effects of such information would have depended on the knowledge and understanding available in the leaders' group about how best to implement system usability. Such understanding appeared to have been lacking in the project group to a large extent.

As has been noted above, the contract between the consultant and the health care authority lacked in many respects, for example with respect to clarity and especially with respect to usability aspects. We believe that this may be a common problem, at least in Sweden. The formulation of contracts between vendors/consultants and their employers appears to be one neglected factor in connection with realizing usability in computer systems.

An informal survey of some standard forms for contracts between consultants and employees in Sweden showed that these standard forms are not helpful at all when it comes to specifying usability goals which should be fulfilled by the project (Allwood, 1992). Likewise, the forms analyzed were not supportive as to making agreements on what forms of developmental techniques, for example usability oriented development methods, should be used in the project. It appears that contract forms need to be developed which include usability aspects.

Since the usability approach is not always well known by actors selling and buying implementation projects it could be helpful if standardized usability goals were developed for different settings and types of computer programs. Techniques for formulating measurable usability goals are well described by Whiteside, Bennett, and Holtzblatt (1988). Ready-written formulations of standardized usability goals could then be utilized by contract writers when preparing contracts.

References:

Allwood, C.M. (1984). En kartläggning av psykosociala aspekter i systemutvecklingsprojekt ur datakonsultens perspektiv. (An investigation of psychosocial aspects of computer system development projects from the viewpoint of the computer consultant). SYSLAB Report, No. 25. University of Stockholm, Stockholm.

Allwood, C.M. (1990). Computer usage by novices. In: Kent A. & Williams J.G. (eds.). Encyclopedia of micro-computers. Vol. 4, pp. 37-56. Marcel Dekker Inc., New York.

Allwood, C.M. (1991). Människa-datorinteraktion -Ett psykologiskt perspektiv. (Human-computer interaction -A psychological perspective). Studentlitteratur, Lund.

Allwood, C.M. (1992). Svårt att göra användbara program. (Difficult to make usable programs). UNIX, april, 42-43.

Allwood, C.M. and Kalén, T. (1992). Using a patient administrative system: A performance evaluation after end-user training. Computers in Human Behavior, (in press).

Floyd, C., Mehl, W-M., Reisin, F-M., Schmidt, G. and Wolf, G. (1987). SCANORAMA^{X}. Methoden, Koncepte, Realisierungbedingungen und Ergebnisse von Initiativen alternativer Softwarenentwicklung und -gestaltung in Skandinavien. (Methods, concepts, conditions for implementations and experiences from software development in Scandinavia). Ministerium für Arbeit, Gesundheit und Soziales des Landes Nordrhein-Westfalen.

Goodwin, N. (1987). Functionality and usability. Communications of the ACM, 30, 229-233.

Helander, M. (ed.). (1988). Handbook of human-computer interaction. Elsevier Science Publishers (North Holland), Amsterdam.

Rosson, M.B. (1985). The role of experience in editing. In: Shackel, B. (ed.). Human-Computer Interaction - INTERACT '84, pp. 45-50. Elsevier Science Publishers B.V. (North-Holland), Amsterdam.

Whiteside, J., Bennett, J. and Holtzblatt, K. (1988). Usability engineering: Our experience and evolution. In: Helander M. (ed.). Handbook of human-computer interaction. Elsevier Science Publishers B.V. (North-Holland), Amsterdam.

Acknowledgement

This research was supported by a grant from the Swedish Work Environment Fund (AMF).

THE ROLE OF PILOT STUDIES IN USER INTERFACE RESEARCH

L. Izsó #, M. Antalovits #, A.P.O.S. Vermeeren*

Department of Ergonomics and Psychology
Technical University of Budapest, Hungary

* Department of Product & Systems Ergonomics
Delft University of Technology, The Netherlands

ABSTRACT

In the literature on evaluation and experimentation nothing is said in general on why pilot studies are so useful. Most of the literature only states that they are useful and not why. We have been looking for literature stating what aspects of an evaluation are important to conduct pilot studies for, but could hardly find anything. Therefore we think it is useful to summarize why we think pilot studies are important in user interface evaluations in general and describe as an example how it helped us in our case study. We can expect a well designed and prepared pilot study to be useful in different fields and at different levels. These fields are: 1) designing the simulated setting, 2) selecting an appropriate set of data collection methods and techniques, 3) tailoring the data collection methods and techniques to the specific experimental situation. The results and experiences of pilot studies in the fields above can be either of general level or of situation (program) specific level.

1. Introduction

As a part of a research project[1] on investigating methods for evaluating human computer interfaces, we conducted an empirical evaluation of two software packages with similar functionality but different user interfaces. The research was devised mainly in order to provide us with information for user interface evaluation methods usability in general. In addition it yielded information on the quality of the specific user interfaces.

In this paper our experiences concerning the usefulness of the pilot studies will be discussed.

In the literature on softwer evaluation and experimentation nothing is said in general on why pilot studies are so useful. Most of the literature only states that they are useful and not why. Some of the fragmentary recommendations we did find on what pilot studies may be useful for, are for example:

- for estimations of variance of some measured variable;

- in cases of questionnaires and interviews:

- to reveal fallacies and unnoticed assumptions in the thinking of the interviewer;

- to filter out ambiguities in questions;

---

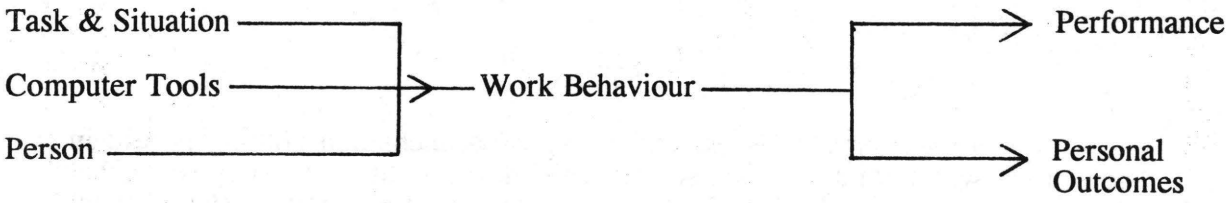
1 The project is part of a cooperative research effort of the Technical University of Budapest (Hungary), Delft University of Technology (The Netherlands) and Tilburg University (The Netherlands).

- to collect possible alternative answers to a question in the case of "closed" questions (like multiple choice questions);

- for training the experimenter in interacting with users;

- for training the staff in conforming to the needs of an evaluation plan.

We suggest to structure the statements on the usefulness of pilot studies on the basis of our working model:



The model reflects the logic and natural time order of a work process. On the left side of the model the subdimensions "Task & Situation" deals with aspects like: task domain, task structure, time constraints, accuracy demands, complexity of task etc. The subdimension "Computer Tools" stands for the programs that are to be evaluated. Programs can differ with respect to, for example: dialogue style, error handling, user support, etc. The subdimension "Person" deals with aspects like skills, personality traits, abilities, etc.

The subdimensions mentioned above can be considered as inputs for the "working behaviour" of a person using a program. This "working behaviour" then results in "performance" (quantity and quality of the work done etc.) as well as "personal outcomes" (satisfaction, change in skills, fatigue etc.).

The objective of the experiment we conducted was to have two software packages evaluated comprehensively. However, due to practical constraints, considerations of controllability of variables and the fact that one of the programs was not yet commercially available, it was not possible for us to evaluate them by means of a field study, so we had to conduct a laboratory study. This meant that a simulated setting of use for the packages had to be designed: the real "Task & Situation", real "Computer Tools" (programs) and real "Users" (the Real Setting of use of the program) had to be simulated and the simulated setting had to resemble the actual setting of the program. The assumption was then, that if that is done properly, the "work behaviour", "performance" and "personal outcomes" in the simulated situation should also resemble the "work behaviour", "performance" and "personal outcomes" in a real setting.

The resulting "work behaviour" in the simulated situation was registrated and analysed using Hacker's action theory. In addition the data on "performance" and "personal outcomes" were registrated and analysed in combination with the data from the "work behaviour" measurements.

Now just looking at this description, we can expect a pilot study to be useful:

1. In designing the simulated setting ("Task & Situation", "Computer Tools" and "Person");

2. In selecting an appropriate set of data collection methods and techniques (for collecting data on "Work Behaviour", "Performance" and "Personal Outcomes").

3. In tailoring the data collection methods and techniques to the specific experimental situation (the training and exact wording of the instructions for use of certain techniques

like CFF or thinking aloud, formulation of questions in an interview, the material used in the interview as triggers to get people talking, etc).

It can be expected that from a pilot study one can learn things both on a general and on situation specific level. On the general level we can expect findings that are generalisable to other similar evaluation exercises. These are findings that help in building up expertise on how to conduct user interface evaluations in general. The more expertise one has in conducting evaluations the less such findings can be expected. On the situation specific level we expect findings that are only valid for one particular evaluation. These findings relate to the specific program that is used, to the specific subjects or to the specific setting in which the evaluation is to be conducted. We think that no matter how experienced the evaluator is, a pilot study will always reveal a number of fallacies in the set-up of the experiment on the situation specific level.

Assuming the statements above are valid one can hypothesize that pilot studies are useful for both inexperienced and experienced evaluators; experienced evaluators will get findings on a situation specific level, while inexperienced evaluators will get findings on both a general and a situation specific level.

In our actual case what we really knowingly expected to get from the pilot studies were:

- testing and correcting techniques;
- training the experimenters in preparing sessions and interacting with users;
- being informed about the real time demands;
- interviewing subjects (some of them were experts) in detail about their experiences.

In the following we will describe the evaluation we conducted and analyse how the pilot study preceding it helped us in designing the final set-up. We will do this on the basis of the framework described above.

2. The case study

2.1. Background

The case study was a part of a methodology developing research work with the concrete subject of evaluating two user interface versions of an e-mail system recently developed by the Dutch PTT.

The whole case study was designed to be carried out in two steps: pilot study at Technical University of Budapest and the actual experiment at Delft University of Technology. We strived to utilize the results of the similar experiences - like that of JORGENSEN (1987) and LATRILLE (1990) - both in selecting methods and designing the details of the experimental sessions.

The pilot study was conducted in May - July 1990 and was related to the earlier version named Memostart (MS), while the actual experiment realized in February - July 1991 compared it with the new version named PCPakket (PC).

2.2. Experiment setup of the pilot study

2.2.1. Global description of a session

An experiment contained the following parts.

1. Oral introduction to the subject, (in Hungarian): (15 min.)

    - what is electronic mail;
    - what is the function of the MS program;
    - what will happen during the experiment: tasks, registration;
    - who are the observers: they are not the designers of MS;
    - tell the subjects that they are not evaluated but the MS program is.

After finishing introduction:

- measuring CFF
- filling in a subjective fatigue scale

2. Training tasks: first an oral explanation of how the MS program works (in Hungarian). Carrying out the training tasks two times. (min. 1/2 hour)

After finishing the tasks:

- measuring CFF
- filling in a subjective fatigue scale

3. Scenario tasks: (min. 1 hour)

After finishing the tasks:

- measuring CFF
- filling in a subjective fatigue scale

4. Interview (in Hungarian) about the opinion of the subjects, the errors which occurred and the experience of the subjects. (min. 1 hour)

5. Psychological tests. (1/2 hour)

2.2.2. Subjects

Six persons participated as subjects, three men and three women. Four persons were experienced in using a computer: two of them were computer experts and two persons were user interface experts.

The other two subjects had only a little experience with the use of computers, one was a secretary and the other a psychologist. Everybody except one used the computer daily and made also use of English computer programs. The secretary only used text-editing programs, the psychologist only data-entry programs and the others made also use of compilers and system editors.

Computers were only used at their work, not at home. They all used a IBM compatible computer, one user interface expert also used a Mackintosh. The subjects used personal computers for three to six years.

Only one user interface expert made use of electronic mail for correspondence.

2.2.3. Measurement of mental effort and strain

The level of arousal of the subjects was measured by the critical fusion frequency (CFF) meter and a subjective fatigue scale based on five items. The measurement took place at the beginning of the experiment, after the training tasks and after the performance of the scenario tasks. Analyzing the intraindividual differences between these "snapshots" on the subjects' arousal level and feeling of fatigue we can get a picture about the measure and dynamics of the mental effort and strain caused by the simaulated work. Because of CFF measurements the subjects were not allowed to drink coffee, tea or alcoholic drinks, to take medicines and to smoke cigarettes in between the experiment.

2.2.4. Training tasks

Before the training session started, the subjects got an explanation of the global structure and the main features of MS, with the help of a schedule and a written manual, and an itroduction of the thinking aloud technique as well as a short training to train this technique.

During the training the subjects got some written tasks to practise with the MS program. These were simple tasks to let the subject learn the useful functions and possibilities of MS. The messages they had to make were selected to be short and meaningless, not to divert the attention of the actions, and to limit the time needed for the training. The in built wordprocessor of MS was used. During performance of the tasks the subjects got feedback by asking them to look at the moved messages etc.

During the performance of the training tasks the subjects were allowed to ask questions and were asked to think aloud, to get used to it for the scenario tasks. But to know the difference between the thinking aloud and asking a question, the subjects had to make this distinction clear. He/she had to turn to the observer and ask a question. The observer only gave help when the subject asked for it, and in a short way.

An example of a training task is the following:

"Edit a new message with the content "222bbb" and save it. Give it the subject "s2" and the address "a2".
Put it in the worktray and export it to an external file named "mess2".
Go to MS DOS and look in directory C if this file "mess2" is present (type "dir/p)."

The errors were registrated manually with the help of a checklist. The Logscan program was used to registrate the screens for later evaluation. Video recordings were also made of the subjects and their thinking aloud. The aim of these measurements was to see in what phase of learning the subjects were, whether they had finished learning. The conditions of the training did not allow detailed registration and analysis.

2.2.5. Scenario tasks

At the beginning of the real experiment the subjects were given a written explanation about the context of the tasks.

The simulated frame of the experiment was the organization of an international conference. The tasks were related to this context, like writing letters, sending and manipulating them. The written tasks were given one by one on a piece of paper from a research member.

The text editing of the letters was not a research topic, so the errors in the use of the built in word processor were not important. A task was finished when the subject went back to MS DOS. This served a twofold purpose:

- there was no doubt about when the subject finished a task;

- it helped in finding task episodes when analysing the Logscan data.

The subjects were not allowed to ask questions, only to look at the manual and the schedule of MS. The subjects were asked to think aloud during the performance of the scenario tasks in their own native language (in Hungarian). In silent periods it was allowed to remind the subject of the fact that he/she had to think aloud; but it was not allowed to ask the subject to speak louder. No indications were given as to what to think aloud. Interruptions were not made during the performance of these tasks. After some practising it seemed that these interruptions would cause too many problems for the subjects.

In the design of the scenario tasks an iventorization of the possible tasks or actions was made first. On the basis of the experience the experimenters had with the MS program a score was given as a global indicator of the difficulty of the task.

The scenario tasks had been choosen from these possible tasks. Some easy and more difficult tasks were taken. An example of a scenario tasks is the following:

"Background information: You have to apply for financial aid to Mr. Wood who is the managing director of a foundation. This foundation formerly promised help.

Import an external file named "needs.con" to the worktray. Give it the subject "needs of conference" and no address.
Read it, the "total cost" is needed for the following letter.
Type the following letter.

Dear sir,

With reference to our former talks herewith I apply for... Hungarian Ft. aid to the conference organization costs.

Faithfully,

Dr. Z.R.

Save it and give it the subject "asking aid" and the address of Wood.
Put the message in the worktray and export it to an external file named "aid.con".
Import it again with the same subject and the same address. Send it to the outtray."

During and after the performance of the scenario tasks the following data were registrated:

1. By the help of Logscan all the screens were recorded with a time-interval of 1.10 seconds. Thus every action of the subjects which influenced the information on the screen was recorded.

Afterwards the following were analysed:

- The time to complete a task.

A task started when the subject got the written task and started reading. A task was finished when the subject went to MS DOS.

- The number and kind of errors of the outcomes of the tasks.

- The number and kind of errors during the performance of the tasks.

2. Video recordings of the subject and of his/her thinking aloud. The task number was given on a piece of paper at the side of the computer, the video recorder had an inner counter to use for searching some specific parts of the recordings.

3. The main actions, the problems of the subjects and the durations of these actions were registrated manually with the help of a checklist. This was needed for the interview afterwards and the error analysis. Also, the observers gave a mark (1 to 3) to every task to indicate the difficulty of this task for the subject.

4. Interview for the general opinion of the subject.
The subject was asked to give his/her general opinion about the system in an unstructured way.

5. Error analysis.

The manually recorded errors the subject made during the experiment were the starting points for the error analysis. The subject was also asked to give a mark (1 to 3) to indicate the difficulty of each task. These marks were compared with the marks of the observer, and the three most difficult tasks were evaluated. The mark of the observer had more meaning than the mark of the subject.

The subjects were asked to explain the problems they got with these tasks, with the help of the video recordings and the Logscan data as a reminder.

6. Asking for the experience of the subjects with computers, electronic mail with the help of a quastionnaire.
The general opinion and the error analysis were recorded by an audio casette recorder as a backup. During these interviews the observers had to write down the major comments of the subject.

7. Psychological testings

The aim of these psychological tests was to examine personality traits relevant to information processing. Three different tests were carried out:

- CPI (California Psychological Inventory)

- James-test (internal or external control)

- CFQ (Cognitive Failure Questionnaire)

3. What we had learned from the pilot study

So how did the pilot study help us? Let us state what concrete information it provided us with (using the categorizations described in the introduction):

3.1. In designing the simulated setting:

General level

- the fact that an observer sitting too close to a subject may affect the task-related behaviour of the subject; for example, in our case subjects started asking questions about the system when they were not supposed to;

- tasks containing phrases like "see if you can find a message that..., if there is no such message then..." make people very uncertain in phases where they are still learning to use a program; subjects keep wondering in such cases whether there is really no message or whether they do something wrong; phrases like these should be avoided in learning phases;

- realistic tasks should be formulated in job-related terms and not be too structured; in the pilot study the tasks were too structured and contained too much system-related terms; this helped subjects in performing the tasks; however, formulating tasks in system-related terms and structuring them to some extent may be useful to speed up the learning phase in an evaluation;

Situation or program specific level

- subjects needed more working space on their desk than was available in the pilot situation;

- the built-in text editor of one of the programs proved to be very irritating to subjects; as it was not intended that we evaluated the text editor we decided to keep the number of typing tasks as limited as possible;

- the first scenario tasks performed by the subjects still proved to be learning tasks, therefore the learning session had to be extended and modified; this was one of the reasons why we changed the set-up of the evaluation from one session per subject in the pilot study to two sessions per subject in the final study;

- one of the tasks proved to be too difficult and frustrating for subjects, and therefore had to be changed or moved to a later phase in the session;

- the layout of the task descriptions had to be improved as some subjects did not notice some of the information on it.

3.2. In selecting an appropriate set of data collection methods and techniques:

General level

- the CFF measurement proved to be sensitive enough to be used in software evaluations like these;
- the action theory proved to be applicable and useful in interpreting and analysing the data from various data collection methods;
- an error analysis interview as was conducted here, is very time-consuming; this was an other reason why we changed the original set-up of one session per subject to two sessions per subject;
- video registration of the screen proved to be necessary in order to make data analysis more efficient (as scrolling back was impossible with the Logscan program we used and as synchronizing the logged screens with the videorecordings of the subject sitting behind the screen proved to be impossible);

Situation or program specific level

- the pilot study showed us that automatically logging screens using the Logscan program does not provide enough information to trace back all keypresses in this case (as the screen and as highlightings on the screens were not registrated); the collection of logged screens therefore could not be considered as a complete protocol of an interaction and additional logging of the interaction by videotaping the screen proved to be necessary;
- thinking aloud proved to be an informative data collection technique for a program like this, especially in the learning sessions;

3.3 In tailoring the data collection methods and techniques to the experimental situation:

General level

- before conducting an evaluation, a strategy has to be decided upon when to interrupt a subject's task behaviour (e.g., in cases where subjects have severe difficulties or enter a part of the system they are not supposed to enter);
- for convenient analysis of thinking aloud protocols a rather sensitive microphone is needed;
- analyzing audio recording of an error analysis as conducted in this study proved to be difficult as the video recorded fragments that are discussed in it, as well as a video recorder itself, cause too much background noise;

Situation or program specific level

- the checklist we made for keeping track of the work behaviour proved to work well; the pilot study suggested how to use it optimally;
- in the pilot study it was sometimes not clear whether subjects were thinking aloud or asking questions to the experimenter; therefore we had to adapt the instructions

to the subjects; we had to instruct them to be very explicit in asking questions to the experimenter;

- the Logscan program proved to require too much computer memory space: this had consequences for the interval time with which screens were logged and made it necessary to make backups of the data after each subject.

3.4. Summary:

Summarizing we found that the pilot study we conducted indeed provided us with valuable information:

1. in designing the simulated setting the most important information it provided us with was both:

   - some general information on designing tasks as well as

   - specific information on designing the task sequences (how long does it take before subjects have learned enough about the system?) and the individual tasks (how difficult can they be?, are the task descriptions clear enough?);

2. in selecting an appropriate set of data collection methods and techniques it provided us with:

   - general information on the usefulness and sensitivity of the techniques that were new to us and with

   - specific information on the usefulness in this specific case of some of the techniques we already knew;

3. in tailoring the data collection methods and techniques to the experimental situation it provided us with:

   - some general practical information on how to use certain techniques that were relatively new to us and

   - specific information on the usability of the checklist we used, on how to instruct the subjects and on the limitations of the equipment and Logscan program we used.

A more detailed description of the pilot study can be found in WENDEL's and VERMEEREN's report (1990).

4. The actual experiment

In designing the actual experiment the importance of experiences gained during the pilot study can not be overestimated. Without these experiences, we are perfectly convinced, it would have been impossible to make an appropriate experiment design. Though the results of the actual experiment have been published elsewhere - e.g. IZSÓ(1991), IZSÓ and ZIJLSTRA (1992) - for the sake of completeness a brief summary is presented also here.

A "between subject" study was designed with 18 secretaries as subjects; half of them worked with MS and the another half with PC. Each subject had two sessions of about 3-4 hour duration, the first aiming at familiarisation and training (TRN session), the second one - one week later - for performing a set of scenario tasks (SNT session). A "keystroke

capture" program was used for registrating all the subjects' keystrokes during the sessions instead of the screen-registration by Logscan what was used before in the pilot study. The screen registration was realized by video technique together with the subjects' behaviour registration. To avoid the difficult problems of precise synchronization these two pictures were recorded on the same tape superposing one to the other in a way which did not disturbe the playback analysis afterwards. Beside the CFF measurements a special ECG equipment and software package (CARSPAN Cardiovascular Spectral Analysis) was used to estimate the objective mental effort. The other registrated parameters were the same as in the pilot study, but with more or less modified data collection. These modifications were also the results of the pilot study.

We identified the following four relevant levels of data processing and hypothesis testing:

- Keystroke level
- Action level (the individual actions were isolated by experimenters during video playback)
- Task level (a TRN session consisted of 14 tasks, an SNT session consisted of 11 tasks)
- Session level.

The main results are as follows:

- Keystroke level: as a consequence of the different user interface structures the total number and frequency distributions of the used keys differ (e.g. in MS more keystrokes, higher percentage of cursor keys, return keys, etc.)
- Action level: in MS there were fewer actions but these consisted of more keystrokes, action preparation time fractions were smaller, etc.
- Task level: tasks requiring longer total action preparatian time subjectively proved to be more difficult for the subjects, etc.
- Session level: as the performance measures basically did not differ between MS and PC, the main points of the comparison were strain measures. MS gives possibility to form longer, but fewer actions which do not require long action preparation, but the execution itself takes longer time and is a bit more stressing.

Over and above these level specific findings we made a lot of practical remarks for the program designers like

- there are some inconsequencies in MS: 1) in menus and word processing "Home" and "End" keys work, but in message lists do not; 2) in archive it is not possible to edit, but the cursor keys still work and can cause to disappear the message; 3) when deleting in archive it accepts invalid date, but afterwards does not delete while the user thinks it has been deleted; etc.
- there are some unnecessary limitations: in MS from archive nothing more is possible but to delete; in PC the "worktray" actually does not exists, the centre of the program is the "outtray", etc.

5. Conclusions

Based on our experiences with this case study we think a pilot study is always useful for preparing an empirical human- computer interface evaluation. In getting more and more experience in evaluation, evaluators may get better in selecting and using all kind of data collection techniques and in tailoring them to an experimental situation. However, for each new combination of program and subjects it is hardly possible to foresee (even for experienced evaluators) how difficult tasks can be, how fast subjects will learn to use a system and how subjects will interpret formulations in instructions and task descriptions. Therefore we believe that the main advantages of conducting pilot tests are in designing the task sequences and individual tasks for the subjects and in formulating subjects' instructions. In addition, when new equipment or software is used pilot studies will confront the evaluator with possible limitations for the use of it.

References:

IZSÓ, L. 1991, Report on a methodology developing user interface study. Internal report. Delft University of Technology, Faculty of Philosophy and Socio-Technical Sciences, Holland.

IZSÓ, L., ZIJLSTRA, F. 1992, Towards an Interface Evaluation Methodology. WWDU '92. Work with Display Units, Berlin, Germany.

JORGENSEN, A.H. 1987, The trouble with UNIX: Initial learning and experts' strategies. Human-Computer Interaction - INTERACT '87. Proceedings, Bullinger and Shackel (Editors), Elsevier Science Publishers B.V. (North-Holland), pp. 847-853.

LATRILLE, J. 1990, Design and Ergonomic Evaluation of the User Interface in an Electronic Information Communication System. 13th International Symposium HFT, Proceedings pp. 91-98. Torino, Italy.

WENDEL, I., VERMEEREN, A. 1990, A pilot study of the evaluation of Memostart. Internal report. Delft University of Technology, Faculty of Industrial Design Engineering. Holland.

part 5
USER INTERFACE DESIGN

Flexible Script Interface: An Intelligent Spatial Interaction Style

Ivan Burmistrov

Faculty of Psychology,
Moscow State University, Moscow, Russia

ABSTRACT

FSI, a Flexible Script Interface, is an intelligent, direct manipulation interaction style, that is based on the script approach to problem solving. FSI is implemented with an interactive script graph which represents a task structure and canalizes user's problem-solving behavior. The main concepts of FSI are: displaying a script graph directly on-screen and making it possible to interact with the system via such a graph; realization of user support with problem guide that is the intelligent basis of the interface using procedural expertise about standard scripts to guide a user in his activities; and on-fly visualization of script graph transformations as immediate feedback to user actions and data processing outcomes. The use of FSI is demonstrated by PersoPlan, a prototype decision support system.

1. Introduction

Behavior of most programs has a purposive nature. Software system's structure itself assumes that user would use it to carry out the definite task, that is to perform a number of actions that are expected to achieve desired goals. The problem is how to help user to arrive the goal with minimal efforts. This paper is concerned with an intelligent, direct manipulation interaction style for complex and/or novel tasks. The proposing Flexible Script Interface (FSI) is based on cognitive script approach to a representation of a task hierarchy. The main ideas of FSI are the following.

(A) Representation of a script in the form of graph directly on-screen. Thus, a user is provided with spatial cognitive map of the task world while solving a problem.

(B) Making it possible to interact with the system via such a graph. Thus, FSI gains advantages of direct manipulation.

(C) Dynamic visualization of script transformations as immediate feedback to user's actions and data processing outcomes.

(D) Giving a user on-line assistance with the task by the problem guide. The problem guide is an intelligent basis of the interface, which uses procedural expertise about standard scripts to support a user in realization of optimum performance during his problem-solving activity.

In this paper, we will discuss theoretical basis for FSI and try to predict those trade-offs that this interface could provide for a end-user, as well as describe implementation of this novel interface for PersoPlan, a prototype decision support system.

2. Theoretical basis for Flexible Script Interface

Generally, for the cognitive approach, human computer interaction is seen as presenting problems which have to be solved. Human problem solving is guided by a person's understanding of the domain of information which the problem under resolution represents. Such understanding, which can be conceptualized as a mental model or schema, organizes and directs a person's selection and usage of information in generating a solution. This paper describes an interaction style, based on the rubric of human problem solvers as builders and users of mental models. In this section, we will outline a number of theoretical assumptions which we have made on the basis of existing theoretical and empirical work in cognitive psychology and cognitive ergonomics.

2.1. Mental models

The term mental model has found increasing usage in human computer interaction, although this term can mean many different things to different researchers. For instance, researchers say subjects form a mental model of a task through imposition of a spatial model (Eberts and Eberts, 1989), or by their goals, plans, and strategies (Johnson et al., 1988).

Clark (1987) cites Hammond et al. (1982) that state: "To learn and to use a complex interactive system, users have to form a mental representation of the information needed for operating on it. The better structured this information is, the more usable the system is likely to be." Hammond et al. (1982) clearly distinguish between the *user model*, a body of knowledge existing in the user's memory, and the design interface image, a design concept acting both as a objective for the system development process and a reflection of the conceptualization that an ideal user might hold. A good design interface image can be judged on whether it helps a user to build up a good user model (Clark, 1987). Hammond et al. (1982) introduce a concept of "signpost," i. e. a task-to-action mapping rule. Clark (1987) infers that what makes for a good user model is having a set of non-confusable signposts when the user is lead through the design interface image's "maze," and stresses the critical role of good documentation and training material in formation of such signposts.

Script, or event schema, is a more elaborate concept of mental model. The concept of scripts has been used to explain text and story comprehension, as a way of understanding behavioral expectancies, and as a basis for designing artificial intelligence systems. Abelson (1981) defines a script as a hypothesized cognitive structure that when activated organizes comprehension of event-based situations. A script represents stereotyped knowledge structure that describes appropriate sequences of goal-directed actions in a particular context, such as "dining in a restaurant" or "attending a workshop." In Shank and Abelson (1977) approach, a plan is used in situations where people deal with events they have never encountered before. Goals are used as part of the plans to form expectations about likely events. Task performer will manipulate the goal structures when solving a problem and analyze the task by breaking it down into proper goals and subgoals.

Scripts consist from a number of scenes, which in turn are constituted by sequences of definite atomic operations. For instance, the restaurant script consists of "entering, ordering, eating, paying, and leaving." Each sequence of operations in scenes has a property of causal chain - every preceding action provides conditions for performing consequent actions.

In their Script-Based Information Processing model, Hershey et al. (1990) posit that scripts provide a framework that organizes the set of operations leading to the solution of a problem. They hypothesize that experts, through experience, develop problem-solving scripts, which are streamlined over time so that unimportant variables are dropped from the set of operations. The expert's first step then, is to select the proper script for a particular problem statement. Once this has been accomplished, proceeding to a solution is simply a matter of applying the algorithms called for by the script.

In our opinion, system developers can efficiently improve user interfaces, if they would provide a novice with expert's problem-solving scripts at early stages of user's communication with the system. Such scripts could help a user to form true signposts within the task world and facilitate the building of the good user model of the task. Usually novices obtain such knowledge from the manuals or on-line help messages. In both cases user has to interrupt the mainstream dialogue with the system in order to obtain some guidance. In FSI he needs not. One of the basic principles of FSI is to represent the task

world in form of a script and allow the user to interact with this script directly on-screen. We hypothesize that such representation facilitates the formation of the true user model of the task and efficiently reduces user's misconceptions and errors during problem solving.

2.2. Spatial reasoning

Some recent experiments indicate having the ability to make use of manipulation of spatial information may make human-computer interaction tasks easier to perform (Eberts and Eberts, 1989). Computer graphics provides an opportunity to present the user with an explicit representation of the task, resulting in facilitation of how the users think about the task, and it can allow the user to internalize the information which allows the user to solve novel problems, i. e. to form adequate mental models of the task (Eberts and Eberts, 1989).

One of the possible forms of script representation, besides pure textual descriptions of conditions and sequences of actions, is a network representation in form of the event graph. Each node of such a graph corresponds to a definite scene (procedural sequence of actions) and may have several entry and exit points. Arcs connecting nodes correspond to transitions from one scene to another.

FSI is based on such a network representation of activities in a man-machine dialogue. Task representation model in FSI is similar to composition graphs (Inman, 1987) and Jackson system development process structure diagrams (Sutcliffe, 1988). The FSI graph represents the hierarchical structure of actions which are performed within the problem-solving activity and the sequence in which they are performed. Sequence generally flows downward throughout the graph from initial node to end node. The main characteristic feature of the FSI graph is that it does not present simultaneously the complete collection of actions and transitions, which are permissible in the system. We believe such complete representation would be useful for a system developer, but not for a novice user who would be confused with a lot of irrelevant information. FSI is based on the principles of task context and cognitive economy in its representation of a task structure. In FSI a user is faced with a predefined "standard" script graph of the top level of task structure hierarchy. This provides a user with a "general view" of a task structure. It is important to emphasize that the script is a generic set of operations. The actual problem-solving process requires that parameters from the current task be plugged into the problem-solving script (Hershey et al., 1990). FSI reflects contextual dependent parameters in on-fly transformations of the generic script graph via mechanism of adding-deleting nodes and blocking-releasing transitions between nodes, and thus, invokes relevant information and removes information irrelevant to user's current focus. FSI operates flexibly to meet different user needs and computing conditions.

2.3. Direct manipulation

Shneiderman (1983) has described the defining characteristics of direct manipulation user interfaces: continuous representation of the object of interest; physical actions (movement and selection by mouse or button presses) instead of complex command syntax; rapid, incremental, reversible operations whose impact on the object of interest is apparent immediately on the display; and layered or spiral approach to learning that permits usage of with minimal knowledge. The essence of such a user interface is that the user seems to operate directly on objects rather than carrying on a dialogue about them (Jacob, 1989). Jacob's (1989) view of direct manipulation interfaces is based on clear models of how the interface should operate. Direct manipulation interfaces may be seen as a solution of problems that arise in the conventional interfaces. These problems typically stem from a failure to implement a task structure that is transparent to the user (Chignell, Hancock and Loewenthal, 1989). Direct manipulation user interfaces reduce a cognitive distance, the mental effort needed to translate from the input actions and output representations to the operations and objects in the problem domain. Another principal advantage of direct manipulation interfaces is that they decrease the demands on the user's short- and long-term memory, because the system continuously displays its internal data, rather than requiring the user to remember them (Jacob, 1989). Moreover, the user has less to keep in short-term memory about "where the system is" and fewer "places" it could be in. Jacob (1989) also states that ultimate success of a direct manipulation interface hinges on the choice of a good visual metaphor for representing the task world in terms of screen objects and input actions. It would be the difficult

problem in the domains that involve abstract objects, which do not have a direct graphical image, such as a time sequence, hierarchy, or conditional statements.

The visual metaphor chosen in FSI is the graph representation of the event schema (script), which shows clearly what the user can do to move from one node (scene) to another, and emphasizes the temporal and causal sequence of user and system actions in the dialogue. FSI permits the user to manipulate conditional statements (switches) and select actions and, as he does, it immediately redisplays the portions of the graph affected by these changes. We can adduce some plausible reasons to support our choice of the network visual metaphor.

The spatial representation of a script is "natural" in some sense. The natural language lexical collocations that are using to describe the area of problem-solving behavior, such as "to reach a goal, to pose a problem, to find a solution, a deep study, a decision space," etc., as well as extensive use of graphical notational tools in scientific representations of task structures, show that spatial metaphors are using widely to express features of problem solving. Thus, a script graph may provide user with an interface which could be akin to the hypothetical structure of his mental model of problem-solving process, and, therefore, could minimize the cognitive distance between the user's model of the task and appearance of the task that is implied by the interface.

Regarding the idea of that a spatial map of the task as well as the dynamic visualization of changes in this map during problem-solving process could make interface more transparent to a user, it could be pointed out that the mechanism of path blocking and releasing has proved its intelligibility in the area of labyrinth adventure video games, where preliminary performed actions such as "press a secret button" or "clear the room from the enemies" are used as prerequisites for successful passing from one compartment of a maze to another.

2.4. Intelligent interfaces

Jacob (1989) has stated that an intelligent user interface should be able to describe and reason about what its user knows and to maintain and use information about the user and his current state of attention and knowledge, the task being performed, and the tools available to perform it. Intelligent user interface "can control the presentation of output based on its model of what the user already knows and is seeking and remove information irrelevant to his current focus." Jacob (1989) infers that combination of intelligent user interface techniques with the direct manipulation promises for providing a highly effective form of man-machine communication.

An intelligent constituent of FSI, the problem guide, uses procedural expertise about standard scripts and their permissible transformations to direct user's focus, to determine current subgoals, and to correct possible user's misconceptions. It controls dynamics of the script graph and uses mechanism of path blocking and releasing to inform user about consequences of his actions and choices and to direct his goal seeking behavior. It also provides a user with on-line assistance by suggesting him the shortest way to the final goal through pointing out current subgoals.

3. Overview of Flexible Script Interface

3.1. Script graph

The syntax of FSI graphs is briefly and informally described as follows. There are three types of nodes and three types of arcs in FSI. A node may be: (1) an atomic one; these nodes correspond to terminal level of nodes' hierarchy, which provides bonds with application part of a program; (2) a complex scene, or subscript, which can be decomposed into atomic ones; these nodes represent medium level of aggregation in nodes' hierarchy, which is used to make interface representation more structural and better perceivable by the user; the decomposition of such a node is visually represented as exploding pop-up window placed over the current node; (3) a modifier, a terminal node which influence the developing the specified script.

The analogy with hierarchical menu system could make these concepts more clear. A script graph corresponds to a whole menu; atomic nodes correspond to terminal menu items, selection of which makes application to perform definite actions; subscripts are similar to submenus; and modifiers are similar to both multivalued setup items and those items of intellectualized menus that support item enabling-disabling mechanism (e. g., when "Paste" item will be accessible only after "Cut" operation being performed).

Arcs, or links, which allow to move from one node to another may be (1) permitted, (2) prohibited (blocked), and (3) recommended. The problem guide supervises user in his problem-solving activity and determines one of three above-mentioned possible states for each link. Prohibited links can change their status to permitted dependent on performing of defined prerequisite actions or changes in state of modifiers. User is free to choose transition through any permitted links, but usually only one of them would be marked as recommended by the problem guide.

3.2. Command input

Direct manipulation is the interaction style used in implementing the FSI driven system. The primary input device for manipulating the interface is the alphanumeric keyboard. The commands are a single keystroke of the cursor control keys, which are used to move through the script graph, and two predefined keys - "Enter" to confirm user's selections or decompose complex scene, and "Space" to confirm actions which are suggested by the problem guide. Mouse pointings of graph nodes and direct selection of nodes through function keys could be also available for the FSI dialogue to provide rapid access to desired nodes.

4. Additional potential advantages of FSI

The FSI makes it possible to implement various types of task sharing between user and machine, from cases when user appears to be imperatively telling the machine what to do, to the higher levels of automation when machine uses script graph as a dynamic flow diagram to inform user about what computer does currently. FSI can provide a framework for adaptive interfaces where machine regulates the workload of the human operator in a shared task.

The FSI has a reasonable symmetry of initiative in dialogue between human and computer. The problem guide provides on-line assistance for the novice but does not annoy and encumber the expert who can refuse to act upon advice and is free in choosing any permitted trajectory through a decision space, because this interface does not determine a fixed hierarchy of operations to satisfy goals. Thus, the interaction mode would be different for an expert and a novice, with smooth transition from novice to expert modes of operation.

Finally, the FSI can support wide choice of input devices such as the function keys, the cursor control buttons, the mouse, the lightpen, the touch screen, and the data tablet, which are useful in supporting spatial dialogues.

5. Domain selection

When designed FSI we have hypothesized that FSI would be maximally efficient at facilitating the complex and/or novel tasks. Chignell, Hancock and Loewenthal (1989) have suggested that the high level of the *task complexity* may be a preliminary criterion to warrant the development of intelligent interface. They have formulated the empirical notion of task complexity: "If users typically need a large amount of training and then continue to make errors after training, or generally fail to understand the model of the task presented in the interface, this indicates sufficient complexity to necessitate intelligent interface development." Although our discussion is mainly based on mental models approach to human-computer interaction, the artificial intelligence oriented model of problem solving, the problem space, will be adopted to conceptualize our understanding of task complexity.

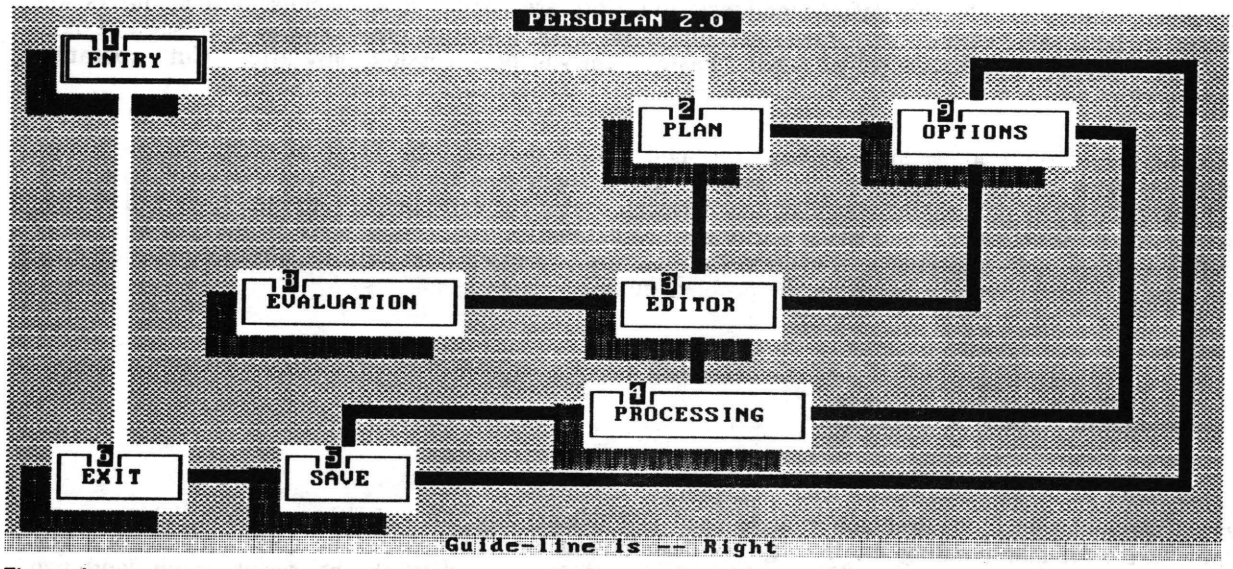


Figure 1.

The problem space model was formulated by Newell and Simon (1972). A problem space consists of a set of states, and a set of operators that are mappings from states to states. A problem is composed of a problem space together with an initial state and a set of goal states. Korf (1988) expresses the complexity of a problem in form of two parameters: the branching factor of the problem space, and the depth of solution of the problem. The edge branching factor is the average number of different operators which are applicable to a given state. The depth of a solution of a problem is the length of a shortest sequence of operators that map the initial state to a goal state.

In fact, this notation of task complexity is a topological one. Complex tasks have a lot of possible paths from initial state to goal states, and require a lot of operators to be performed to reach a goal. Clark (1987) also examines the topological aspect of task complexity and states that residual difficulties with human-computer interface after prolonged use may have more to do with the structure of the task itself, rather than the repertoire of mental information structures the user brings to the problem.

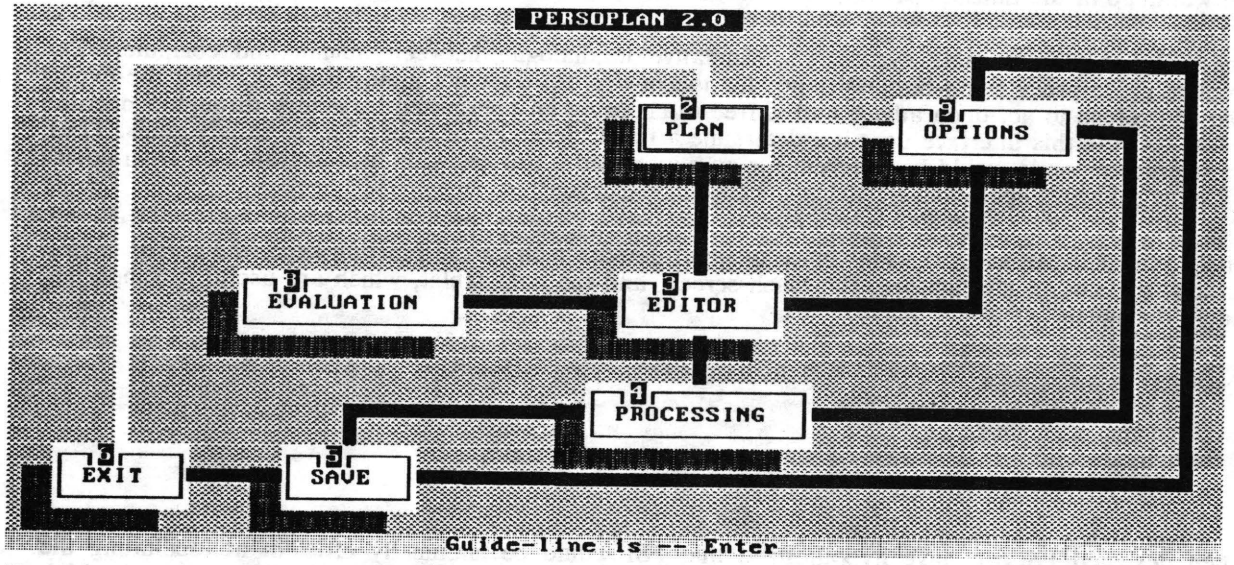


Figure 2.

In our opinion, complex tasks require high cognitive loads on user's working memory, because they involve many different operators and possible states, and require long sequences of actions to be performed to arrive a goal. Such tasks are perplexing for both novice and expert users.

Task novelty has a profound effect on the information search and selection strategies of problem solvers, because novel tasks keep the subject's knowledge from playing a role in generating a solution

and may force a subject into "weak" problem solving behavior due to their elimination of the role of task or domain specific knowledge in performance (Hershey et al., 1990).

Sometimes the novice users themselves are confused about their true needs and goals. The user does not fully understand his own true needs until he actually starts to use the system. We believe that FSI would give clear trade-off for a novice user as this interface represents general view of the system goals and operators at early stage of user's dialogue with the system.

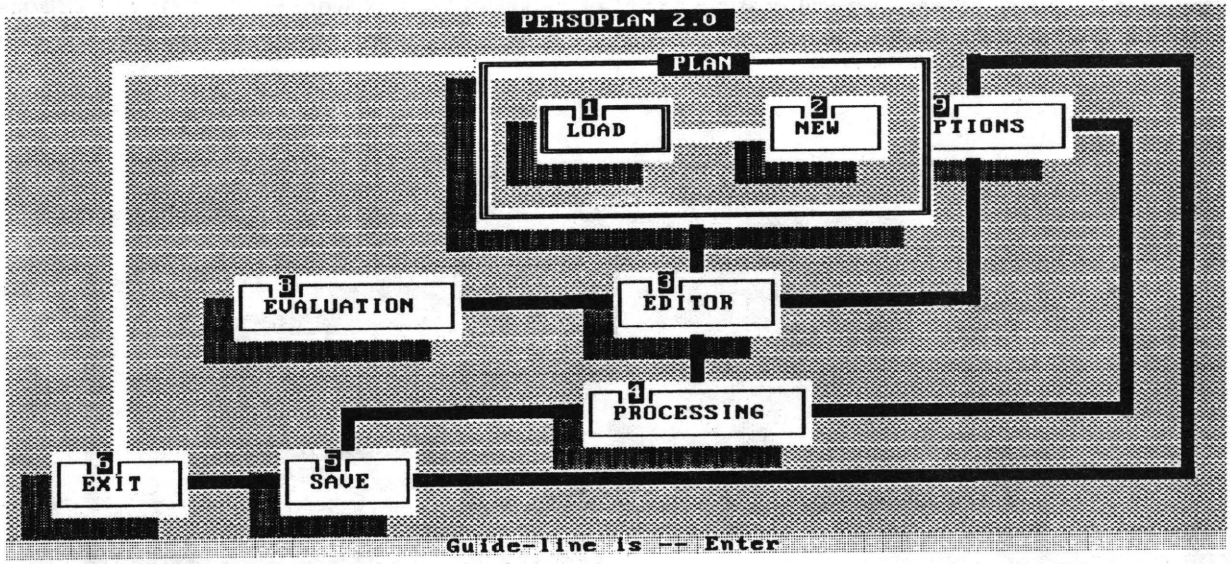


Figure 3.

Since the main goal of FSI development was to assess its efficiency in facilitation of solving complex and/or novel tasks, we sought a domain that is sufficiently complex and novel to encourage adding more intelligence to the interface. Decision support systems were selected as meeting these criteria. Decision support systems are informationally and operationally complex: knowledge of the domain specifies what variables are relevant, and how they must be manipulated in order to achieve a correct solution. Additionally, decision making tasks can be analyzed to derive a prescriptive sequence of "correct" steps by conducting a task analysis to identify the conceptual elements one needs to consider, as well as the procedures one must follow in order to reach a correct solution. Requirement of novelty is satisfied in for most decision support systems, because, as a rule, every such a system has been based on its own unique methodology, thus facing its possible user with a new task world and reducing user's expertise transfer from one decision support system to another.

6. Persoplan: an implementation of Flexible Script Interface

6.1. A terminological warning

We need at this point to forestall a possible confusion over terminology. As some other decision support systems, PersoPlan makes use of some terms, which occur also in the domain of cognitive psychology, with specific meanings. It is beyond the scope of this paper to discuss PersoPlan's methodology, consequently, in order to avoid possible concept interference, the reader is suggested to perceive such terms as "goal, plan, motive" just as hieroglyphs, when these terms are used in PersoPlan's context.

6.2. PersoPlan

Flexible Script Interface has been implemented for the PersoPlan (Personal Planning), a fuzzy decision support system based on psychological analysis of individual's motivation in decision making. This program is intended to assist user in multicriterial choices, helping him in comparison and evaluation of

multiple desirable and undesirable consequences of some decision. PersoPlan can be applied to various decision choice problems, e. g. selection of goods, vocational decision, etc. User can adapt the program to the domain of his interest, viz to create his individual plan by determining his personal goals, motives for selection these goals, and means for reaching these goals. The generic script includes the following phases: (1) creation of a new plan or loading of existing plan from disk; (2) judgment of subjective weights (e. g., goals would be ranked by "urgency" criterion); (3) judgment of correlations between means and goals; (4) judgment of correlations between goals and motives; and as a consequence (5) evaluation of priorities for goals. The typical script is not linear, however, because the program allows the user to choose between several possible modes of judgment such as score rating, pair comparisons, ranking and graphic scaling, to perform some actions in an arbitrary order, and to execute some sequences of actions iteratively.

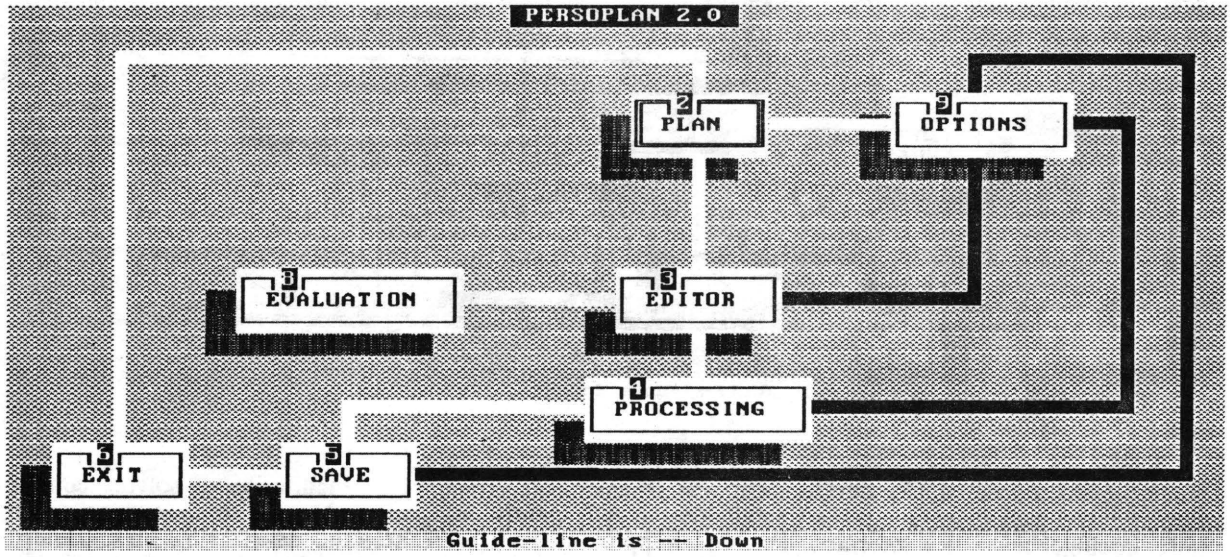


Figure 4.

The early 1.1 version of PersoPlan implemented menu-driven dialogue as well as question/answer interface and form filling. Context-sensitive help system provided on-line assistance to a user. Nevertheless, we have found that users typically needed a large amount of training and then continued to make errors after training, that indicated sufficient complexity of the problem to encourage intelligent interface development (Chignell, Hancock and Loewenthal, 1989). Thus, the conventional interaction styles did not provide user with sufficient support in operating with the program, so we have decided to improve interface radically when elaborated the new, 2.0 version of the PersoPlan based on script interaction style.

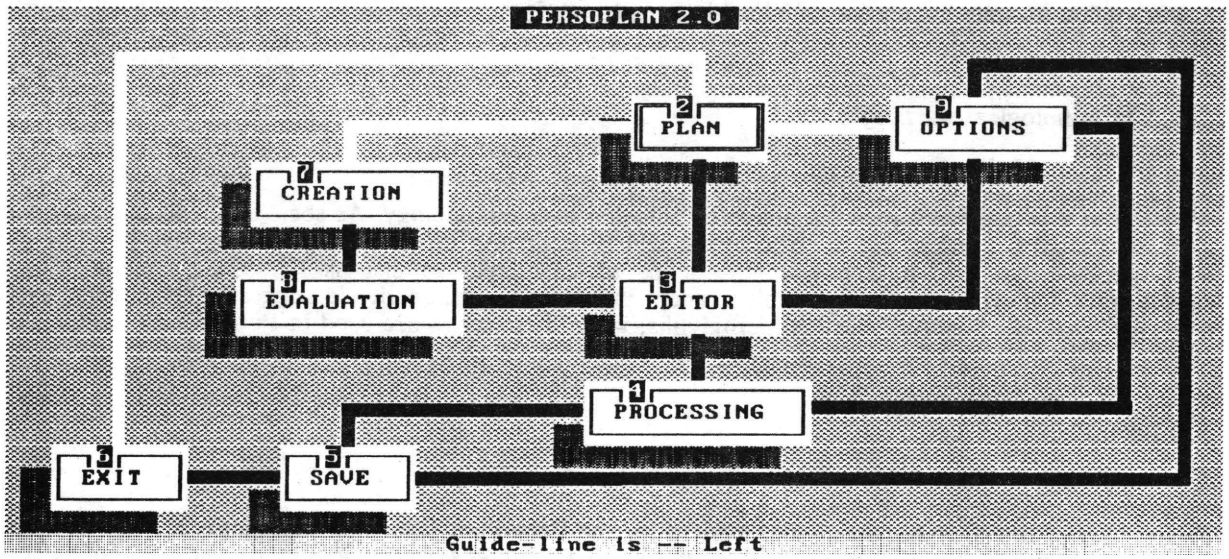


Figure 5.

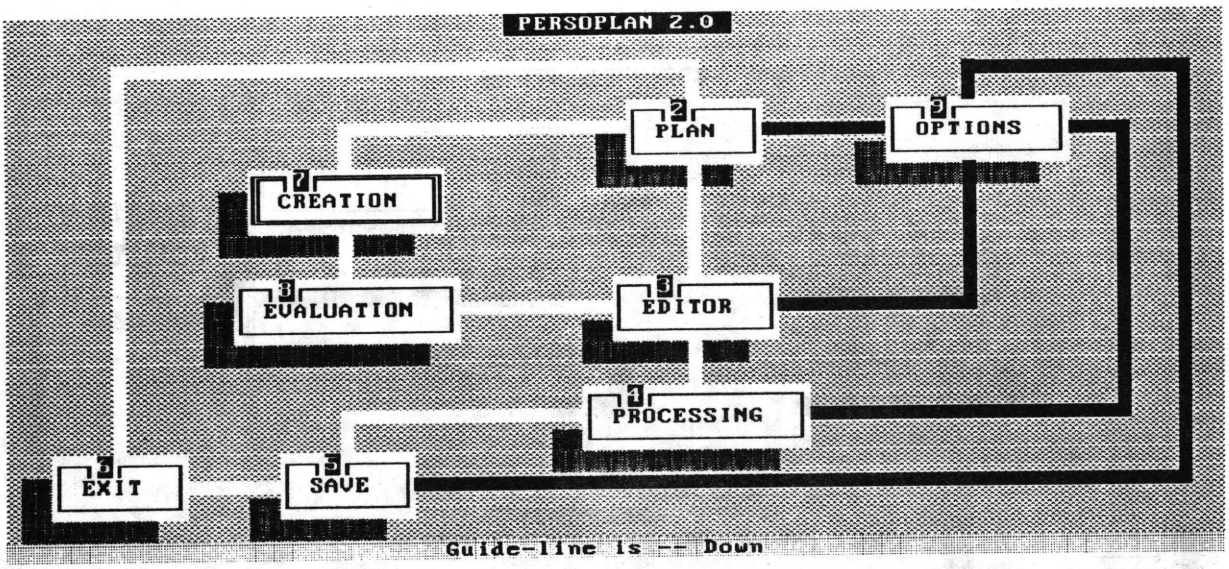


Figure 6.

Examples of screen dumps of PersoPlan's 2.0 interface are shown in figures 1 - 8. Permitted links are white, prohibited are black. An active (current) node has a double-bordered frame. Figure 1 shows the initial state of script graph, which represents definite standard default script, assuming that a user would operate with a previously created plan stored on a disk. There are two permitted transitions from this initial state: ENTRY --> EXIT, and ENTRY --> PLAN, and the last is the recommended one (see problem guide's message at the bottom of the screen). Figure 2 shows the state of script graph after the suggested transition has been performed. ENTRY node has been dynamically erased, because this node would not be needed in future. PLAN node becomes the active one. There are three possible transitions in this case: PLAN --> EXIT, PLAN --> OPTIONS, and to enter into PLAN. Problem guide recommends to enter into PLAN, a node that is a subscript. Figure 3 represents decomposition of subscript PLAN into atomic nodes LOAD and NEW. LOAD node becomes the current one, because default script assumes working with a preexisting plan stored on a disk. Figure 4 shows the state of the script graph after plan has been loaded from disk. This state make it possible to perform any transitions through arcs PLAN --> EDITOR, EDITOR --> EVALUATION, EDITOR --> PROCESSING, etc. Transition PLAN --> EDITOR is recommended by the problem guide, because this transition brings user near to the goal node, namely PROCESSING.

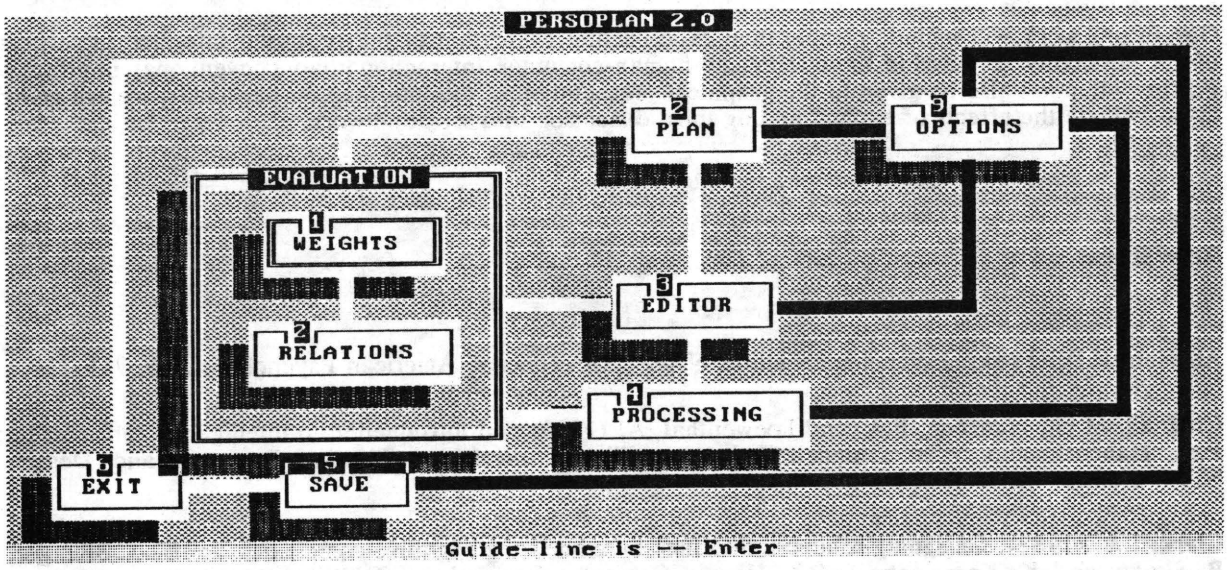


Figure 7.

Figure 5 presents the transformation of script graph for a case when user ignores recommendation of problem guide to select node LOAD (see Figure 3), and chooses node NEW. The new node CREATION is dynamically allocated on-screen, and transition PLAN --> CREATION becomes the recommended one. Note that transition PLAN --> EDITOR --> PROCESSING is prohibited in this case, because a plan must be created before its processing would be possible.

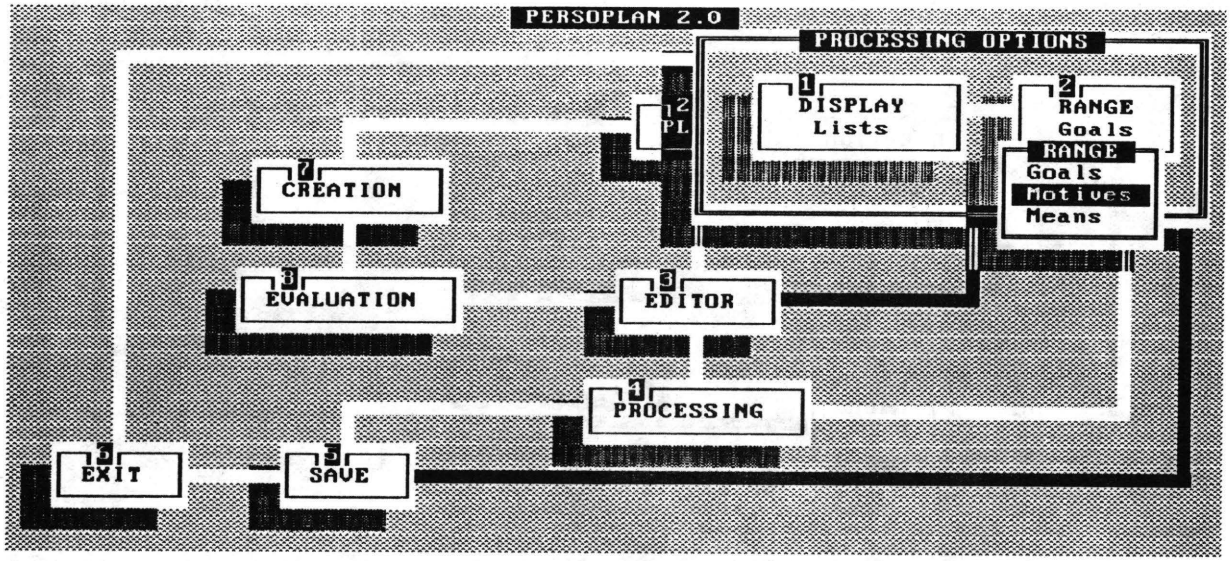


Figure 8.

Figure 6 shows the state of the graph after the new plan has been created. Figure 7 illustrates a decomposition of subscript EVALUATION into atomic nodes WEIGHTS and RELATIONS. Finally, Figure 8 demonstrates the selection of PROCESSING OPTIONS for the case when PROCESSING is the active node.

7. Conclusion and future work

This paper has presented an intelligent, direct manipulation interface for complex and novel systems. Unfortunately, our arguments in support of FSI currently rest on intuitive force, rather than empirical evidence. However, it is hoped that ideas presented in this paper would be interesting for system developers.

Our future work will include comparative experimental evaluation of the proposed interface in respect of its possibility to provide more efficient human-computer interaction than conventional interfaces. Further attention also be given to the graphical presentation of the interface, in particular, we are going to investigate the effects of explicit adding third dimension into script network.

References:

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

Chignell, M.H., Hancock, P.A. and Loewenthal, A. (1989). An introduction to intelligent interfaces. In: P.A. Hancock and M.H. Chignell (eds.). Intelligent Interfaces: Theory, Research and Design. North-Holland: Amsterdam, 1-26.

Clark, I.A. (1987). Designing a human interface by minimising cognitive complexity. In: H.-J. Billinger, B. Shakel and K. Kornwachs (eds.). Human-Computer Interaction - INTERACT'87. Proceedings of the Second IFIP Conference on Human-Computer Interaction. Stuttgart, 1-4 September 1987. North-Holland: Amsterdam, 101-108.

Eberts, R.E. and Eberts, C.G. (1989). Four approaches to human computer interaction. In: P.A. Hancock and M.H. Chignell (eds.). Intelligent Interfaces: Theory, Research and Design. North-Holland: Amsterdam, 69-127.

Hammond, N.V., Morton, J., MacLean, A., Barnard, P.J. and Long, J.B. (1982). Knowledge fragments and users' models of systems. IBM Hursley Human Factors Report No HF071. Hursley: Winchester.

Hershey, D.A., Walsh, D.A., Read, S.J. and Chulef, A.S. (1990). The effects of expertise on financial problem solving: Evidence for goal-directed, problem-solving scripts. Organizational Behavior and Human Decision Processes, 46, 77-101.

Inman, E. (1987). A syntax-directed graphics editor. In: H.-J. Billinger, B. Shakel and K. Kornwachs (eds.). Human-Computer Interaction - INTERACT'87. Proceedings of the Second IFIP Conference on Human-Computer Interaction. Stuttgart, 1-4 September 1987. North-Holland: Amsterdam, 119-124.

Jacob, R.J.K. (1989). Direct manipulation in the intelligent interface. In: P.A. Hancock and M.H. Chignell (eds.). Intelligent Interfaces: Theory, Research and Design. North-Holland: Amsterdam, 165-212.

Johnson, P., Johnson, H., Waddington, R. and Shouls, A. (1988). Task-related knowledge structures: Analysis, modelling and application. In: D.M. Jones and R. Winder (eds.). People and Computers IV. Proceedings of the Fourth Conference of the British Computer Society Human-Computer Interaction Specialist Group. Manchester, 5-9 September 1988. Cambridge University Press: Cambridge, 35-62.

Korf, R.E. (1988). Optimal path-finding algorithms. In: L. Kanal and V. Kumar (eds.). Search in Artificial Intelligence. Springer-Verlag: New York, 223-267.

Newell, A. and Simon, H.A. (1972). Human Problem Solving. Prentice-Hall: Englewood Cliffs, NJ.

Schank, R.C. and Abelson, R.P. (1977). Scripts, Plans, Goals and Understanding. Wiley: New York.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming languages, IEEE Computer, 16, 57-69.

Sutcliffe, A. (1988). Some experiences in integrating specification of human computer interaction within a structured system development method. In: D.M. Jones and R. Winder (eds.). People and Computers IV. Proceedings of the Fourth Conference of the British Computer Society Human-Computer Interaction Specialist Group. Manchester, 5-9 September 1988. Cambridge University Press: Cambridge, 145-160.

Visualisation and Graphical Interaction: Contrapuntal Support for Knowledge-Workers

S.M. O'Brien, E.A.Edmonds, L. Candy and N.P. Rousseau

LUTCHI Research Centre, Loughborough University of Technology, Loughborough, Leicestershire, LE11 3TU, U.K.

ABSTRACT

Visualisation of knowledge, and graphical interaction, are significant components in the development of knowledge-support systems for users with complex tasks, and visually-oriented expertise, who are non-programming personnel. For end-user knowledge manipulation systems, direct manipulation depends upon the development of appropriate knowledge representations which take the form of domain-familiar objects and concepts, and with which domain experts can interact in a harmonious manner. The notion of the *polyphonic* knowledge support system, in which modules offer contrapuntal support to the domain expert is introduced. Such systems are discussed with reference to knowledge acquisition, refinement, or manipulation, in complex domains or tasks which have specialist requirements. Further advances in such systems will depend on the robustness and extensibility of the communication medium which they provide for end-users. Work on such an interface to a speech science knowledge support system is described. In particular, the application of a specific intelligent interface architecture is reported.

1. Introduction

An early concentration of research effort on the provision of natural language interfaces for knowledge-based systems overshadowed the significant role which visualisation and diagrammatic representation has in the presentation, interrogation, explanation and manipulation of knowledge. A example of this natural language preference may be seen in the specialised interfaces and logic-based textual query languages to information systems such as on-line bibliographic information retrieval catalogues and scientific databases. Many information systems seem to have been designed with individual database structures which result in inflexible organising principles and retrieval mechanisms. Most interfaces do not mask the underlying query language, and interrogation requires complex query construction using elaborate command languages (Belkin et al., 1991). Alternative perspectives of the structure of databases in other than a linear manner is usually not supported. Some natural language interfaces can respond to a query by depicting information graphically, but they are unable to handle further questioning which arises from this, because related queries are implicitly about the syntax and semantics of the generated picture, and they lack an appropriate representation of these properties.

We address the particular needs of knowledge-workers whose expertise is visually oriented, and/or for whom diagrammatic reasoning is an integral part of the habitual problem-solving methodology. Obvious examples of the former are the skilled interpretation of signals such as foetal-distress monitoring traces, electro-cardiograms or electro-encephalograms. An interesting case of the latter is the use of diagrams to solve a meiosis problem by experienced and less-experienced geneticists, where there seems to be

noteworthy differences between the knowledge-dependent representational variability employed by the two groups (Hildebrand, 1989). Visually-based knowledge specifications in graphical form are being successfully used as an intermediate knowledge-level specification for knowledge-workers to express their knowledge which can be transformed into an appropriate formalism for the problem-solving engine in an organisational knowledge system (Davis & Bonnell, 1991).

The need for a *polyphonic* interface is identified and work on such interfaces in a speech science knowledge support system described. In particular, the application of an intelligent interface architecture is reported.

2. Visualisation

> *" Our story has a silent and immobile hero: the digital computer. There can be little doubt that computers have acted as the most forceful forceps in extracting fractals from the dark recesses of abstract mathematics and delivering their geometrical intricacies into broad daylight."*
> (Schroeder, 1989).

The interface design issue discussed above is being partially redressed by work in scientific visualisation and graphics simulation (see Schroeder, 1989 for a discussion of computer-supported work in number theory and fractals). Sophisticated graphics and imaging tools are credited with the acceleration of scientific progress and understanding. An example of this is the 'revolutionary' effect of accessible graphics software for PET and MRI techniques in brain research: the high-quality visualisation of domain-source data, and the multiple views and representations it supports, are reported to be generating new insights, uncovering new information, and confirming old hypotheses which could not be tested previously (Kiely, 1991). In interesting overviews of the Human Genome Initiative (Frenkel, 1991; Lander, Langridge & Saccocio, 1991) biology is reported to be undergoing a technology-driven paradigm shift. One of the challenges to computing technologists is to deliver tools and techniques to the domain-specialists, which will allow them to process experimentally gathered data, and to create and manipulate the databases of knowledge which will be an unparalleled research resource in "one of the monumental scientific adventures of history" (Frenkel, 1991, pg. 41).

Despite such demonstrations of the *computer-as-emancipatory-tool*, and as a medium of innovation, the use of computers in complex cognitive tasks and endeavours has a chequered history (Fox, 1990; O'Brien, 1992a for a discussion of the limited success of knowledge-based systems for automatic speech recognition; Rodd, 1990 for a similar discussion concerning machine-vision). Misunderstandings about the nature of expert knowledge, and disappointing implementations, have lead to a reluctance to use the term *expert systems* , as reported in Berry & Hart (1990). Conversely, so many software packages are promoted as expert systems, that it has become subject to semantic degradation, and (by association) diminished respect for the human competence which such systems purport to embody.

Visual programming is the use of visual representations (graphics, drawings or icons) in the process of programming. Program visualisation is the use of visual representations (graphics, images or animation sequences) to illustrate a program, data, structure of a complex system, or dynamic behaviour of a complex system. An example of program visualisation is GRAIL (Stepney, 1989) which allows the differences between sequential and parallel programs to be made visually explicit, and highlights parallel communication paths which are found in OCCAM which is based on concurrency and communication. GRAIL is restricted to the display of monitoring data amd it can not be used to debug or edit graphically. A visual language is the structured use of visual expressions to convey meaning. We are particularly concerned with visualisation and graphical interaction as techniques which allow knowledge-workers and computers to develop a working relationship through the medium of a pre-determined graphical domain-model (typically

derived from a task analysis). Visual programming for knowledge engineers is quite a complex matter (Murray and McDaid, 1992) but for knowledge-workers it may take the (relatively simple) form of flow-chart construction or may involve the manipulation of icons, or graphic markers, which represent domain-objects. Some implementations of such systems are described below.

3. Interface design for complex tasks

There is a re-emergence of interest in creativity in its many forms: artificial intelligence, scientific, artistic, problem-solving, personal, collaborative e.g. between people, or human and computer agents. Studies of creative processes are concerned with the amplification of skills and competence (Candy, Edmonds & O'Brien, 1992), not their levelling. We are particularly concerned with knowledge-workers who are engaged in the externalisation of domain knowledge, both to capture and refine it for the purposes of a knowledge acquisition exercise, and to further their understanding.

Creative potential, both scientific and artistic, may be stifled when knowledge-workers have to adapt their knowledge to an unfamiliar formalism, or communicate in an impoverished medium. An interesting case is the work of Harold Cohen, whose endeavours and experience are documented in Aaron's Code (McCorduck, 1991). Cohen's attunement and consciousness of his knowledge developed, and was externalised, through working with computers: he believes that his struggle to work with this medium has served to clarify his art in a precise and rigorous manner. In connection with this, McCorduck argues that externalisation transforms the nature of knowledge. In this view, validation depends on the formalisation of knowledge into computer languages. Thus, it is argued, the producers and users of knowledge must possess the means of expressing themselves in such languages. The concept of the computer , in fact, might have even more fundamental implications for knowledge (Edmonds, 1987). The underlying issues here are complex and concern the nature of knowledge itself. We may make a simple point, however. The means by which knowledge is represented are closely intertwined with the nature of the knowledge itself.

Computer-mediated knowledge support would seem advantageous and desirable. Yet, for knowledge-workers with complex tasks to perform, and valuable resources of knowledge to bring to that work, the motivation behind the design of interfaces to knowledge-support systems may seem unclear at best, obscurantist at worst. There is no implicit agenda in interface design which seeks to reduce knowledge-workers to data-filters (see figure 1). However, unless there is a rich medium of communication between the human and computer co-workers, augmented by a wide range of modalities, the latter will not fulfil their potential as a tool which enable users "to reason with an intelligent partner" (Thimbleby, 1990). There is concern about the use of decision support systems in safety critical situations (see, e.g., Boden, 1989), and the limited *accountability* of such systems (Fox & Krause, 1992).

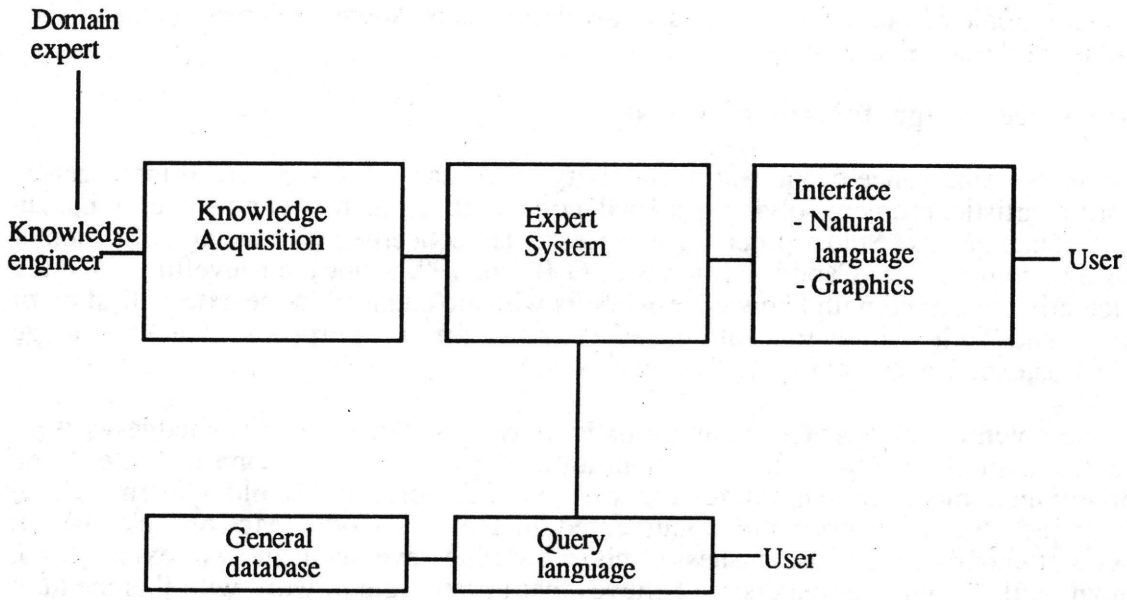


Figure 1. The typical relationship of an expert system (ES) to the environment in which it exists (adapted from Myers, 1986, pp. 101). On the left side of the figure, the domain expert is remote from the knowledge-base which is constructed by technical intermediaries (here, a knowledge engineer or system designer), human expertise is 'levelled' to a machine-tractable form. On right side, the ES interfaces to the user who inputs facts and suppositions of varying degrees into the system, using a formal language and receives answers, recommendations or diagnoses as output.

4 The polyphonic interface

Support for knowledge acquisition and manipulation in complex domains is technically and intellectually challenging. It is a special case of the problem of 'natural' programming in which the objects of concern are real world objects. Green (1990) says of object-oriented programming that it is supposed to simulate how we naturally conceive the world. However, in this case, we are supposed to conceive the world not as logical relationships but as objects and classes of objects communicating by sending messages. Yet, as he points out, the inter-relationships between real world objects are far more subtle and various than the inter-relationships that are available in any programming system. A key point is that we can easily hold a set of alternative views of the same object concurrently. This issue is central to the our concerns.

The potential of knowledge visualisation and graphical interaction techniques, and their implications, have yet to be assimilated satisfactorily into interface design for knowledge support systems, or extended to non-scientific disciplines. Laurel's interesting argument against the concept of the *interface* (Laurel, 1991) is followed in this paper by borrowing metaphor from music rather than drama. We introduce the notion of the *polyphonic interface*, in which the visualisation of domain knowledge, and the extension of graphical manipulation techniques to afford visual programming, can offer contrapuntal support to the activities of knowledge-workers.

Implementations which are currently being developed as support for end-users for knowledge acquisition, refinement, or manipulation, in complex domains or tasks, are briefly reviewed below. We then discuss the appropriate design goals for such knowledge support systems and point to a system architecture that is being employed to construct them.

5. Supporting the expression of complex knowledge

Episodic memory-based representation has been identified as a a significant model of innovative and creative design activity (McLaughlin & Gero, 1989). There may be a strong element of this in the behaviour of the study of participants in the meiosis problem discussed in the introduction (Hildebrand, 1989). The more-expert geneticists generated and used a greater number of chromosome replications (between two to eight) in solving the problem, than did the less-expert (two at most). The author of the study notes that the more expert geneticists flexibly altered their chromosome representations in accordance with the needs of the immediate working-segment of the problem, and produced fine-tuned diagrams which emphasised knowledge relevant to the task. Hildebrand argues that the knowledge thus high-lighted seemed to cue recognition of the salience of that knowledge to the task-in-hand. In marked contrast, the less-expert participants either over-specified their diagrams so that the salient attributes were difficult to discern among the irrelevant, or under-specified them, omitting relevant information.

The analysis of a user's goals when working with a knowledge support system reveals that the facility to construct, edit, or discard such visualisations and representations is essential to the knowledge-worker's practice and must be supported (Candy, O'Brien & Edmonds, 1992). Indeed the flexibility of these facilities can be critical if they are to enhance rather than constrain the exploratory processes that seem to be a central part of a knowledge-worker's activities. Thus, the knowledge-worker will often be making hypotheses about the data or rules and wanting to test these hypotheses through, for example, running customised rule sets, or alternative problem-solving methods, and then exploring the outcomes. The form these hypotheses take and the level of theory to which they refer can vary so that the information requirements of the worker in relation to establishing the validity of the hypotheses may be highly unpredictable.

It is improbable that there are consistent metaphors which are sufficiently robust to encompass a complex domain and anticipate its evolving needs, *while* being amenable to representation in the interface. However, our linguistic exposure to, yet tolerance of, inconsistent yet *coherent* metaphors and similes (Lakoff & Johnson, 1980), might suggest that there may be inconsistent yet coherent *visual* metaphors for representing functionality to the end-user, and providing interesting means for scrutinising and manipulating domain-specific source data. An interesting linguistic example of this is "to keep someone at arm's length", which is discussed in Lakoff (1987; pp. 447-449). Lakoff demonstrates that the linguistic expression does not specify any detail, but the hundreds of people he has asked have reported the same, or very similar, associated image. The physical image re-inforcing the psychological distance, has never been overtly represented to a native speaker, and yet it appears to be generally shared and recognisable.

Kocabas (1991) provides a useful comparison of several forms of knowledge representation in terms of their expressiveness and the types of reasoning or learning they facilitate. Several of these would be of use to knowledge-workers, because they support a variety of problem-solving strategies. It is desirable that these representations continue to afford their full power to the knowledge-worker, who can access them via a domain-tailored model. The representational problems of achieving this goal are not fully solved but deserve considerable attention.

6. End-user knowledge manipulation tools and systems

A strong motivation of knowledge support systems is to provide a interactive tools or environments with which domain expertise can be encapsulated and encoded or manipulated. This is independent of the computational representation required by the target internal system. It is interesting to note that these systems outlined below are explicitly concerned with complex domains, and the needs of knowledge-workers with complex tasks and goals.

Tranowski (1988) reports an integrated knowledge acquisition environment for domain experts which is used to capture and refine expertise in aerial scene analysis - a heterogeneous and complex domain. The domain expert interacts through the mediation of the interface manager which has pre-defined domain models containing the intended actions, goals and vocabulary of the application. The domain-models are expressed in a combination of graphics and text which are domain-familiar; the user is thereby shielded from the internal knowledge representation which is required by the system but can interact with, and manipulate, the information in an approximately natural manner. For the terrain analysis task in this application, the supported imagery (ie. the external representation) is of maps, terrain features and their attributes etc.; associated likelihood probabilities are indicated by the colour intensity of categories and objects. No further results are available of this work, and the degree of success is not commented upon, but it is a noteworthy knowledge acquisition environment.

CHAUS (Akutsu, Suzuki & Ohsuga, 1991) supports the knowledge for novel synthesis of chemical compounds in a representational form which is suitable for checking against the Chemical Abstract Service database (which contains more than 10M chemical compounds). The chemist users have an interface which supports knowledge representation in a graphical form, using familiar chemical structures and formulaic symbols for expressing bonds, valency etc., all of which can be modified. The external knowledge representation of chemical structures are treated as basic objects; the internal knowledge representation implementation in predicate logic of the data structure is hidden from the domain experts.

Several graphical knowledge-based model editors have been constructed using SimKit™ (Cypher & Stelzner, 1991). These are designed to function as graphical interfaces to knowledge-based systems, and the designers are keen to adapt to the types of domain knowledge which are present in KBS, and to exploit appropriate graphical metaphors to present that knowledge in the way that is most useful to the knowledge-workers. They use frames to represent objects; these objects belong to classes, and their properties are entered as slot values: slots can also be used to specify constraints between objects. The authors give an overview of several example interfaces to expert systems. Although some of the applications seem to involve only program visualisation, others can use heuristics to examine information, and generate "What if" facilities which engage the user and machine in an exchange which is aimed at achieving conflict resolution.

The Speech Knowledge Interface (SKI) (O'Brien, Candy, Edmonds & Foster, 1992) supports investigations in acoustic-phonetics which are concerned with deriving an enhanced model of speech production. SKI's pre-defined domain model motivates the graphical description of spectrograms (which are a domain-familiar visual representation of speech data). The graphical annotations are drawn on top of the spectrogram, and some resemble the phenomena which they represent: e.g., laryngeal vibration presents as visual stripes in appropriate sections of the speech signal, and the icon used to mark this occurrence is a set of stripes; the amplitude of a segment is denoted by the degree of shading of the intensity marker. Rules are created by typing domain-familiar terminology into the appropriate time-slots of a rule-window, and are concerned with the identification of speech sounds: they may be tested and refined interactively by the user. The annotations and rules are translated into a Prolog representation which is concealed from the user, who sees only the domain-familiar graphical objects and terminology which she has used. SKI supports research in a domain with interesting requirements for knowledge representation. SKI consists of modules which support spectrogram **annotation, rule editing, testing and inspection**; it incorporates:-

i) a representation of domain-specific source data that can be viewed continuously or selectively, and annotated using graphical means.

ii) a knowledge base, that can be manipulated, in the form of rules about specific visual information defined in the annotation; the rules can be created, modified and

validated without the need to directly use the underlying knowledge representation of production rules.

In the use of SKI the phonetician has been able to formalise some of her acoustic-phonetic knowledge and obtain good identification rates for her experimental task (O'Brien, 1992b). Progressive refinement and analysis of the spectral features which she has evaluated, in addition to her formalised knowledge, has contributed to the creation of an automated system which is designed to partition, graphically annotate, and classify the signal, in a manner similar to her practice (Edmonds, Pan & O'Brien, 1992).

SKI enables the investigator to flexibly explore areas of interest through the collection of carefully selected data samples that are represented as spectrograms, annotated and analysed using selected rule-bases. Hypotheses can be tested by simply creating rule-bases with new rules but what is clear is that more revolutionary discoveries rely on the investigator being able to develop new annotation types or new views of the data and to control the structure and execution of the rule-bases. Thus, the system should provide flexibility both in terms of the sequence of events in any one session and in terms of the very elements under investigation. These are often unpredictable and, over time, new versions of the system must be implemented to support emerging requirements. Designing a system that provides maximum flexibility in order to be useful over as long a period of future work as possible poses singular challenges to the development team. In this design and redesign process, the need for careful task analysis of the investigator's activities has proven vital in identifying both the basic procedures followed, the flexibility currently exploited and areas where greater flexibility is required. These issues are being addressed in the new system, SKSS, referred to below.

7 Design goals and a system architecture

In Lakoff's description for the experiential bases of metaphors "each metaphor has a source domain, a target domain, and a source-to-target mapping" (Lakoff, 1987), which is what an interface designer is interested in re-creating. In various degrees, this is what the designers in the systems described above have achieved.

Arnheim (1970) argues for the existence of a cognitive treasury of "visual thinking" which is sufficiently rich to accommodate the polyplastic nature of knowledge. Facilities for end-user programming and more imaginative use of visual metaphor may be a means of employing the implied opportunity into interface design.

Robust and flexible knowledge support systems will be an increasingly demanded provision for knowledge-workers (Fischer, 1992). Robustness, and the support for strategic and control knowledge (perhaps through graphical inferencing techniques) are issues which have not yet been addressed by the systems described above, but are the key design considerations (O'Brien, Candy, Edmonds, Foster & McDaid, 1992). Fox (1990) urges the need to separate decision procedures from the details of their implementation and applications; he lists the following as desirable attributes for decision support:

- The ability to reflect upon knowledge: to know whether or not there is knowledge available which is pertinent to a decision, whether it is partial, degraded, unreliable, and if so why.
- The ability to reason about inference techniques: decision techniques can have similar purposes but are appropriate under different conditions.
- The ability to know how and why a conclusion is derived - eg the need for explanation facilities to be couched in domain familiar descriptive objects, and transparent to the expert user.

- The ability to reflect on the decision process: to examine decisions - past, current and future; to determine if and when a decision should be attempted; to plan and control decision processes using appropriate information; to identify decision options etc.

In a study of a knowledge worker's practice, using SKI, seven design goals were identified that conform with and extend Fox's list (Candy, O'Brien and Edmonds, 1992). These goals are:-

- Enable an holistic view of visual data that includes multiple views.
- Enable simultaneous access to all forms of relevant source data.
- Enable a flexible and exploratory mode of interaction.
- The screen should be designed to take into account the impact of the method of interaction upon the user's perception of the visual data.
- The structure of the knowledge bases should be flexible in order to allow reflection upon them and upon their relative importance.
- Provide rapid feedback to the user about the rules employed and support evaluation with both positive and negative results.
- Enable knowledge bases to handle partial information in any aspect of the system.

Fulfilling these goals must be an aim for the development and extension of further knowledge support systems suitable for complex domains and tasks. A flexible architecture must be the foundation for any knowledge-support system with a non-determined or non-fixed number of knowledge modules to support and integrate (Tyler, et al, 1991). In the construction of SKSS, the successor to SKI (Aggarwal, et al, 1992), just such an architecture is being employed. It has been described by Edmonds et. al. (1992) and is illustrated in fig. 2. It provides a method whereby the user interface software acts as the integrating component in the system. The architecture provides a single consistent graphical user interface to the knowledge bases as well as any existing systems that the user needs access to.

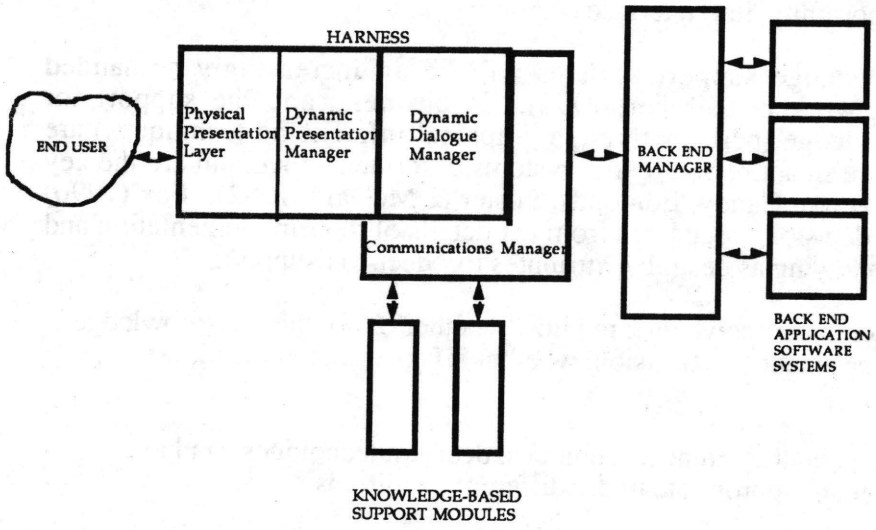


Figure 2. An architecture for the integration and re-use of existing software systems. In this architecture, a set of specifically constructed knowledge-based support modules and a set of existing back-end application software systems are integrated by a user interface management system known as

the Harness. The Back-end Manager has the specific role of translating the input/output conventions of the back-end application systems from and to the abstract interaction descriptions employed by the Harness. The various components of the Harness together deliver the physical interaction objects in a consistent graphical user interface.

8. Conclusions

Visualisation of knowledge, and graphical interaction, are significant components in the development of knowledge-support systems for users with complex tasks, and visually-oriented expertise, who are non-programming personnel. For end-user knowledge manipulation systems, direct manipulation depends upon the development of appropriate knowledge representations which take the form of domain-familiar objects and concepts, and with which domain experts can interact in a harmonious manner. The notion of the *polyphonic* knowledge support system, in which modules offer contrapuntal support to the domain expert was introduced. Further advances in such systems will depend on the robustness and extensibility of the communication medium which they provide for end-users. Work on such an interfaces to a speech science knowledge support system was described. In particular, the application of a specific intelligent interface architecture was reported.

Beyond this, it is a challenge to interface designers to provide a polyphonic interface in which all of the agents (human and computer) contribute to a coherent texture, and work together in accomplishing complex tasks and endeavours. The reported systems attempt this: perhaps only manner of student's counterpoint, but such systems will mature, and may achieve the flexibility and confidence of composer's counterpoint.

Acknowledgements:

The authors are grateful to Trevor Foster, Eddie McDaid and Anjli Aggarwal for their work on implementing SKI and SKSS. The work was partly funded by the UK's Tri-Council Initiative in Cognitive Science/HCI.

References:

Aggarwal, A., Foster, T.J. & Candy, L. (1992). Speech knowledge support system design. LUTCHI Research Report, K/10/4.1.

Akutsu, T., Suzuki, E. & Ohsuga, S., (1991). Logic-based approach to expert systems in chemistry. Knowledge-Based Systems Jnl, **4**, pp. 103-116.

Arnheim, R. (1970). Visual thinking. Faber and Faber, London.

Belkin, N.J., Marchetti, P.G., Albrecht, M., Fusco, L., Skogvold, S., Stokke, H. & Troina, G. (1991). User interfaces for information systems. Jnl. of Information Science, **17**, pp. 327-344.

Boden, M. (1989). Benefits and risks of knowledge-based systems. Report of Council for Science and Society. Oxford University Press, Oxford.

Candy, L., Edmonds, E.A. & O'Brien, S.M., (1992). Amplifying creativity: the role of end-user knowledge manipulation systems. In: Dartnell, T. (ed), Studies in Cognitive Systems. Kluwer, Dordrecht (to appear).

Candy, L., O'Brien, S.M. & Edmonds, E.A. (1992). End user manipulation of a knowledge-based system: a study of an expert's practice. International Jnl. of Man-Machine Studies (to appear).

Cypher, A. & Stelzner, M. (1991). Graphical knowledge-based model editors. In: Sullivan, J.W. & Tyler, S.W. (eds.), Intelligent User Interfaces. Chapter 17, pp. 403-420. ACM Press, Addison-Wesley, New York.

Davis, J.P. & Bonnell, R.D. (1991). Producing visually-based knowledge specifications for acquiring organizational knowledge. In: Motoda, H., Mizoguchi, R., Boose, J.H. & Gaines, B.R. (eds.), Knowledge Acquisition for Knowledge-Based Systems. pp. 105-121. IOS: Amsterdam.

Edmonds, E.A. (1987). Beyond computable numbers. Inaugural Lecture, Loughborough University.

Edmonds, E.A., Pan, L.Y. & O'Brien, S.M. (1992). Automatic feature extraction from spectrograms for acoustic-phonetic analysis. To appear: Proc. 11th IAPR International Conference on Pattern Recognition. IEEE Computer Society, Press. The Hague, The Netherlands. 30.8.-3.9.92

Fischer, G. (1992). Creativity enhancing design environments. In: Gero, J.S. & Maher, M.L., (eds.), Modelling Creativity and Knowledge-Based Creative Design. pp. 269-282. Erlbaum, Hillsdale: New Jersey.

Fox, J. (1990). Symbolic decision procedures for knowledge-based systems. In: Adeli, H. (ed.), Knowledge Engineering: Applications II. Chapter 2, pp. 26-55. McGraw-Hill, New York.

Fox, J. & Krause, P. (1992). Qualitative frameworks for decision support: lessons from medicine. The Knowledge Engineering Review, **7**, (1), pp. 19-33.

Frenkel, K.A. (1991). The human genome project and informatics. Communications of the ACM, 34, (11), pp. 40-51.

Green, T.G.R. (1990). The nature of programming. In The Psychology of Programming. Academic Press. pp. 21-44.

Hildebrand, A.C. (1989). Pictorial representations and understanding genetics: An expert/novice study of meiosis knowledge. PhD dissertation, University of California, Berkeley.

Kiely, T. (1991). Beholding the brain. Computer Graphics World, **14** (12), pp. 26-32.

Kocabas, S. (1991). A review of learning. The Knowledge Engineering Review, **6**, (3), pp. 195-222.

Lakoff, G. (1987). Women, fire, and dangerous things. University of Chicago Press, Chicago.

Lakoff, G. & Johnson, M. (1980). Metaphors we live by. University of Chicago Press, Chicago.

Lander, E.S., Langridge, R. & Saccocio, D.M. (1991). Mapping and interpreting biological information. Communications of the ACM, 34, (11), pp. 32-39.

Laurel, B. (1991). Computers as theatre. Addison-Wesley: Reading, M.A..

McCorduck, P. (1991). Aaron's Code: meta-art, artificial intelligence, and the work of Harold Cohen. W.H. Freeman & Company, New York.

McLaughlin, S. & Gero, J.S. (1989). Requirements of a reasoning system that supports creative and innovative design activity. Knowledge-Based Systems Jnl. **2**, (1), pp. 62-71.

Murray, B.S., Edmonds, E.A., Ghazikhanian, J. and Heggie S.P. (1992). The re-use and integration of existing software: a central role for the intelligent user interface. HCI '92, York, (to appear).

Murray, B.S. & McDaid, E. (1992). Visualising and representing knowledge for the end user: a review. International Jnl. of Man-Machine Studies (to appear).

Myers, W. (1986). Introduction to expert systems. IEEE Expert, **1**, pp. 100-109.

Berry, D.C. & Hart, A., (1990). Expert Systems in Perspective. In: Berry, D.C. & Hart, A. (eds.), Expert Systems: Human Issues. Chapter 1, pp. 7-9. Chapman & Hall, U.K..

O'Brien, S.M. (1992a). Knowledge-based systems in automatic speech recognition: a survey. International Jnl. of Man-Machine Studies (to appear).

O'Brien, S.M. (1992b). Spectral features of plosives in connected-speech signals. International Jnl. of Man-Machine Studies (to appear).

O'Brien, S.M., Candy, L., Edmonds, E.A. & Foster, T.J. (1992). Knowledge acquisition and refinement with end-user knowledge manipulation systems. In: Applications of Artificial Intelligence X: Knowledge-Based Systems Conference, SPIE Proceedings, Vol. 1707, pp. 25-36, Orlando, Florida, April.

O'Brien, S.M., Candy, L., Edmonds, E.A., Foster, T.J. & McDaid, E. (1992). End-user knowledge manipulation systems: the speech knowledge interface. In: Agrawal, J.P., Kumar, V. & Wallentine, V. (eds). Proc. ACM '92, Computer Science conference, pp. 359-366.

Rodd, M.G. (1990). Knowledge-based vision systems. In: Adeli, H. (ed.), Knowledge Engineering: Applications II. Chapter 8, pp. 245-276. McGraw-Hill, New York.

Schroeder, M. (1989). Self-similarity and fractals in science and art. Jnl. of the Audio. Eng. Soc., **37**, (10), pp. 795-808.

Stepney, S. (1989). Pictorial representation of parallel programs. In: Kilgour, A. & Earnshaw, R. (eds.), Graphics Tools for Software Engineers. pp. 46-59. CUP, Cambridge.

Thimbleby, H. (1990). User interface design. ACM Press, Addison-Wesley, New York.

Tranowski, D. (1988). A knowledge acquisition environment for scene analysis. International Jnl. of Man-Machine Studies, **29**, pp. 197-213.

Tyler, S.W., Schlossberg, J.L., Gargan, R.A., Cook, L.K. & Sullivan, J.W. (1991). An intelligent interface architecture for adaptive interaction. In: Sullivan, J.W. & Tyler, S.W. (eds.), Intelligent User Interfaces. Chapter 5, pp. 85-109. ACM Press, Addison-Wesley, New York.

Searching for Linguistic Arguments:
A Preliminary Assessment of the Usefulness of Linguistic Representations in an Editorial Support Environment*

Alice Dijkstra@ and Carla Huls#

@Unit of Experimental and Behavioural Sciences,
University of Leiden, Leiden, The Netherlands

Nijmegen Institute for Cognition and Information,
University of Nijmegen, Nijmegen, The Netherlands

ABSTRACT

We have started to investigate the possible usefulness of an editor equipped with linguistic functionality. For that purpose, we have studied logfiles of actual word processing sessions and investigated whether users might benefit from applying editor functions to linguistically defined elements. When applying functions to linguistic arguments, the effort to be exerted by the user to make certain revisions might be decreased and certain errors with respect to capitalization, punctuation and spacing could be prevented. Therefore we specifically searched the logfiles for possible function applications to content-based arguments. The logfiles indicate that not the whole range of linguistic elements constitute a useful set of arguments. Useful arguments seem to be: *entire-sentence*, *begin-sentence*, *end-sentence*, *NP* (Noun Phrase) and *PP* (Prepositional Phrase). Consequently, a sentence boundary analysis and a partial sentence analysis would seem to suffice for these purposes. Furthermore, quite a large number of errors with respect to capitalization, punctuation, and spacing were found, which might have been prevented provided the editor had access to a (partial) linguistic representation of the text, and the user had been able to apply suitable linguistic functionality.

1. Introduction

In the context of determining the functionality of an Editorial Support Environment (ESE), we evaluated a number of critiquing systems (Dijkstra, Huls, Claassen, Cremers and van Vliembergen, 1990). In Dijkstra et al. (1990) the following conclusions were drawn. Proper grammar checking of performance and competence errors requires syntactic analysis, and false alarms should be avoided, especially with respect to competence errors. Furthermore, we have developed a framework that structures word processing functions and the arguments they can be applied to (Huls, Dijkstra and Claassen, 1991). In the framework word processing functions do not constitute an arbitrary set of editorial support tools, but form a logically structured set of four types of functions which can be applied to three types of arguments.

---

* This research was carried out within the framework of the research programme *Human-Computer Communication Using Natural Language (MMC)*, subproject *Description of the Functionality of an Editorial Support Environment*. The MMC-programme is sponsored by SPIN Stimuleringsprojectteam Informaticaonderzoek, BSO, Digital Equipment B.V.

A first version of this paper has appeared as the internal SPIN-MMC report Dijkstra & Huls, 1992.

The function types are *text editing*, *text checking*, *layout editing* and *layout checking*. Three of them are directly related to the three levels of processing, viz. the generation process, the formulation process and the evaluation process, described by Hayes and Flower (1980) in their cognitive model of the writing process. To this model Huls et al. (1991) have added a fourth level, viz. the presentation process, to account for the composition of the actual layout of the text, which since the advent of powerful word processing systems formatting has become an integral part of the writing process.

The set of possible arguments consists of *content-based* text fragments — which are mainly based on text analysis — *media-based* text fragments and *user-based*, arbitrary text fragments.

		content-based						media-based			user-based
		text	section	paragraph	sentence	constituent	word	page	line	character	arbitrary
text	editing functions										
	checking functions										
layout	editing functions										
	checking functions										

Table I: The framework (taken from Huls et al. '91).

Apart from structuring and properly defining existing functions and arguments, the framework introduces — in order to minimize the distance between user goals and device operations — some new possible content-based arguments, i.e. a complete range of linguistic elements.

> *They [content-based text fragments based on linguistic analysis] facilitate the mapping from user goals to intentions, i.e. operations the user thinks he needs to perform in order to accomplish his goal. This idea is related to the concept of* distance *described by Hutchins, Hollan and Norman (1986) and the concept of* Bridging the Gulf *described by Norman (1986). It is also related to one of the premises of Moran's ETIT-model (Payne, 1985; Moran, 1983) and the Yoked State Space hypothesis described by Payne, Squibb and Howes (1990), viz. that the user has to map his goals onto device operations. (Huls and Dijkstra, in press)*

For the proper definition of linguistic arguments, such as *word*, *sentence*, *paragraph* etc, a syntactic analysis of the text is required. As a result of this linguistic analysis, new linguistic arguments will become available, i.e. syntactic constituents such as for instance *NP* and *PP*. The advantages of having access to a linguistic representation can be summarized as follows:

- The effort to be exerted by the user to make certain revisions might be decreased.
- Certain capitalization, punctuation, spacing, and agreement errors could be prevented.

The question now arises whether or not writers actually will use this type of functionality, i.e. the functions and the proposed set of text fragments as arguments. The fact that we have based the functionality on a combined cognitive and linguistic model of the writing process does not guarantee that writers will actually use it. Much depends on whether or not the users will benefit from using the functionality and whether or not the functionality is easy to use. That is, whether or not users can relate the functionality offered to the mental model they have of the task they want to accomplish. Or as Cookson states:

One challenge is to design tools or environments in which the writer feels his/her writing task is supported, and not obstructed, interfered with, or made more difficult. The second challenge is to competently design the human/computer interface to complement the tool/environment. (Cookson, 1989, p 21)

Our main interest lies in assessing the usefulness of a linguistic editor, that is an editor capable of computing and working with a linguistic representation of a text. For that purpose we have to investigate whether or not users are capable of and benefit from applying editor functions to linguistically defined elements. If they do, then we must determine which linguistic elements constitute a useful set of arguments. Furthermore, we have to determine whether the availability of a linguistic representation would indeed prevent the user from making certain specific mistakes.

Before engaging in a proper usability study involving subjects actually working with a research prototype of the linguistic editor, we decided to analyze actual word processing sessions, in order to find out which content-based text fragments users manipulate and to find the number of capitalization, punctuation, spacing, and agreement errors. This was accomplished by means of analyzing logfiles of these sessions, that is recordings of all keystrokes and intervals between them in such a way that the sessions can be re-enacted in slow motion afterwards. From these logfiles we determined and categorized the type of revisions and/or corrections users make when writing and revising their texts and also determined and categorized the types of errors that are made in the typing and revision phases. On the basis of these results we determine (linguistic) functionality that seems useful for these users. That is we suggest certain functionality that might decrease the amount of effort needed to perform the task of writing, revising and correcting a document, or functionality that might actually prevent users from making certain types of mistakes.

In the context of his PhD-research, van Waes (1988, 1991) studied the influence of a word processing system on revision behaviour. For this purpose, he collected a number of these so-called logfiles in order to study and analyze them. For our purposes some interesting conclusions can be drawn from van Waes' results. The results of his level of revision analysis are interesting, because they give some indication as to the frequency (of the type) of text fragment that is revised. These fragments are: word (53%), part of sentence (24%), sentence (5%) and paragraph (1%). The level 'part of sentence' was defined as being two or more consecutive words.

Van Waes' test does not address which revision acts and text fragments were actualy combined, nor does it concern the manner in which it was accomplished. His analysis does show that writing with a word processing system for whatever reason increases the number of form revisions but he does not differentiate between form, spelling and punctuation. However, the fact that more spelling, punctuation and format revisions are made indicates the usefulness of functionality that supports the writer in making those revisions or, if the revisions are in fact error corrections, functionality preventing the writer from making these errors.

Consequently, we decided to re-analyse van Waes' data in order to:

- further identify the level 'part of sentence',
- identify function-argument combinations.

At the same time we could check to see whether in the typing phase and/or the revision phases errors have been made, that could have been corrected or prevented if certain (linguistic) functionality would have been available:

- check for errors concerning capitalization, punctuation, spacing and agreement.

2. Method

The logfiles we have studied are taken from the ones collected by van Waes for the purpose of his PhD-thesis. He collected his experimental material by means of having subjects write letters, for example a letter to convince people of the usefulness of a certain measure taken by a bank. His subjects were second year students in Applied Economics following the course Business Communication. The letters were written either with pen and paper or with an IBM-PC with a standard screen. The letters written on the computer were all written with WordPerfect which all subjects had a reasonable command of. The keystrokes were recorded in a logfile and the subjects were also filmed. The pen & paper sessions were re-enacted in WordPerfect on the basis of the film and as such recorded in a logfile. Both the logfiles and the films were subsequently analyzed in order to study the subjects' behaviour as to pauzing and revising.

Following this experiment, he collected more logfiles from word processing sessions in which he not only compared pen & paper with computer, but varied the 'computer' subjects over those writing with a normal 80/25 screen and those writing with an A4 screen.

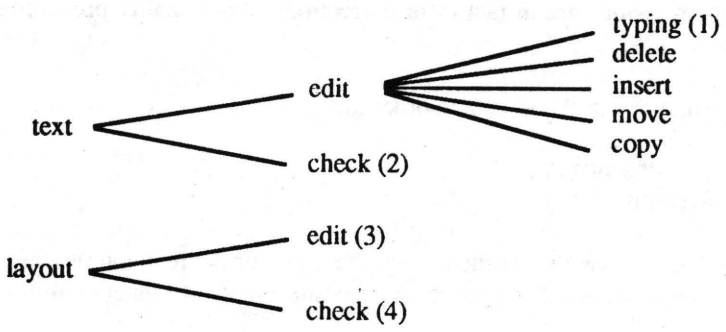
We reanalysed a total number of thirty two logfiles, with a mean length of 800 words. Sixteen from a first experiment, eight of them written on an IBM with a standard screen and eight written with pen & paper, and sixteen from a second experiment, eight of them written on an IBM standard screen and eight written on an ETAP A4 screen.[1]

The results were transferred to the following table:

	sentence					*constituent*				*text*	*ERRORS*			
	entire	*main*	*sub*	*begin*	*end*	*NP*	*PP*	*N/PP*	*ADV*		*caps*	*punc*	*space*	*agree*
typing														
delete														
insert														
move														
copy														
check														
layout														

Table III: Scoring Table

This table is related to the framework described in the introduction as follows:



[1] A more elaborate description of the method can be found in van Waes (1988, 1991).

Following the framework, four editor function types should have been distinguished: *text editing and checking functions*, and *layout editing and checking functions*. As you can see, we have not looked for any *layout checking* (4). The reason being, that this function type was not defined within the framework until after the reanalysis of the van Waes logfiles had already taken place. In the logfiles only one type of *layout editing* has been observed, i.e. *underlining* (3).

Within text editing five specific actions have been observed, i.e. *typing*, *deleting*, *inserting*, *moving* and *copying*. In fact, typing equals an insert action applied at the end of the text (1), and as such it is the only function in the test that is not applied to an instantiation (of a type) of sentence, constituent or word. The reason for scoring it anyhow is that we are interested in the type of errors with respect to form, i.e. *capitalization*, *punctuation*, *spacing* and *agreement*, that are caused while typing. The other four text edit functions represent the revision functions. We only scored text checking whenever a subject actually called the SPELL WORD function provided by WordPerfect.

Two main argument types are distinguished: *sentence*, and *constituent*. There is a subdivision within the argument type *sentence*: i.e. *entire*, *main*, *sub*, *begin* and *end*. Keep in mind that the arguments *begin-sentence* and *end-sentence* can carry a different meaning depending on the function that is applied to them. In combination with *delete* it means *delete from the cursor to the begin/end of the sentence*. In combination with *insert* it means *insert at the end/begin of the sentence*, and in combination with *move* or *copy* it means *move/copy something to the begin/end of the sentence*. Within the argument type *constituent*, the following four subtypes can be found: i.e. *NP* (noun phrase), *PP* (prepositional phrase), *N/PP* (noun phrase with a modifying prepositional phrase) and *ADV* (adverbial phrase).

It should be clear that none of these combinations of functions and arguments were actually used, but merely reflect the actions that were one way or another performed by the user. Almost all delete actions for example were accomplished by means of a series of delete-character actions. It must also be noted that the argument text was only scored if a function was applied to the text as a whole.

We can also see the number and type of errors which are made in the typing phase and the revision phase. The analysis shows whether the use of certain revision functions often resulted in making such mistakes. In the last four columns of the scoring form the errors with respect to capitalization, punctuation, spacing and agreement can be found that were made either when typing or after applying one of the revision functions. In fact, it is exactly these errors which might have been detected or prevented if appropriate linguistic functionality would have been used.

3. Results

The results of the reanalysis of the van Waes logfiles are summarized in Table II. In this table the frequency is represented that the function types *text edit*, *text check*, and *layout edit* could have been used in order to achieve a specific revision of a sentence-type or an constituent-type argument. Furthermore, the frequency is shown of certain error types that occur in the typing phase, or occur as an unwanted side effect of a revision.

In the next paragraphs we will zoom in on these results in order to show the frequency of the specific functions, the frequency of the sentence-type and constituent-type arguments, and the way in which they are combined. Finally, we will go into more detail with respect to certain specific errors that are made in the typing phase and the revision phase.

	sentence					*constituent*					*ERRORS*				
	entire	*main*	*sub*	*begin*	*end*	*NP*	*PP*	*N/PP*	*ADV*		*caps*	*punc*	*space*	*agree*	
text edit (typing)											5		4	9	**E**
											12		5	6	**I**
											9	2	5	4	**I**
											2		2	9	**P**
text edit (revising)	14	3	1	18	3	14	18	2	1		24	3	4	6	**E**
	7		1	31	1	6	9	1			12	2	2		**I**
	8	1		41	7	7	9				22	2	6		**I**
	9			22	7	3	7				8	1			**P**
text check										2					**E**
										2					**I**
															I
															P
layout edit						4									**E**
													2		**I**
															I
															P

Table II, summarizing the results
Conditions in the last column: E = ETAP/A4, I = IBM, P = Pen & Paper

The total number of function-argument combinations scored is 270. Almost all, 262, are text edit functions. The text check functions were only called four times and the layout edit functions were applied four times to a constituent-type argument, viz. a noun phrase.

		totals		
delete	**E**	55	157	195
	I	50		
	I	52		
	P	38	38	
insert	**E**	12	32	37
	I	5		
	I	15		
	P	5	5	
move	**E**	6	12	17
	I	0		
	I	6		
	P	5	5	
copy	**E**	1	2	2
	I	1		
	I	0		
	P	0	0	
under-line	**E**	4	4	4
	I	0		
	I	0		
	P	0	0	

Table III: Frequency of Edit Functions

The edit functions were intensively used within the revision phases. In table III you can find the results with respect to all the specific text edit and layout edit functions (that could have been) applied to the sentence-type and constituent-type arguments Note that the frequency of applying the spell checking function is absent here, the reason being that it was not applied to a sentence-type or constituent-type argument but only to the argument text.

Table III shows that the edit function most frequently used is the *delete* function, followed at a large distance by the *insert*, *move* and *copy* function, respectively. Conform the results found by van Waes, the total number of functions used when working with a computer (207 : 3 = 69) is larger than when working with pen & paper (52).

The results further indicate that only specific sentence-type or constituent-type arguments are regularly revised (see Table IV). These arguments are: *entire-sentence*, *begin-sentence*, *end-sentence*, *NP* and *PP*. The other sentence-type and constituent-type arguments, i.e. *main-sentence*, *subordinate-sentence*, *N/PP* and *ADV* were seldom revised. Evidently, no functions were in reality applied to an instantiation of any of these argument types. What did happen was that users sequentially applied a number of functions to a number of arguments which in the end resulted in a situation which could also have been

attained by applying a function to a sentence-type or constituent-type argument. For example, in order to erase a false start, a user might press the *delete-key* (i.e. *delete-character*) a number of times. On the other hand, he might – if available – also have applied the function *delete* to the argument *begin-sentence*.

	sentence					*constituent*				*totals*	
	entire	*main*	*sub*	*begin*	*end*	*NP*	*PP*	*N/PP*	*ADV*		
function	14	3	1	18	3	18	18	2	1	78	E
	7		1	31	1	6	9	1		56	I
	8	1		41	7	7	9			73	I
	9			22	7	3	7			48	P
total-C	29	4	2	90	11	31	36	3	1	207	C
total-P	9			22	7	3	7			48	P
total-CP	38	4	2	112	18	34	43	3	1	255	C+P

Table IV: Frequency of Sentence-Type and Constituent-Type Arguments (C = E + I)

Now we will look at the frequency of the combinations of text edit functions and sentence-type and constituent-type arguments. As you can see from the table (Table V), there are in general only two text edit functions that are regularly applied to sentence-type and constituent-type arguments, i.e. *delete* and *insert*. More specifically, the *delete* function is often applied to *entire-sentence*, *begin-sentence*, *end-sentence*, *NP* and *PP*, and the *insert* function is applied to *entire-sentence* and *begin-sentence*. Other sentence-type and constituent-type arguments appear to have been revised only very infrequently, if ever, by means of applying edit functions to sentence-type and constituent-type arguments.

	sentence					*constituent*				*ERRORS*				
	entire	*main*	*sub*	*begin*	*end*	*NP*	*PP*	*N/PP*	*ADV*	*caps*	*punc*	*space*	*agree*	
typing										5		4	9	E
										12		5	6	I
										9	2	5	4	I
										2		2	9	P
delete	9	2	1	13	3	12	12	2	1	7		3	5	E
	5		1	27	1	6	9	1		4	2	2		I
	5	1		27	6	7	6			6	2	2		I
	7			16	6	2	7			3				P
insert	2	1		5		2	2			17	3	1	1	E
	1			4						8				I
	1			14						14		2		I
				5						4				P
move	2						4							E
														I
	2				1		3			2		2		I
	2			1	1	1				1	1			P
copy	1													E
	1													I
														I
														P

Table V: Applying Functions to Arguments, and Resulting Errors

In table V the number of the specific errors can be found that were made in the typing phase, and those errors which are unwanted side effects from applying editor functions to sentence-type or constituent-type arguments. In particular we will discuss punctuation errors, capitalization errors, spacing errors, and agreement errors.

Capitalization errors occur most often at the beginning of a sentence. This class also includes words that should always be written with a capital because they are proper nouns or because a capital is required in the Dutch language. This is quite a large class of errors, 28 occurrences in the typing phase and 66 in the revision phase. In the typing phase they involve either forgotten capitals, or double capitals, because the shift-key was pressed to long. In the revision phase similar problems occur as those that will be described in the context of the punctuation errors. So almost always these errors are the result of something being deleted at the beginning of the sentence or something being moved or copied to that position. An extra difficulty in these cases is that whenever for example a word is moved from the middle to the beginning of the sentence, the user must capitalize that word and decapitalize the word following it, provided it is not a proper name or another word obligatorily starting with a capital.

Punctuation errors in this table do not involve wrongly placed commas but involve doubled punctuation marks, or forgotten periods or question marks etc. at the end of a sentence. This type of error sometimes already occurs in the typing phase (2 occurrences), but specifically happens in the revision phase (8 occurrences), often as a result of deleting the last part of the sentence, when the period is also inadvertently deleted. Punctuation errors are also caused whenever the writer inserts something at the end of a sentence or moves something to that position, and accidently does so after the period. Furthermore, when moving something from the end of the sentence to a position in the middle, the period may be accidently moved to that position as well.

Spacing errors are those errors which concern double spaces between words, forgotten spaces, or a space before a punctuation mark. This type of error quite frequently occurs when typing (16 occurrences). The main reason for making such spacing mistakes is that, contrary to writing with pen & paper, when writing on a computer spaces must be explicitly entered and/or deleted. When revising, writers also often have problems in maintaining correct spacing (12 occurrences). If a word must be deleted from a sentence then, depending on the position of this word in the sentence, the space before or following that word must be deleted as well.

In fact, most agreement errors are already made when typing and they often involve cases like: *25% van de gebruikers waren ..... (25% of the users were....)*, which is correct in English but requires a third person singular verb in Dutch. The number of agreement errors resulting from a revision was quite small.

4. Discussion and Future Research

We have investigated the possible usefulness of an editor equiped with linguistic functionality based on the framework described in Huls et al. (1991). A linguistic editor is a syntax-directed editor, that is an editor that 'knows' the syntactic rules of the language the author is writing in. Writers might benefit from such an editor because the availability of structured and properly defined functions and arguments, content-based, media-based and user-based, might not only decrease the effort to be exerted by the user to make certain revisions, but it might also prevent the user from making certain mistakes.

Not unexpectantly, almost all applied functions that we observed in the logfiles were text edit functions. This result can however not be considered to justify not having checking and/or layout functions available. The absence of subjects using these functions may be explained by the experimental setup of which the logfiles were the result.

The fact that only a few subjects used spell correction might for example be explained by the unwillingness of writers to use this type of function because of its inferior quality — too many false alarms or wrong error correction suggestions — or because writers are still not used to or unaware of having such functionality available. The absence of spell checking in the pen & paper conditions is evident because it was only scored whenever the function SPELL CHECK was actually called.

Obviously, much more (non-automatic) checking took place because a number of spelling, agreement, punctuation, capitalization and spacing errors were observed to be successfully corrected. It is however impossible to determine at what (linguistic) level this checking actually took place.

The task to be performed by the subjects was mainly directed at the second and third phase of the writing process, i.e. the formulation and evaluation process, and less directed at the fourth level, the presentation process. The subjects were not specifically requested to deliver a camera-ready copy of their text but only a draft proposal. Furthermore, the subjects were not using WordPerfect on a regular basis in order to obtain a perfect text layout but merely as an advanced typewriter that allows them to store and revise written texts. Other types of subjects and another task might therefore have produced different results as to using layout functions.

In our research we specifically searched the logfiles for possible function applications to content-based arguments, in order to determine which set of linguistic arguments might be considered useful. The logfiles indicate that not the whole range of linguistic elements constitute a useful set of arguments, only certain sentence-type and constituent-type elements appear to be subject to revision acts. The sentence-type elements involved are *entire-sentence*, *begin-sentence*, and *end-sentence*. The constituent-type elements involved are *NP* and *PP*. Consequently, a sentence boundary analysis and a partial sentence analysis would seem to suffice for these purposes. This is a very important conclusion, considering the fact that a linguistic editor requires an online syntactic analysis, and the fact that a complete syntactic analysis is computationally very expensive.

The test also indicates that quite a large number of errors with respect to capitalization, punctuation, and spacing were made in both the typing and the revision phase. They might have been prevented provided the editor had access to a (partial) linguistic representation of the text instead of a graphemic representation, and the user had been able to apply suitable functions to specific fragments of this representation. Then the user would not have to bother about these aspects of the text because the editor would be handling them automatically. We also found a number of agreement errors that could have been prevented, but only if a complete linguistic representation of the text had been available.

The logfiles show that users of word processing systems could benefit from the availibility of an online partial linguistic representation. They demonstrate that linguistic text fragments, such as sentences, NPs and PPs, are being revised regularly. If users could apply functions directly to these linguistic text fragments, then the effort to be exerted by the user would diminish and so would the number of the types of revision errors described before (Huls et al., 1991; Payne, Squibb and Howes, 1990; Hutchins, Hollan and Norman, 1986; Moran, 1983). We have started to investigate how to obtain a partial linguistic representation of a text which properly identifies these linguistic text fragments (see Terhorst, 1992).

The question however still remains whether users will actually use such functionality. For example, if a user wants to delete the subject of a sentence, realized by a noun phrase consisting of three words, he can opt for several methods. Amongst others he can hit the backspace key a (large) number of times, or apply the delete function three times to the content-based argument word, or apply the delete function to the content-based argument, i.e. the constituent, as a whole.

When opting for the first method, the user can immediately after determining the starting point of his delete action start deleting and — while hitting the backspace as many times as there are characters and spaces in the to be deleted phrase — decide where he should stop deleting. When sequentially applying the delete function to a word, less actions are required: select word each time followed by hitting the backspace as many times as there are words in the phrase to be deleted. When choosing the last option, the least number of actions are required: the to be deleted phrase can be selected as a whole and it can be deleted by hitting the backspace.

Obviously, users can also opt for selecting a (series of) user-based argument(s) by means of block selection, where the begin position and end position of the phrase to be deleted have to be exactly marked by the user. Subsequently the user can delete the selection as a whole by means of hitting the backspace. The user has to remember to include one of the surrounding spaces in the block selection in order not to leave a superfluous space behind. Especially when using a proportional font such a mistake cannot easily be spotted.

When choosing the *word* method and especially when choosing the *constituent* method, users need to carefully plan the action and evaluate its consequences before applying it. This contrasts with the first method where users can immediately start deleting. In her experiment about learning to use a word processor, Waern (1991) found that the general hypothesis of her model, i.e. suggesting that people think first and act later, proved to be supported by 85% of her subjects. Her results suggest that users may indeed apply functions to larger units than characters.

An important factor in deciding whether or not to apply functions to certain specific linguistic arguments is the user interface. It has to make clear to the user which linguistic arguments are available and it has to make these arguments easily accessible. A usability study of a prototype linguistic editor with an appropriate interface which allows subjects to manipulate context-based text fragments is planned in order to check whether or not users of a text processing system will choose to apply functions to linguistic text fragments such as noun phrases and prepositional phrases.

References

Cookson, S. (1989). Designing Computational Writing Tools. In: Computers and Writing. Williams, N. and P. Holt (eds.). Blackwell Scientific Publications Ltd, Oxford, Great Britain.

Dijkstra, A., Huls, C., Claassen, W., Cremers, L. and van Vliembergen, E. (1990). Grammar Checkers Checked, SPIN-MMC Research Report no. 9. NICI, Nijmegen.

Dijkstra, A.and Huls, C. (1992). Assessing the Usefulness of Linguistic Representations in ESE's, SPIN-MMC Research Report no. 28 NICI, Nijmegen.

Huls, C., Dijkstra, A. and Claassen, W. (1991). A Structured Design of Word Processing Functionality, SPIN-MMC Research Report no. 12. NICI, Nijmegen.

Huls, C. and Dijkstra, A. (in press). A Structured Design of Word Processing Functionality, Proceedings of the HCI '92 York. Cambridge University Press.

Hutchins, E., Hollan, J. and Norman, D. (1986). Direct manipulation interfaces. In Norman D. and S. Draper (eds.) User centered system design: New perspectives on human computer design. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Moran, T. (1983). Getting into a system: External-internal task mapping analysis. Proceedings of the CHI '83 conference on human factors in computing systems, 45-49. New York: ACM.

Norman, D. (1986). Cognitive Engeneering. In: D. Norman and S. Draper, (eds.) User Centered System Design: new Perspectives on Human-Computer Interaction. Hilsdale, NJ: Lawrence Erlbaum Associates.

Payne, S. (1985). Task-action Grammars. In: B. Shackel (ed.) Human-Computer Interaction - Interact '84. Amsterdam, Elsevier science publishers B.V. 527-531.

Payne, S., Squibb, H. and Howes, A. (1990). The nature of device models: the yoked state space hypothesis and some experiments with text editors. Human-computer interaction, 5, 415-444.

Terhorst, N. (1992). A linguistic editor, SPIN-MMC Research Report no. 26. NICI, Nijmegen.

Waern, Y (1991). On the microstructure of learning a wordprocessor. Acta Psychologica 78, pp 287-304.

Waes, L. van (1988). De invloed van tekstverwerkers op het revisiegedrag. In: Taalbeheersing in Ontwikkeling, F.H. van Eemeren and R. Grootendorst (eds.). Foris Publication, Dordrecht, Holland.

Waes, L. van (1991). De computer en het schrijfproces: De invloed van de tekstverwerker op het pauze- en revisiegedrag van schrijvers. (PhD-thesis) Universiteit Twente, Wijsbegeerte en Maatschappijwetenschappen, Enschede, The Netherlands.

Williams, N. and Holt, P. (eds.) (1989). Computers and Writing. Blackwell Scientific Publications Ltd, Oxford, Great Britain.

Adaptively supported Adaptability[1]

Reinhard Oppermann
Institute for Applied Information Technology (FIT),
German National Research Center for Computer Science (GMD), Sankt Augustin, Germany

Abstract

The paper presents an adaptive and adaptable system and its evaluation. The system is based on a commercial spreadsheet application and provides adaptation opportunities for defining user and task specific commands as new menu entries or key shortcuts as well as facilities for changing parameter defaults. Adaptive tips are presented in parallel to the continuing task performance by graphical and sound icons and are accompanied by an environment for carrying out user initiated adaptations. The development followed an evolutionary design-evaluation-redesign approach. The process has shown that adaptations are accepted if the user has the opportunity to control their timing and content. This does not necessarily mean that the adaptation is initiated and performed by the user alone (adaptability). On the contrary, the strictly user controlled adaptation is too demanding for the user. The user is distracted from the real task and does not identify when and what to adapt. Adaptations by the user have to be supported. The support, if performed by the system, can prepare the user's own adaptations by initial adaptive suggestions showing the rationale of adaptations, i.e. examples of efficiency improving changements of the system's interface and the way how to perform them. Adaptations have to be performable in the workstream of the current task, supported by appropriate tools, and the adapted functions being integrated in the system and being presented in a way easily to survey. Adaptations can be task and user specific but the user should be able to define and select different versions of the adapted system for different classes of tasks. The current version should be displayed obviously and being interchangeable within groups of users, it should not undermine cooperative work.

1. Introduction

Computer applications tend to dominate the working environment of many people. Through the efforts of computer scientists, domain experts, and human factors scientists computer systems became more than calculators and support systems for formalized routine tasks. Complex unstructured problems are supported by computerized systems. The complexity of the user's task is reflected by the complexity of the system. A new kind of applications is designed to become an assistant of the user rather than a rigid tool. An assistant performs a different behaviour than current systems do. An ideal human assistant adopts the perspective of the client: The assistant learns about the client, knows about the user's needs, preferences, and intentions. What can a technical assistant adopt from the user? In human-human communication both participants can adapt their own behaviour according to the characteristics of the partners. There are many initial cues about the features of the person, the intentions to open a communication, and the kind of communication style. The communication process enriches and refines the knowledge of both partners about each other: they both learn from the utterances about the partners, their personality and their communication aim and style. The knowledge enables the partners to improve the communication process, to acceler-

1 The contribution was produced as part of the research project SAGA (a German acronym for "Software-ergonomical analysis and design of adaptivity"). The project was supported by a grant from the "Work and Technology" (former: "Humanisation of Work") Fund of the Ministry of Science and Technology of the Federal Republic of Germany and is a part of a GMD research programme about „Assisting Computing".

ate the discovering of (common/conflicting) aims, to optimize the efficiency of the communication, and to increase the satisfaction of the partners. So far human-human communication – what about human-computer communication? Why not to transfer this effective mechanism to the interaction between a user and a system? Why not to model the characteristics of the user in a knowledge base and to let the system learn about the user – to adapt the system to the user? In fact, for ten years there have been efforts of computer scientists to develop modelling methods and adaptive features. But the attempts did not result in a final conclusion – they show at best anecdotal impressions. This paper is to discuss problems and possibilities of adaptive systems and present an example of an unpretentious but successful adaptive behaviour.

Aim of adaptivity

Application systems are not designed for a single user and a particular task. They are designed and distributed for a class of users and diverse tasks. This is true for standard systems, this is even true for special systems where the individual users change and the tasks vary and evolve. Customization capabilities expand the flexibility of the system to endure from the design phase into the usage phase, to make the user more independent of the designer, and to avoid to force the designer to decide about user specific optima. The flexibility of the system's features increases the degrees of freedom of usage, improves the correspondence between user, task, and system characteristics, and optimizes the mental or operational workload of the user.

Different kinds of adaptivity

Adaptivity does not necessarily mean an automatic adaptation of a system to the user. It is an attractive objective to provide the user with support facilities for tailoring the system according to particular tasks and needs. There is a wide spectrum of tools and methods in commercial applications to customize the interface of the system. Special menus or key combinations open the access to interface presentation, naming, and dynamics. There are macro mechanisms to combine functions. The facilities can be used by the „end-user", by a „super-user", or a system consultant (cf. Edmonds, 1981).

Adaptive or, better, auto-adaptive systems modify the interface or the system behaviour referring to how the user interacts with the system. An adaptive system must have knowledge about the system, its interface, the task domain and the user (cf. Norcio/Stanley, 1989). It must be able to match particular system responses with particular usage profiles. Edmonds (1987) describes five areas the system can take into account: user errors, user characteristics, user performance, user goals and the information environment. He suggests the user errors to be the most prominent candidate for automatic adaptations. Benyon/Murray (1988) propose adaptations of the system to the environment of the user at the interface level rather than the user's conceptual task level. Salvendy (1991, 64) refers to personality traits such as field dependence versus field independence (cf. also Fowler/Macaulay/Siripoksup, 1987) or introversion versus extraversion.

Elements of adaptivity

Adaptive systems, i.e. their architecture, their behaviour, their application, and their limitations can be described as generally consisting of three parts, an *afferential*, an *inferential*, and an *efferential component*. This nomenclature refers to a simple physiological model of an organism with an afferential sub-system of *sensors and nerves* for internal and external stimuli, with an inferential sub-system of *processors* to interpret the incoming information, and with an efferential sub-system of *effectors and muscles* to respond to the internal and external stimuli. Contrary to mechanical systems a biological system is capable of reflecting the input by considering the meaning and the context of sequences and not only to react to an input pattern with a conditioned response or a monosynaptical reflex.

Afferential component of adaptivity

Adaptive systems observe and record the user behaviour and system reactions on a metalevel in addition to directly executing a function according to the command input. They protocol regular interaction patterns, frequently occurring errors, ineffective courses and so on. The protocol can gather data on different levels. It can include information about key strokes, mouse draggings and clicks, it can record information about functions or error types, and it can contain task specific information.

Inferential component of adaptivity

Adaptive systems analyze the data gathered to draw conclusions, i.e. to identify from user behaviour possible indicators for adaptation. This is the most crucial point. The inferential component defines what specific system behaviour – other than the initial behaviour – should correspond to the actual usage profile. This implies to specify a basis (theory, set of rules) for drawing inferences. This also implies defining the kind of data to be protocolled (afferential component) and defining how the system should be adapted (efferential component). Thus, the inferential component is the switchbox of an adaptive system. The inference of adaptations can be based on rules and heuristics that are represented in a model of appropriate relationships between the system and the user. The adaptive system can describe the system characteristics in a system model, the user characteristics in a user model, and the task characteristics in a task model (cf. Norcio/Stanley, 1989).

Efferential component of adaptivity

Adaptations lead to modifications of the system's behaviour. The changing may concern different presentations of objects, functions or tools, different default values for parameters, different sequences of dialogue, or different system messages (status, execution, help, error messages). The changing can have two different target objects, the *application itself* or the *error and help system* . The effect of the adaptation can also be differentiated by the degree of determination of the changes, they can result in one *particular effect* being offered by the adaptive system or in a *proposed range of alternatives* to be selected by the user.

Deterministic adaptations of the application that change the system's behaviour in a definite way are considered to be most critical. In this case the requirement for the inferential component to infer a specific outcome from a specific usage profile in a strict if-then-else relationship overburdens the inferential component due to possibly unforeseeable contexts influencing the value of an intervention. The user only has the choice to accept or to reject the adaptive decision. On the other hand, proposals of a *range of alternatives with different views on the system* are the least problematical alternative. They are less ambitious and leave more degrees of freedom to the user. The user can make use of an introduced set of alternatives and can navigate through a space of proposed possible adaptations.

object of adaptation: degree of determinism:	application (presentation of objects, sequence of dialogue, defaults of parameters)	error and help system
one-shot	most critical	medium
range of alternatives	medium	less critical

Figure 1: More or less critical adaptive behaviour

2. State of the art

Although there are attempts to find psychological theories to describe human characteristics that are of more or less persistence and relevance for the adaptation process (cf. Norcio/Stanley, 1989; Van der Veer et al., 1985; Benyon/Murray, 1988; Benyon/ Murray/Jennings, 1990), to our knowledge no working applications have been developed for the relationship between personal *trait* and system behaviour. The most solid example for a relationship between personal *behaviour* (without an underlying general personality trait) and system behaviour is a system reported in Brown/ Totterdell/Norman (1990). The adaptive feature of the system is concerned with the correction of misspellings. The users can be described by their error frequency (missing letter, extra letter, wrong letter, transposed adjacent letters) and be supported by appropriately arranged lists of suggestions for correction. The limitation of this example is obvious. It can not be generalized and the advantage is limited to reduced milliseconds of correction time. The same kind of limited advantage provide the adaptive menus in the version of adaptive menu defaults where the mouse cur-

sor is pre-positioned at the user's most likely selection (cf. Brown/Totterdell/Norman, 1990). In another example the adaptive effect is a reordering of menus where the most frequently used menu options are presented on top of the menu. This leads to a decreased access time but also – especially in the learning course of the system – to an increased search time (cf. Mitchell/Shneiderman, 1989).

Proofs for the feasibility and accountability of adaptive systems with respect to *personality traits* such as introversion/extraversion, field dependence/field independence (cf. Salvendy 1991) where relevant system behaviour corresponds to personality differences have yet to be shown – I wouldn't be at all surprised if we had to wait for them very patiently!

3. Adaptively supported adaptability – an example

Considering the difficulties in developing adaptive systems that respond to personality specific traits we tried to concentrate on *task specific regularities* that could be related to differentiated system behaviour. Considering the existing examples of adaptive systems in ad hoc prototypes with limited functionality we started with a complex application with an *advanced user interface*, the commercial spreadsheet program EXCEL™. This program offered opportunities for afferential observations and efferential interventions. The *afferential* opportunity was a protocol of the user's actions as the basis for inferences from user actions on user needs. The *efferential opportunity* was the possibility to build a customized application consisting of a set of functions represented by macros with an authentic or modified functionality of the basic system. The protocol of the user actions could be interpreted by rules on a coprocessor in the background and be reflected in appropriate system reactions.

The concept of the adaptivity study was to provide intervention facilities for the interaction with the spreadsheet application. Two kinds of adaptation facilities were developed. The first kind was a tool to adapt the system on the initiative of the user. The second kind was auto-adaptive with the system taking the adaptation initiative. These two components should work together and allow the user to perform or at least to accept/reject adaptations. The user should have the opportunity to do his or her own adaptations with the adaptation tool. The adaptive component should present suggestions proposing modifications of the interaction appropriate for the actual use of the system. The user may select from offered suggestions. The user's control includes two aspects. First is the opportunity to work *autonomously* without having to wait for or being dependent on the adaptive component. Second the user has the *authority to reflect on, to accept and to reject* the system initiated adaptation suggestion. The adaptation is subject to an explicit decision of the user.

3.1. Design of the adaptation

The experimental system with adaptation opportunities called Flexcel („Flexible Excel") was developed on the basis of the EXCEL™ program. The system was implemented on a MacIvory™, a two processor machine. The application EXCEL™ runs on the Macintosh™ part whereas the part of the system evaluating user actions has been implemented on the LISP-machine. These two parts of Flexcel communicated in writing and reading from files: a protocol was written by the Macintosh part, inference results were returned from the LISP-machine. The experimental system is a customized spreadsheet application that provides most of the functionality of the original Excel (some more than the basic version with „Short Menus"). The functions were designed in part in a modified way to allow different types of adaptations:

- altering given parameter defaults and
- defining key shortcuts and new menu options for different function parametrizations.

These types were selected as a result of an analysis of tutorials and authentic tasks for adaptation possibilities in the selected spreadsheet system (cf. design cycles in chapter 3.2). For each type of adaptation the relevant functions were identified and provided with adaptive and adaptable facilities. The selection encloses the following Excel functions: Clear; Paste special; Insert cells; Delete cells; Number format; Alignment; Font; Border; Extract data.

The experimental system Flexcel includes an adaptable and an adaptive opportunity. The access to these facilities is provided in a separate tool bar beside the worksheet (cf. figure 2) that serves as the main „control panel" for the adaptations. The adaptation facility is also accessible via a check box in the dialogue box of each adaptable function (cf. figure 4).

The rationale of the adaptation environment is that the user is provided with the facility to define the new parameter default for the given function or to define the new menu entry or key shortcut when using the spreadsheet for tasks that require executing a function repeatedly with the same parameter(s). An example may illustrate the idea: a user frequently needs to add the value of a cell to a set of entries of other cells. This normally requires selecting the function „paste contents" from the menu, then specifying the parameter „values" and „add" and finally click the „OK" button in the dialogue box. In Flexcel, the user can eliminate this dialogue step by defining a new menu entry and/or a key shortcut for „add value". The adaptive mechanism works rule based inferring user needs from usage regularities. Controlled by variable thresholds the use of identical parameters for a given function is prompted with an adaptive suggestion to define a key shortcut or a menu entry respectively according to (a) the user's preferred use of key shortcuts or menus and (b) the user's error record handling key shortcuts. The knowledge base consists of a set of rules and a dynamic usage profile (cf. Krogsæter/Brüning/Thomas, 1992 for details).

The adaptation effect is actually limited to the altering of parameter defaults and defining shortcuts of dialogues by new menu entries or keystrokes. The limitation is due to restricted intervention opportunities into the basic spreadsheet Excel™. In designing the adaptive system there has been a tradeoff between a limited adaptation spectrum and a rich functionality for realistic usage evaluations or a large range of sophisticated adaptations in an arbitrary experimental environment. A decision was made in favour of the first alternative to show and test a small but realistic adaptation mechanism that can principally be used for authentic tasks with a commercial application. The idea of adaptive features can be transfered to a larger range of adaptation effects.

The adaptations are accessible via the separate tool bar shown in figure 2.

Adaptive suggestions are presented as „tips" and are indicated by a sound icon, 3 times blinking of the button „The Tip"in the adaptation tool bar and a following presentation of the icon with a corona. In order to read the tip, the user can click the button.

The unread tips are collected in a „Tip List" being addressable by the user at any appropriate time.

The adaptability is accessible with the button „Adapt" which brings the user into a selection box of all adaptable functions.

The button „Overview" displays a summary of all user-defined keystrokes and menu entries.

The „Critics" button may be clicked, for instance, at the end of a session, or whenever the user wants the interaction with the adaptation environment to be analyzed. The critique given is constructive, telling the user in which way the adaptation tools can be used more effectively.

The „Tutorial" button starts a tutorial explaining how to use the adaptation facilities.

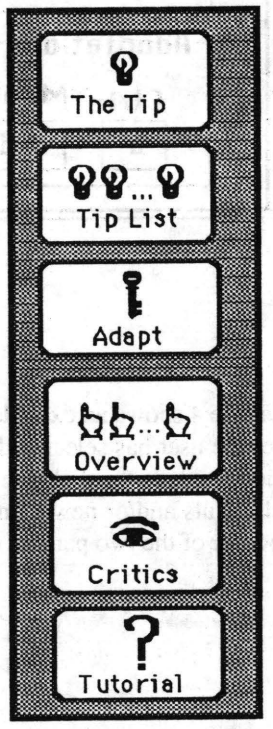


Figure 2:
The Flexcel
adaptation tool bar

Flexcel provides three kinds of suggestions: „adaptation tips", „usage tips" and „tutorial tips". An *adaptation tip* recommends the user to define a menu entry and/or a key shortcut for a function parametrization

after it has been repeatedly executed with identical parameters. A *usage tip* reminds the user of a self-defined adaptation that seems to be forgotten. A usage tip may appear after a function has been repeatedly executed with a parametrization for which there already exists a menu entry and/or a key shortcut. The *tutorial tip* augments the user's capability to proceed from the adaptive to the adaptable concept by recommending a tutorial that explains how to adapt the system. A tutorial tip appears if the user regularly accepts adaptation suggestions and uses the adaptations, but does not adapt independently.

Figure 3 shows the presentation form of a tip that may appear after clicking the „The Tip" button in the tool bar. As can be seen, even if the system recommends defining a new menu entry the user has the possibility to additionally or alternatively define a new keystroke for the suggested parametrization.

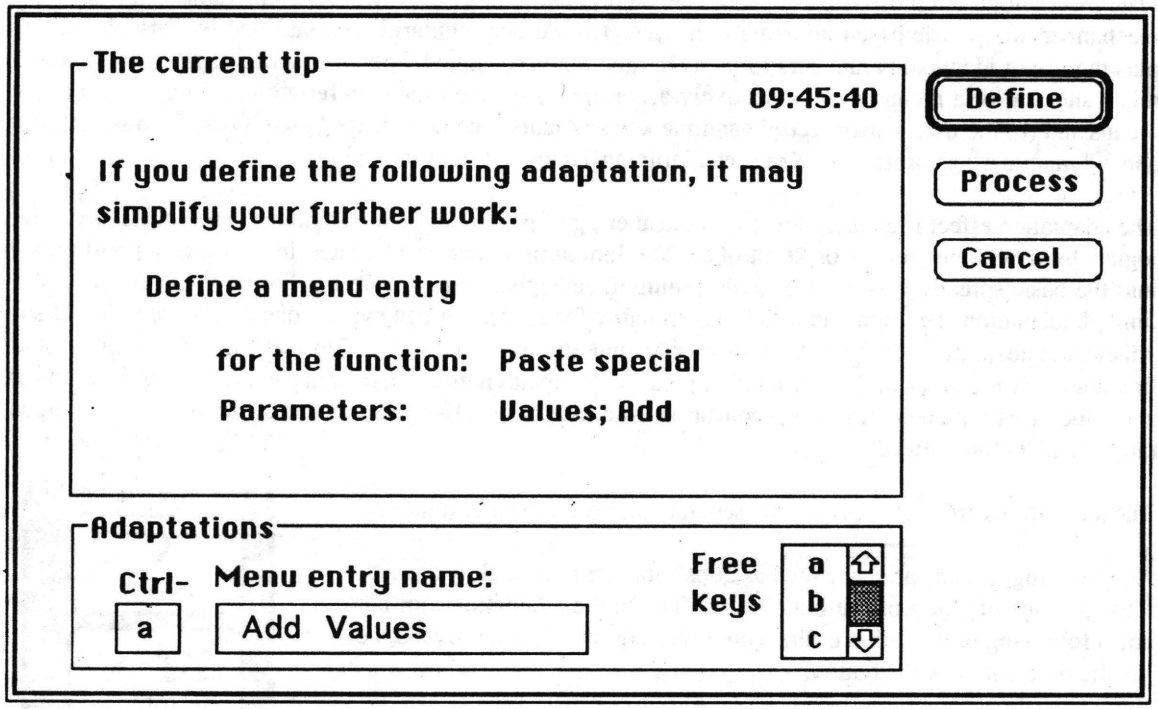


Figure 3: A Flexcel adaptation tip

Figure 4 shows an extended dialogue box for a user initiated adaptation of „Paste special" as it appears after the user has selected the control field „with adaptations" in the upper right part of the box. This selection expands the dialogue box to a combined execution and adaptation box where the user can define key shortcuts and/or new menu entries for the selected parameters in the upper part of the dialogue box (the border of the two parts of the dialogue box is indicated by a dotted line for illustration in this paper only).

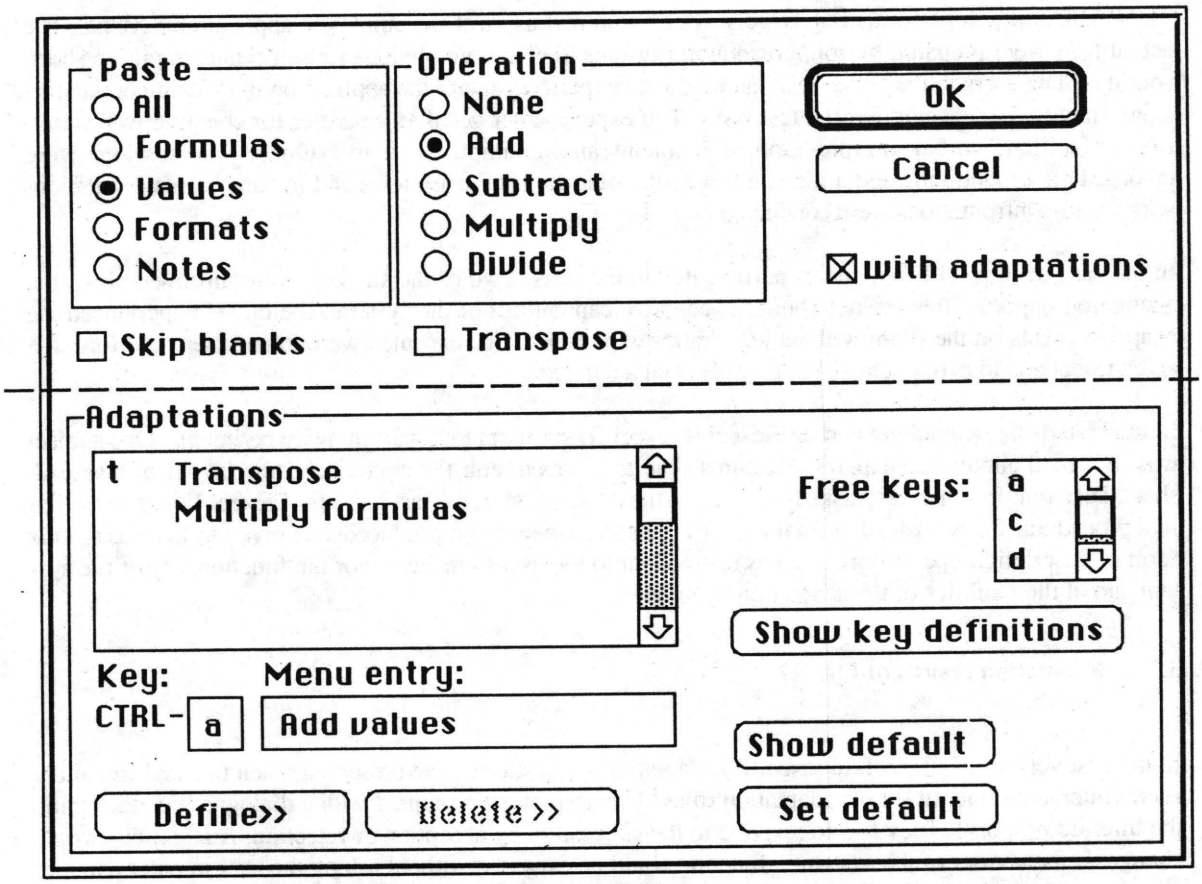


Figure 4: Combined execution/adaptation dialogue box for the function „Paste special"

The adaptation part of the dialogue box in Figure 4 is basically the same for all adaptable functions. The user may enter parameters in the upper part of the box; then a key shortcut and/or a menu entry for the specified parametrization can be defined. The user can do so subsequently for different parameter combinations (symbolized by the >> with the „Define" button). The user can select a free key from an actual key list or directly type in a key. A list can be inspected with the actually defined entries for parametrizations within the function at hand and an overview of shortcut definitions of the entire system can be called.

3.2. Design-evaluation-redesign cycles of Flexcel

The design of the adaptation concept of Flexcel was not entirely specified in advance. Rather it reflects results of design–evaluation–redesign cycles. The process started with a first version of Flexcel (Flexcel I) based on the analysis of adaptive and adaptable concepts in existing systems or prototypes and general claims of human factors research (cf. Oppermann et al., 1992). This initial version showed different features in timing and presenting adaptive suggestions not described in this paper in full detail.

The evaluation was not performed according to a classical experimental setting. The evaluation method was selected in accordance with the question to be answered. This was a question of „What is bad with the system and why?" rather than „How good is the system?" where different alternatives are compared under certain aspects or a given system is tested against fixed criteria—cf. Fähnrich/Ziegler (1987). The aim of the evaluation was to identify the cognitive model users develop in interacting with the adaptation component of the system and what problems they have in using and understanding its mechanisms and behaviour. The evaluation followed an explorative approach using „weak methods" like video observations, video confrontation, and interviews (cf. Carroll and Campbell, 1986). Experiments are characterized as "hard" methods when statistically analyzable data like time and error are collected (quantitative, theory driven—cf. Newell and Card, 1985; see also Oppermann, 1991).

The evaluation was performed in usage sessions with real users of the employed application EXCEL. The actual tests were prepared by some reconnoitring runs to eliminate obvious technical and cognitive shortcomings. The users in the actual test had moderate experiences with the application in its commercial version. They had to execute written test tasks. The experimenter acted as a partner for constructive interaction to facilitate verbal interpretation of problems and assumptions of the subject. The sessions were recorded on a video-tape and a log-file protocol. After the tests interviews and in some cases discussions with video-confrontations were conducted.

In the case of Flexcel I five[2] users participated in the tests. Two of the subjects were informed about the adaptation opportunities but not about the adaptive capabilities of the system. The others experienced the adaptive events on the flight without preparation. The tasks to be executed were to complete and format a spreadsheet and to extract classes of records from a database.

In the case of the second version of Flexcel (Flexcel II) six users took part in the experiments. One subject was informed about the adaptation features. The task was to edit the content and the layout of a spreadsheet. Different from the test tasks for Flexcel I the description of the test tasks for Flexcel II was not given in a procedural form. Instead, a final form of the spreadsheet to be produced was given to the users. This form of description opens more degrees of freedom to the user to make use of the functionality of the system and of the facilities of the adaptation component.

3.3. Evaluation results of Flexcel

In the first version of Flexcel („Flexcel I") adaptation suggestions were presented when the analysis of the user's interaction met (a set of) adaptation rules. The users were prompted with a dialogue box describing the inferred proposal. They had to respond to the suggestion by acception or rejection. Adaptations would change the behaviour of the system: a function could be executed without asking the user to enter parameters, it could be given different parameter defaults, and changed key shortcuts could be used. Tests have shown that users can easily get confused by such changes when using the changed function in a later phase of the session even after explicit acceptance. Adaptations are performed after learning the basic concepts and the basic behaviour of the system when proceeding to routine tasks. They confront the user with problems similar to those with different system modes. The user has to remember initial and adapted versions of the system and this difference has to be handled with care when being suggested and has to be supported when performed. Our solution in the first version of Flexcel was insufficient in this respect. With an adaptation proposal the user was presented with a suggestion, asked for confirmation, and informed about the later different behaviour of the system. Hence, the user was confronted with three kinds of cognitive demands:

(a) to turn from task execution to an adaptation suggestion and to devote the attention to a metatask;
(b) to evaluate the suggested configuration of the system according to present and future tasks and personal preferences and to decide about acceptance;
(c) to grasp and memorize the described different features and different handlings.

This is a cognitive overload, a concentration of too many mental perspectives in a single situation. From each cognitive problem domain we derived specific design requirements for the second version of Flexcel („Flexcel II").

3.3.1. Task performance and adaptation

The switch from the task to the adaptation should not be the exclusive business of the system. Firstly this means that the system should never force the user to interrupt his task and perform an adaptation or even consider it in reaction to an adaptation suggestion presented by the system. In Flexcel I we confronted the user with a dialogue box that intervened the current workflow/workstream. This was too coarse. In Flexcel II we presented the adaptation in a separate tool bar. The presence of an adaptation suggestion was indi-

2 No more tests were necessary to identify cognitive and operative problems and design requirements.

cated with (a) a sound icon, (b) a three-times blinking tip button, and (c) a subsequent continuous corona with the light in the adaptation tool bar. These combined cautious hints to an actual adaptation tip did not interfere the sequence of operations. The tip can be ignored or taken up at any appropriate time. Tests of this presentation showed increased users' acceptance, because the users could wait until their current workstream allowed an interruption to switch over to the adaptation task.

3.3.2. System or user initiated adaptation

If the adaptation should not only be the business of the system, the users must have their own opportunities to perform adaptations – not only when the system suggests them. In most cases, the adaptation suggestions motivated users to make their own adaptations, i.e. to transfer and generalize the adaptation suggestions. Suggestions actively show adaptation possibilities. Thereby, the system demonstrates to the user in which cases adaptations may simplify the work and so inspires the user to adapt the system independently of system suggestions. The users want to be able to use their own adaptation facility to overcome the dependency on the adaptive component. This tendency has to be supported because users have difficulties to detect and learn the appropriate access and handling of the respective tools. In Flexcel I we included an adaptation check box in each dialogue box of adaptable functions. No test user identified this button as an adaptation access. Even after they had become familiar with the adaptation concept in form of adaptive suggestions and even when asking for own adaptation opportunities in constructive interactions with an instructed co-user they failed to see the concept. More prominent presentations of adaptation facilities or explicit instructions were necessary to introduce the adaptable concept to the user. More spectacular presentations by eyecatching codings (colour, size, placement, movement) compete with the presentation of other concepts and tend to yield an inflation of prominence. In Flexcel II we presented different accesses to the adaptation: the adaptation check box in the dialogue boxes („local access"), an additional adaptation button in the generic tool bar („global access"), and an auxiliary adaptive tip when the user showed an acceptance and use of adaptive suggested adaptations but did not make use of the own adaptation opportunities. These tips are called „tutorial tips" and describe the access button for the adaptation and refer to the tutorial button of the tool bar for details. They actively present the proposal to adapt on their own or to consult a tutorial to get familiar with the adaptation tools. The evaluation of the transition from adaptation suggestions of the system to adaptations by the user of Flexcel II is better than of Flexcel I but not yet satisfying. The problem is less severe after the user has been instructed via a tutorial tip or personal communication.The first self-initiated adaptation renders difficulties.

We think that we are on the right way but have to improve the transition from adaptation by the system to the adaptation by the user to locate the „locus of control" for the adaptation with the user. The user can decide if and when to respond to adaptation suggestions. The user can select an adaptation of a sample of options. And the user has access to own adaptation opportunities (via a button in the tip box, a button in the adaptation tool bar, and a check box in each dialogue box of the adaptable functions).

3.3.3. Accuracy of adaptation: the user's degree of freedom

When presenting adaptation tips the appropriate time and content for the adaptation is significant. The user has particular tasks and particular personal needs and preferences that have to be reflected by the adaptation proposal. This asks for an indepth analysis of the human-computer interaction. The analysis not only has to reflect the current situation of the dialogue but also the usage history of the specific user to reach a high accuracy of adaptation suggestions to the actual user needs. We considered the history of the usage with respect to the user's interaction preferences (preferred use of menus or keys) and the user's adaptation preferences (acceptance/rejection of adaptive tips, use of performed adaptations, relationship between system and user initiated adaptations) according to a rule base described in Krogsæter/Brüning/ Thomas (1992). The evaluation of the use of the different versions of the system has shown that it is hardly possible to meet the exact needs of the users. Not only machines, even human advisers have problems to meet these criteria as shown in Wizard of Oz-settings in our laboratory. In these situations users had to perform a test task and a system expert observed the session via a system-system connection so being restricted to the information of the technical interaction without verbal communication, gesture, and using external task or system documentation. In a critical situation the human adviser could only derive a set of hypotheses about problems and needs of the user but not a reliable decision about the exact support requirement. In consequence an adaptive system has to be modest and not fire one-shot solutions. It should „place one in an information space" (Fischer/Henninger/Nakakoji, 1992, 5) like system-initiated delivery mechanisms of help

systems. The users have to be presented with a spectrum of adaptations for navigating and selecting an item from a sample of options. It is challenging enough to compose the relevant items for the possible user support. Tests show that the users are ready to process a sample of suggestions if the spectrum is relevant for the tasks and needs of the users. To present a sample of suggestions augments the motivation and the consciousness of the user to consider appropriate adaptations. Experiments could test whether, with a given sample of adaptation items, the user is directed to go beyond the range prespecified by the system looking for earlier adaptations or other adaptation features. Adaptation suggestions can induce an explorative use of tools to customize the application. In our test sessions there is support for the hypothesis that, having a degree of freedom when confronted with adaptation tips, users will tend to look for more autonomous and competent system configuration than when faced with a singular proposal. This also facilitates the understanding of users having experienced some adaptation proposals from the system and expecting equivalent suggestions for other operation sequences *or* easily accessible adaptation tools.

3.3.4. Performance and control support of adaptation

Supporting the adaptation

To perform an adaptation efficiently and effectively as well as to visualize and eventually modify its results in later sessions the user needs support from the system. The tests have shown that the support has to cover different aspects.

- It has to contain a short and clear description of the rationale of the adaptation: its reason, its content, its consequence.
- It has to present a comprehensive adaptation environment: selection and definition opportunities of adaptations.
- It has to offer a survey of performed adaptations: visualization of new/changed functions, new/changed system dynamics, new/changed system presentation.
- It has to contain an environment for later modifications of performed adaptations.

We learned from user observations that adaptations are iterative processes not being performed in one definite step. Sometimes they are triggered by a specific adaptive tip or performed autonomously due to a specific task condition. Sometimes they are modified, extended, or completed in the course of a quasi excursion unspecific for the actual task devoting the attention to the customization of the work environment. We identified the demands mentioned above partly in an earlier study about the use of adaptable systems (cf. Karger/Oppermann, 1991) as a starting point for the design of Flexcel I. We improved the initial concepts in the evaluation cycles and tried to support them with respective design features. In the first version of Flexcel we presented all information in one dialogue box describing why and what the user could adapt with which effect (cf. 3.3). This was insufficient because the user had no direct view on the modifications of the system. The modified function replaced the original function which could only be executed with its former effect by a key combination to be memorized by the user in the course of the initial adaptation. Definitions of new key shortcuts had to be entered without an overview of the already set definitions. Uses of already defined keys were prompted with a message opening a question and answer dialogue. An overview of new key entries was only accessible via a separate menu option that none of the users actually used in the test sessions. In Flexcel II the appropriate adaptation design could be better accomplished[3]. When prompting the user with an adaptive tip the dialogue box showed the relevant parameters for the function to be adapted and the exposed entry fields for shortcuts with a selection list of free items and a text field for new menu entries. The user had the opportunity to define the adaptations directly according to the derived proposal or could turn to the complete adaptation environment for the given function (cf. figure 3). This adaptation environment was consistent from all possible accesses: from the adaptive tip by clicking the „process" button, from the tool bar by clicking the „adapt" button and selecting the respective function, and from the dialogue box of the function by clicking the control field „with adaptations" (cf. figure 4). The control facilities in the adaptation environment (cf. section 3.1) offered all relevant options and exposed all relevant information to the user at a glance. Defined adaptations were displayed in a selection list showing the respective parameter(s) of the function when selected. Defined adaptations were integrated in

3 The difference was due to the experience of empirical test of Flexcel I and to more powerful design opportunities for a customized application for Flexcel II because fortunately we were provided with the new version 3.0 of EXCEL when proceeding from Flexcel I to Flexcel II

the presentation of the system by including them in the menu indented underneath the function entry. This was selected from a variety of options because it has the advantage to preserve the task oriented context. The user only has to memorize the location of the basic function; all derived adaptations are exposed in this environment and form a comprehensive family.

Giving adaptations a name

The test users understood the adaptation environment easily and handled the tools after shortly scanning the system. The only serious problem for the users was the decision whether to define a key shortcut or also to include the new adapted functions into the menu and how to name the new menu entry. The adaptations appeared to be an ad hoc action with relevance only for the task sequence at hand[4]. A shortcut or the unconcerned definition of a menu entry seemed to be sufficient. The naming of the command was sometimes reduced to a nonsense set of letters quickly and dirtily entered. After own experience with difficulties in remembering the meaning of the arbitrary menu entries users criticized their own product and tried to replace the given name and used better option names later on. Also after the experience the users had difficulties in finding really meaningful characterizations. In very few cases only a composition principle could be identified for the given names.

Criticizing adaptations

To criticize the naming of new functions can only be reflected by the user. This is motivated when difficulties with unintuitive names are experienced. System capabilities are far from producing appropriate naming or criticizing user defined ones. An advisory component could be of help in explaining the rationale of names to be composed on an abstract level, for instance: „Use characteristic and discriminating elements for entry names". Other aspects of the adaptation can be criticized by the system in an interactive sense deducing violations of standards from the user's actual interaction pattern. Inferences can be made, for instance, about the adequate use the user makes of the adaptation facilities. This means that the system can compare whether a user identified as a „menu user" defines new functions with respective menu entries and a user identified as a „key user" defines new functions with key shortcuts. In the critique component of Flexcel II constructive advice is prepared for seven possible improvements in using the adaptation facilities: „clear the adaptations", „use adaptations", „adapt on your own", and so on. Users addressed to the critique in the test sessions only from sheer curiosity. This corresponds to the design principle of the critique to reflect a longer period of interactions and not to help in actual problems. The contribution of the improvement of the adaptations by the critique can only be evaluated in long term sessions.

Control over the adaptation profile

When a system is adapted the result is specific for a task and for a user. This does not mean that the adaptations are only specific for a single task and not even for the single user alone. Adaptations may meet the requirements of a class of tasks for a group of users. So, adaptations should be controllable. In Flexcel II this was permitted by two supplementary entries in the file menu: „save profile" and „load profile". The „profile" stores all adaptations of the basic system the user has performed at some time: all additional menu options, shortcuts, and parameter defaults. The user gives the profile a name which is displayed in the header of the main menu right of the last entry. The user is always informed about the current profile and can load another one when turning to a different task. The profile is stored on an external floppy disk. For loading a profile the user has to insert the disk and to select „load profile" from the menu for the respective file. The profile separates the resulting adapted system from the application and from the user documents. The concentration in a summarizing profile opens the possibility to contribute to the privacy of the sensible data of the usage protocol. The user has the control over the data in taking the disk with sensible information that allow conclusions about working style, preferences, fuzziness, errors, and so on. The user can put the disk into the pocket if necessary.[5]

---

4 This is possibly due to the test condition in the lab and it would be a question for a long term experiment whether the effect disappeared when the users define adaptations for their authentic tasks.

5 This is certainly not an ideal solution for practical considerations because most of the users have no or no „free space" in the pocket. But even if the user has the discette at the personal work place the profile is not accessible for centralized inspection and evaluation.

In the discussion of the profile concept with the test users the presented solution was unanimously appreciated both for reasons of providing a comfortable access to (different) user/task specific profiles and for reasons of privacy. Further research is needed for the conceptualization and long term evaluation of individual and group oriented exchange facilities to shared adaptations and its acceptance. Ideas of a „purse" for the exchange of profiles with successful adaptations for groups of users with similar tasks are discussed.

3.4. Discussion

The presented adaptive system differs from existing examples in providing mechanisms to adapt the user interface of a commercial system other than at the level of context-sensitive help (cf. Fischer/Lemke/Schwab, 1985). Based on empirical analyses of the potential of the application for adapting the interface to task and user specific needs the identified spectrum of adaptations was realized. Adaptations were realized in an adaptive and an adaptable version to analyze the relative benefit of the two alternatives. The most important result of the study is that adaptive and adaptable systems are *no* alternatives. Adaptable features are–at least at present–no common part of the user interface. Users are not (yet) familiar with the existence, handling, and gain of adaptabilities. Many systems provide no or only very limited adaptation capabilities. Adaptive features keep the users in dependence of suggestions with respect to time and content of (tips for) changes. Users deny to accept to be at a system's mercy. This holds true for two dimensions. The dependence on adaptive features can (1) intervene with a cognitive and operative model the user develops or has developed about the system. The dependence on adaptivity (2) also produces an emotional suspense.

The cognitive and operative interference was demonstrated in other studies about adaptive systems. Mitchell/Shneiderman (1989) showed disadvantages of dynamic versus static menus where the position of menu entries were rearranged according to usage frequencies of specific users. The most critical point of studies like this using quantitative performance criteria for the evaluation of adaptive features is the time dimension (cf. Browne/Totterdell/Norman, 1990, 167).

- When does the adaptations take place: When a user begins to learn a system? When a user begins first to use the system for authentic tasks? When a user has already got real experience with the system and a system has got experience with the user?
- It is also important to determine the time for the evaluation of the adaptation effects: When the adaptation is being defined? When the adaptation is just terminated? When the user has worked with the adapted system a certain time?

In the presented study we relinquished to use quantitative performance measures but analysed the cognitive understanding and interaction of users with the adaptivity. The introduction of adaptivity follows the basic understanding of the application and the evaluation is concerned with the process of learning and assimilation. We found the evaluation of adaptive and adaptable concepts to be most promising when both features are linked to cooperate. The existing evaluation of adaptive systems test adaptive effects more or less exclusively against static systems (cf. Browne/Totterdell/Norman, 1990, 164). We found the adaptive component to be best suited to prepare the user to adapt the system, to induce a reflection about the suitability of the application, to present clues when to turn from the domain task to the meta-task of adaptation. The emotional suspense mentioned above induces the critical motivation for a latent attention for this kind of task when being transferable into own active adaptation opportunities. The clues of adaptive suggestions („tips") should hold the balance between a massive interruption of the user's workstream like with active help systems and merely mute potential like with passive help systems (for a similar conception for design critic messages also cf. Fischer et al., 1992, 19; Fischer/Henninger/Nakakoji, 1992, 4). This balance is to ensure to exhaust the potential of the adaptivity and to augment the control of the users over their own working tools and working styles. The augmentation of the adaptation capability of the user is supported in the presented Flexcel II in three ways:

- It actively shows adaptation possibilities by adaptation suggestions. Thereby, it demonstrates to the user in which cases adaptations may simplify the work and so inspires the user to adapt the system independently of system suggestions.
- It actively presents tips to adapt independently and to consult a tutorial to get familiar with the adaptation tools.

- A critique component can be invoked by the user. The user's interaction with the adaptation tools is then analysed, possibly resulting in suggestions of how to use the tools more effectively.

References

Benyon, David/Dianne Murray/Frances Jennings (1990). An Adaptive System Developer's Tool-kit. In: D. Diaper et al. (Eds.) Human-Computer Interaction–INTERACT'90. North Holland, Amsterdam New York Oxford Tokyo, pp. 573-577.

Benyon, David/Dianne Murray (1988). Experience with Adaptive Interfaces. The Computer Journal 31 (1988), 5, 465-473.

Browne, Dermot/Peter Totterdell/Mike Norman (1990). Adaptive User Interfaces. London: Academic Press.

Carroll, John M. and R.L. Campbell (1986). Softening up hard science: Reply to Newell and Card. User Interface Inst., IBM Watson Research Centre, Yorktown Height, New York.

Edmonds, Ernest A. (1981). Adaptive Man-Computer Interfaces. In: M.C. Coombs/J.L. Alty (Eds.) (1981): Computing Skills and the User Interface. London: Academic Press, pp. 389-426.

Edmonds, Ernest A. (1987). Adaptation, response and knowledge. Knowledge Based Systems 1 (1987), 1, 3-10.

Fähnrich, Klaus-Peter/Jürgen Ziegler (1987). Software-Ergonomie: Stand und Entwicklung. In: Klaus-Peter Fähnrich (Hrsg.): Software-Ergonomie. München: Oldenbourg Verlag, S. 9-28.

Fischer, Gerhard/Jonathan Grudin/Andreas Lemke/Raymond McCall/Jonathan Ostwald/Brent Reeves/ Frank Shipman (1992). Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments. Paper submitted to the Journal „Human-Computer Interaction", Special Issue on Computer Supported Cooperative Work.

Fischer, Gerhard/Scott Henninger/Kumiyo Nakakoji (1992). Helping Designers Identify their Information Needs and Access Relevant Information. Submission to SIGIR'92, Copenhagen Denmark, June 1992.

Fischer, Gerhard/Andreas Lemke/Thomas Schwab (1985). Knowledge-Based Help Systems. Human Factors in Computing Systems. Proceedings of the CHI'85 Conference (San Francisco, CA) ACM, New York, April 1985, pp. 161-167.

Fowler, C.J.H./L.A. Macaulay/S. Siripoksup (1987). An Evaluation of the Effectiveness of the Adaptive Interface Module (AIM) in Matching Dialogues to Users. In: Dan Diaper/R. Winder (Eds.) People and Computers III. Cambridge: Cambridge University Press, pp. 346 - 359.

Karger, Cornelia/Reinhard Oppermann (1991). Empirische Nutzungsuntersuchung adaptierbarer Schnittstelleneigenschaften. In: David Ackermann/Eberhard Ulich (Hrsg.) (1991): Software-Ergonomie 91. Benutzerorientierte Software-Entwicklung. Stuttgart: Teubner, S. 272-280.

Krogsæter, Mette/Inke Brüning/Christoph Thomas (1992). FLEXCEL: Adding Flexibility to a Commercially Available Software System. Paper accepted to Engineering for Human-Computer Interaction. 5th International IFIP Working Conference on User Interfaces. Ellivuori, Finland, August 10-14 1992.

Mitchell, Jeffrey/Ben Shneiderman (1989). Dynamic versus Static Menus: An Exploratory Comparison. SIGCHI Bulletin 20 (1989), 4, 33-37.

Newell, Allen/Stuart K. Card (1985). The Prospects for Psychological Science in Human-Computer Interaction. In: Thomas P. Moran (Ed.): Human-Computer Interaction Vol. 1. Hillsdale, N.J., London: Lawrence Erlbaum Associates, Publishers, pp. 209-242.

Norcio, Anthony F./Jaki Stanley (1989). Adaptive Human-Computer Interfaces: A Literature Survey and Perspective. IEEE Transactions on Systems, Man, and Cybernetics 19 (1989), 2, 399-408.

Oppermann, Reinhard (1990). Experiences with Evaluation Methods for Human-Computer Interaction. Paper presented at the Fifth European Conference on Cognitive Ergonomics (ECCE-5), September 3-6, 1990 in Urbino (Italy)

Oppermann, Reinhard/Bernd Murchner/Harald Reiterer/Manfred Koch (1992). Software-ergonomische Evaluation. Der Leitfaden EVADIS II. Berlin: de Gruyter Verlag.

Salvendy, Gavriel (1991). Design of Adaptive Interfaces and Flexible Mass Production of Knowledge-Based Systems. In: Hans-Jörg Bullinger (Ed.): Human Aspects in Computing. Design and Use of Interactive Systems and Work with Terminals. Amsterdam-London-New York-Tokyo: Elsevier, pp.55-68.

Veer, Gerrit van der/ Michael Tauber/Yvon Waern/B. van Muylwijk (1985). On the Interaction between System and User Characteristics. Behaviour and Information Technology 4 (1985), 4, 289-308.

part 6
USER INTERFACE THEORY

Etag as the Basis for Intelligent Help Systems

G. de Haan, G.C. van der Veer

Department of Computer Science
Free University
Amsterdam, The Netherlands

ABSTRACT

ETAG is a method for representing interactive computer systems for design purposes, on the basis of a representation of what a competent (perfectly knowing) user knows about the structure and use of a system. An ETAG representation is a conceptual model which contains all information a user might want to have of a computer system. As such, ETAG representations may serve as the basis for intelligent help facilities. A number of studies will be reviewed in which ETAG was used to build non-interactive and interactive facilities providing help information about computer systems. In order to provide static -non changing- information about the computer system, the ETAG representations contain sufficient information. To provide dynamic information about interacting with the system as well, the information in ETAG representations needs to be supplemented with history information. In order to cope with dynamically changing information, it proved to be necessary and more widely advantageous to use an intermediate (Prolog) representation of ETAG and history information. In different respects, following a modular system architecture proved to be advantageous.

1. Introduction

When using a computer system, users employ a mental representation of that system, in order to plan how to attain task-goals, to interpret system responses, and to evaluate task performance (eg. Norman, 1983). From this point of view, the design of a computer system should be based on considering the ease with which users can develop and maintain a mental model of the target system. One way of doing this is by designing a system around a hypothetically perfect mental model of a user. This is what Norman (1983) called the conceptual model of the computer system: a correct, complete and non-ambiguous representation of a system. Extended Task-Action Grammar (ETAG) was originally proposed by Tauber (1988, 1990) as a specification method for designing interactive computer systems. ETAG is a method which provides the language, concepts and conceptual framework to build conceptual models. More specifically, the results of applying an ETAG analysis is an ETAG model, which is a representation of the knowledge of a hypothetical "competent" user, who is supposed to have a complete and correct understanding of the structure and workings of the computer system. Note that the concept of a competent user differs from that of the ideal user, who is assumed to perform tasks perfectly. ETAG as a competence model is restricted to specifying the knowledge needed to use interactive systems. ETAG does not say anything about how this knowledge is actually used.

Although ETAG is originally developed for the purpose of design specification (Tauber; 1988, 1990), its ability to represent much of the relevant computer system knowledge on behalf of the user should make it more widely applicable. Formal models can be used for task-analysis, user knowledge analysis, user performance prediction and specification for design (de Haan, van der Veer, and van Vliet; 1991). ETAG can be applied to specify and analyze the knowledge of users of various computer systems, which is required, among other, to predict the places where performance difficulties might arise (de Haan and van der Veer, 1992).

The basic principle that ETAG represents all the knowledge of a system that should and might have in memory makes ETAG a gratified candidate for use in an even more applied role: the development of intelligent help systems. An ETAG model consists of all system knowledge a user might need, and with the addition of a means to interrogate the model it should be possible to build systems to supply the user with this knowledge. Depending on the extent to which such a help system can adapt to the circumstances in which help is requested, it may behave more or less intelligently. In this paper systems will be discussed which use ETAG to provide help information interactively or non-interactively, and to provide static help information about the system and dynamic

help information about previous interaction with the system.

In the next section (2) the structure and content of an ETAG model will be described. Section (3) discusses why to use a formal model like ETAG as the basis for help systems and describes ETAG based help systems in general terms. Subsequently, in section (4) several ETAG based systems will be reviewed which provide the user with either static or dynamic help information. Finally, in section (5) the conclusions on ETAG based intelligent help systems will be summarized and requirements for future research will be formulated.

2. The Structure of ETAG

An ETAG model of a computer system consists of three main components: (1) the User's Virtual Machine (UVM) describes the semantics of the task environment, (2) the Dictionary of Basic Tasks describes the tasks the system provides and the parameters required, (3) the Production Rules describe how the user should specify task commands.

2.1. The User's Virtual Machine (UVM)

The UVM is a formal description of the knowledge of the task world in terms of objects, attributes, relations and events. For example, the 'facts' that files may contain text, and may be moved, removed or copied from the place they take up. In terms of Kieras and Polson (1985) the UVM describes the "how it works" knowledge the user should have.

a) The type specification is a description of the relations between the relevant objects, events, places and attributes of a computer system, in terms of concepts or types. The type specification itself describes what each (instance of a) particular type consists of, such as attributes and storage places for object types, and the pre- and postconditions and the effects of event types. For example, an object of the file type provides storage space for text, carries the attribute type 'name', and is specified as existing at a certain storage space or not at all.
b) The type hierarchy describes the hierarchical structure between objects, in which lower-level objects inherit the properties of the higher-level objects.
c) The spatial hierarchy describes the relations between things (generally objects) in terms of providing a location for each other, such as the display screen providing places for data fields which may contain strings, characters, etc.

2.2. The Dictionary of Basic Tasks

The dictionary of basic tasks connects the semantics of the task world, described in the UVM, to the commands the user may issue, and the related command parameters. The dictionary of basic tasks represents the functionality of a computer system. Basic Tasks are defined in terms of elementary functions the system provides. Basic tasks are not defined in terms of what the user perceives as single commands: "unit tasks" in the GOMS model (Card, Moran and Newell; 1983) or "simple tasks" in Task-Action Grammar (Payne and Green; 1986).

2.3. The Production Rules

The production rules of an ETAG description describe the grammar of the computer system's task language by means of a feature grammar, borrowed from Task-Action Grammar (Payne and Green, 1986). Instead of TAG's arbitrary levels of abstraction, ETAG makes use of a well chosen hierarchy of four levels of abstraction, loosely borrowed from Moran's (1981) Command Language Grammar: (1) the specification level, (2) the reference level, (3) the lexical level, and (4) the keystroke level.
Whereas the UVM describes the "how it works" part of the knowledge of a system, and the dictionary of basic tasks "what can be done", the production rules describe what Kieras and Polson (1985) call the "how to do it" part of the knowledge. ETAG uses these four levels because they correspond to the main levels of abstraction in interaction, and to make the choice between design alternatives (eg. style of interaction, syntax, naming, physical actions) at each main point of specification explicit.

The specification level describes the order in which the event and the object or objects are to be specified to the computer system. For example, the choice between an object-action or an action-object order of specifying command elements.
The production rules at the reference level specify how the action and the object(s) are referred to, and more generally, the style of interaction. This level allows for the choice between for example, Direct Manipulation interfaces, which generally use pointing, and command line interfaces which use naming.
The production rules at the lexical level specify how a command element should be labelled, or how to translate it as a semantic concept into a verbal or visual equivalent. For example, the classical studies of command naming and abbreviation can be placed at this level.
The lowest, keystroke level of production rules in ETAG describe how the names and labels for command

elements are to be specified as physical actions, such as keystrokes, mouse actions, or perhaps speech acts.

3. Etag as the Basis for Help Systems

In designing a help system for a computer system there are three main points of concern: (a) ensure that the user is supplied with the help that is needed in the particular circumstances; (b) ensure that the help conforms to the characteristics and the behaviour of the system, and, (c) ensure that there are no internal inconsistencies in the help information.
The question of how to ensure that the information provided is appropriate can not really be answered by ETAG. We have restricted ourselves to user-initiated help requests. What needs to be done is to define the set of possible or 'allowable' user questions and determine whether the answers to them are appropriate. The formality of ETAG will facilitate to do this, at least in comparison to using a non-formal system representation.

Earlier, it was argued that a formal model of the knowledge of a competent user, like ETAG, should be used as the basis for the design of computer systems. There are good reasons to use the same formal model for designing help information for such systems: when the system and the help facilities share the same source, their mutual integrity is guaranteed: the help will automatically keep up with (changes in) the system. A uniform formal basis for the help information will also ensure the internal integrity of help information. ETAG can in principle be used as the basis for help, and especially in comparison to using non-formal representations it will be advantageous to do so.

3.1. The general structure of an ETAG help system

A general structure of ETAG based help systems is developed by van der Veer (1990). The main ideas behind this structure are to insure that the help system is independent of the target system and follows a modular structure. Target system independence is meant to insure a wide applicability of help systems, including usage for research and prototyping purposes. Apart from usual considerations, modularity is important for practical reasons (the extent of a student project) and to enable construction of special purpose systems for, eg. user help, tutoring, etc. using read-made building blocks. The role of an (eg. ETAG-based) help system within a general architecture for modular user interfaces is presented in figure 1. Part of any (static or dynamic) ETAG based help system is a natural language interface, to translate between the user's natural language and the formal language used internally by the help system. The Natural Language Back End (NL-BE) translates formal information into more or less natural English. Similarly, a Natural Language Front End (NL-FE) is used to formalize questions of the user. The NL-FE receives questions from the doorkeeper machine, which distinguishes help requests from commands directed to the system. A Formal Question Answering Machine is the module which actually retrieves the requested information. A module to translate an ETAG representation into a suitable format for question answering is called the ETAG Translator.
In help systems, static information is the information about a computer system which does not change by interacting with it, such as knowing what the concept file means. On the contrary, dynamic information may change during interaction, such as knowing if a particular file is still there. To provide dynamic information several modules are required, in addition to those of a static help system. In ETAG-based help systems, independence of the target system is reached by simulating the target system in terms of ETAG's basic tasks in an ETAG Simulator module.
During a session, the actions of a user are collected in a User Log file. This file is parsed by a Log Parser module to determine which basic task and task arguments the user has -correctly- specified. Basic tasks change the state of the system, which is kept by the System State Manager. The System State Manager communicates with the Database Manager to store and retrieve facts about the interaction history in the History Database. The real or target system could be another part of the history system. However, to ensure the application independence of the help system, it was decided to simulate the computer system in terms of ETAG's Basic Tasks, by means of a module,
The general structure just described may be extended with additional modules to make it suitable for different purposes. For example, to build intelligent teaching systems, modules might be added, like user monitors, process inference modules and coaching machines (see: van der Veer, 1990 for further details).

4. Examples of ETAG-based Help Systems

In this section we will report about research efforts involving the use of ETAG representations as the basis for systems to provide users with various kinds of help. The information about computer systems was either generated solemnly on the basis of ETAG, or generated using ETAG in combination with additional sources. Information was provided by means of non-interactive on-line manuals or by means of interactive help systems. In the latter case, the system could either provide static information about the system, or dynamic information about interacting with the system. The research reported below took place under our supervision and was based

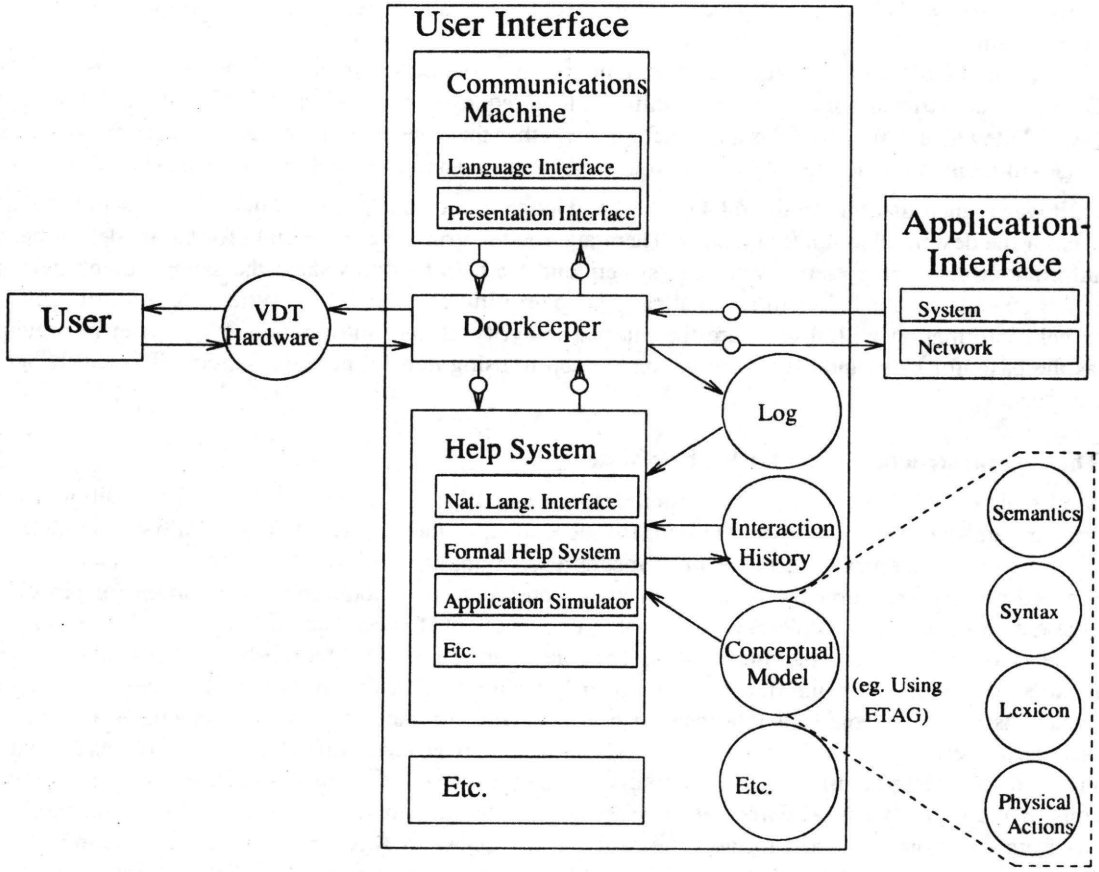


Fig. 1. The Help System within a general architecture of user interfaces (Van der Veer, 1990).

on the ideas of the authors. The actual work and the implementation of the systems involved was done by master thesis students to whom we would like to express our gratitude.

4.1. Help Systems to provide On-line Manuals

A first project on ETAG based help systems, reported in Broos (1989), involved creating a system to automatically generate natural language help text on the basis of a description of the UVM of a computer system. The system is a non-interactive Natural Language Back End. A Natural Language Front End was outside the scope of the project. Instead, a limited set of questions is assumed to be specified by means of, for instance, a simple menu interface listing all possible questions. Task level help is not provided, but semantic ("what is ... ?"), syntactic ("how do I ... a task ?"), and keystroke level ("how do I ... an action ?") is. Figure 2 graphically presents the structure of the translator module. The following example presents a clipboard as it would appear in the type specification part of an ETAG representation of a typical macintosh application.

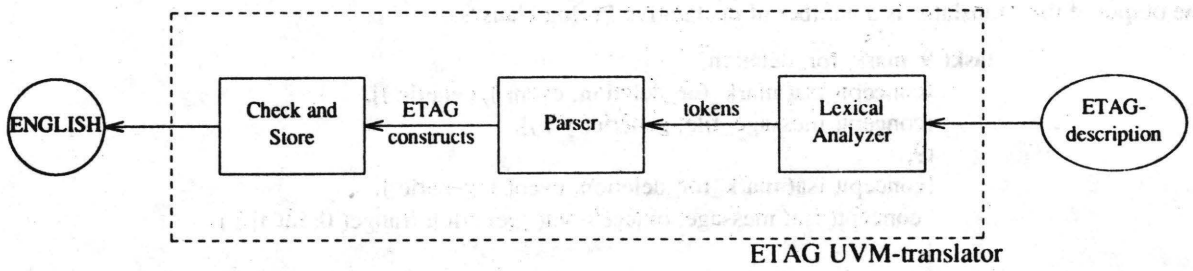


Fig. 2. The ETAG UVM-translator

```
type [OBJECT = CLIPBOARD]
     supertype:    [SINGLE_OBJECT_BOARD] ;
     themes:       [STRING: *s] | [WORD: *w] | [PICTURE: *p] | [RULER: *r] ;
     instances:    [CLIPBOARD: #clipboard] ;
end [CLIPBOARD]
```

In essence, this specification indicates that there is one clipboard, called "clipboard", which may contain either a string, a word, etc. The help system would translate this specification into something readable like the following:

> Here is some information about the type CLIPBOARD
> It is a special kind of SINGLE OBJECT BOARD.
> It can contain a STRING, a WORD, a RULER, or a PICTURE.
> There is a CLIPBOARD, indicated with "clipboard".

The help system is implemented using the lexical analyzer Lex (Lesk, 1975), and the specification language for user interface management systems SYNICS (Guest, 1982). Semantic, or rather, strategic information about task procedures is not supported. Also, the natural language output of the system is readable but not very 'natural'. This project showed that ETAG-based help systems are feasible. Furthermore, the help provided may easily be extended or changed by using different ETAG representations.

A practical application of creating an on-line manual involved the design and implementation of a help function for an experimental electronic mail system, based on an ETAG description of it (Tamboer, 1991). The electronic mail system was developed as part of a European research project of the COST-11-ter working group (see: van der Veer et.al., 1988). Due to some radical changes of the ETAG notation, it was no longer possible to automatically translate ETAG information into help messages so it had to be done by hand. Nevertheless, this project showed that it is possible to use help messages created on the basis of an ETAG description to build a help function as an integral part of a computer application.

4.2. Help Systems to provide Static Information

The goal of this project was to investigate possibilities to create an interactive help machine, as a step further then facilities to translate information, formally represented in ETAG, into a kind of natural language. Part of this project involved building a help system around a formal help machine (Fokke, 1990). An automatic ETAG Translator transforms a given ETAG description into a Prolog database of clauses or 'facts' representing the static information contained in the ETAG model. This part involved choosing an appropriate representation format for the database. As an example, in some electronic mail systems there is a task "mark for deletion" to discard one or more messages upon leaving the mail environment. In the Dictionary of Basic Tasks of an e-mail system it can be described as shown below. The ETAG description tells that this task is identified by the arbitrary number 9, and that it invokes an associated basic event with the same name. Upon execution of the task, the system will supply the name of the so-called message file, whereas the user should specify the name of the task and the messages to be marked for deletion:

```
ENTRY 9:
[MARK_FOR_DELETION isa TASK],
[MARK_FOR_DELETION isa EVENT],
[MESSAGE_FILE: *y],
T9 [MARK_FOR_DELETION isa EVENT] [MESSAGE isa OBJECT: {*}]
```

The output of the Translator is a number of declarative Prolog clauses:

```
task( 9, mark_for_deletion,
      [concept( isa( mark_for_deletion, event ), generic )],
      [concept( message_file, generic( y ) )],
      t9,
      [concept( isa( mark_for_deletion, event ), generic ),
       concept( isa( message, object ), set( [generic], [range( 0, inf )] ) )
      ],
).
```

Using a specially designed formal query language, an Interpreter can consult the database to generate a formal answer to a formally specified question. Figure 3 presents the architecture of such a formal help machine.

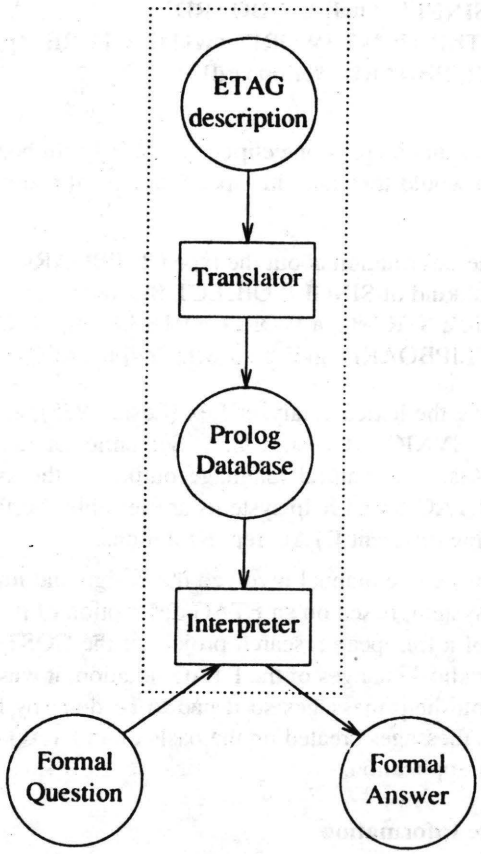


Fig. 3. the Architecture of the Formal Help Machine (Fokke, 1990)

In addition to using the database to answer questions, it may also be used for different purposes, such as checking the consistency of the ETAG representation. The translator is a Prolog program generator which is the result from applying Lex (Lesk 1975) and the compiler-compiler Yacc (Johnson, 1975) to a Backus-Nauer Form (BNF) representation of the ETAG notation, and compiling the resulting C program. To capture changes in the ETAG notation, and consequently its BNF representation only requires one to re-generate the Translator program.

Recently, a simple but ingenious natural language front end has been designed and implemented by Go (1992). It is a simple tool, because it restricts the user in asking questions. However, it employs an intelligent mechanism to find out what the user's questions are about. Interaction is simplified by presenting the user with predefined menu-choices to select particular types of questions, like asking for the procedure to perform a task or explaining a concept.

How can I	<task keystrokes or syntax>
What happens if I	<task effect>
Does the system ..... if I	<task event>
What is	<concept explanation>

After specifying the type of question, the user still has considerable freedom to ask questions. Using standard language processing procedures (lexical and other analyses, a general purpose lexicon, syntax rules, etc.), a number of parse trees is created, representing alternative interpretations of what the user might have meant to ask. To select one of the interpretations as best, the system makes use of a encyclopedia which contains knowledge rules about how ETAG concepts are referred to in the formal Prolog representation. From among the alternative questions, one is selected for further formalization that comes closest to being answerable, or "unification", in Prolog terms. A minor drawback of this project is, that changing the (kind of) ETAG representation requires some effort to build a new encyclopedia. Given that there is an ETAG representation and a general purpose lexicon, the rest of the process is automatic. Figure 4 presents the architecture of the Natural Language front-end of the static help machine.

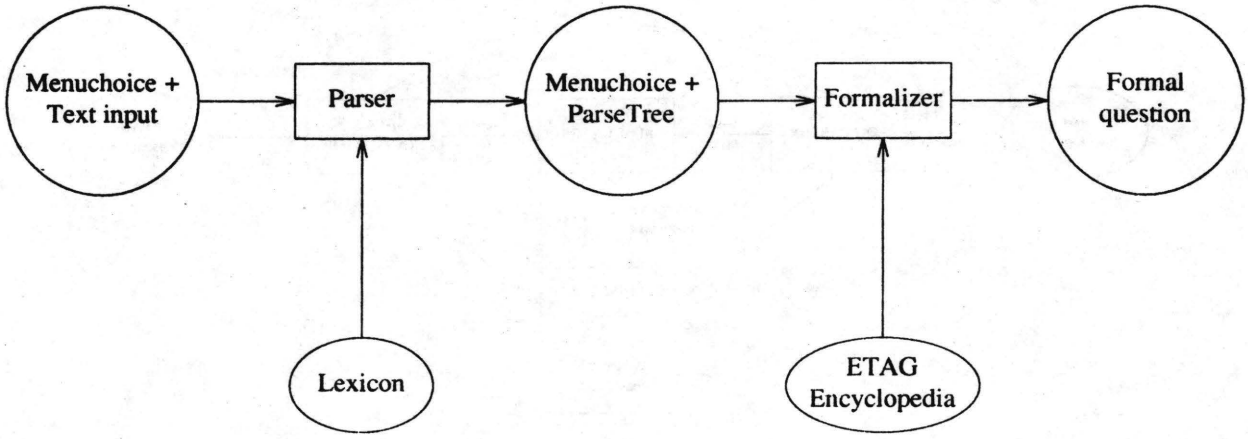


Fig. 4. the Natural Language Front-End (Go, 1992)

A practical example of how to use ETAG in combination with a Prolog database comes from a project by Smit (1991). Instead of directly translating from ETAG to natural language, a Natural Language Back End (NL-BE) to the database was built, which could produce a manual of the computer system in readable but rather imperfect English. Using an intermediate Prolog representation does not change the actual content or quality of the output of the help system. As such, the example presented in chapter 4.1 still valid. Provided that ETAG representations exist, manuals for different computer systems can be generated by running the ETAG translator and the Natural Language Back End.

Both of these projects show that it is feasible to use ETAG to create interactive help systems for static help systems. In addition, using the Prolog database as an intermediate representation creates opportunities to use the information which is statically represented in ETAG in a more active way, by manipulating it. Example of this are: consistency checking of ETAG representations, and adding history information to the database in order to provide dynamic information about interacting with a system.

4.3. A Help System to provide Dynamic Information based on an ETAG Simulator

An extension of using an intermediate representation of ETAG information to provide static help information is to construct a help machine which also stores information about the interaction. Such information can be derived by asking the system being used to provide it. However, this would limit the flexibility of the target and help system as a whole, because the help system would need to know what is happening within the target system, and the target system should be adapted to supply this information. In addition, it would require that the target system should actually be used in order to work with the help system.

To circumvent such limitations, and to allow the help and target systems to be used independently (including simultaneously), it was decided to simulate how the "virtual" target system should work on the basis of only the ETAG description of it. In principle, the user will not be able to differentiate between using the target system, the help system, or both of them simultaneously, because the system responses will be exactly similar. Except, of course, that in the first case, no help will be available. Similarly, because the its input will be the same, the target system will not know the difference, although a help system might have grabbed all available input for its own processing, because the help system will pass on all command input that was originally directed to the target system.

To build an ETAG-based help system for dynamic information, a History System is needed, which simulates the target system using an ETAG description of it, a log of user activities and a History Database of system states. A first project (Bakker and van Wetering, 1991) involved the design and implementation of the main software modules of the history system, and (the format of) the History Database. The format of the previously used Prolog database had to be extended in order to store 'facts' about past system states, starting from the so-called Initial State. The Initial State is the state the system is in when the application environment is entered for the first time. Whenever a user tries to execute a basic task, simulated basic events will happen. These event occurrences serve as matching points in the History Database, in whose terms facts about the interaction are stored, and in whose terms questions of the user all have to be reformulated. The structure of the history system is presented in figure 5.

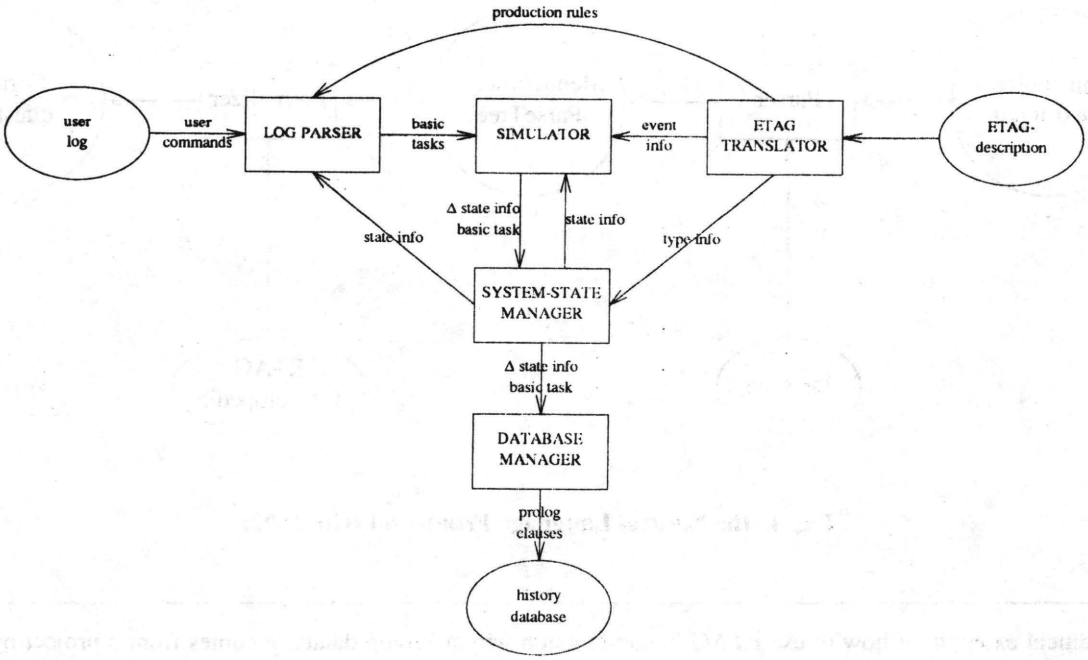


Fig. 5. the Structure of the History System (Bakker and van Wetering, 1991)

The kernel of the history system is the System State Manager, which keeps track of, and provides information about the system state, and which passes state information to the Database Manager for storage purposes. The activities in the User's Log are parsed by a Log Parser to identify basic task and their arguments, guided by information from the History Database. Information about basic tasks is passed to the ETAG Simulator, where they are simulated as basic events. To simulate basic events, state information is provided by the State Manager, which also receives information about the system state (changes) to have it stored in the database by the Database Manager. Finally, information about the content and requirements of the basic tasks is provided by the

ETAG Translator.
An interesting aspect of the project is the following. Earlier it was noted that the ETAG Translator could be used outside the context of help systems, to check the syntax and various semantic aspects of the ETAG representation. Here, in a similar vein, parts of the History System, and particularly, the ETAG Simulator can be used in a wider context, to inspect time-dependent aspects of computer systems, and even more promising, for prototyping purposes. The opportunities to use these modules for quite different purposes than what they were meant seems to justify the modularity of the architecture.

Kohli and Woudstra (1992) designed and implemented the Log Parser. This module has a threefold function. The first function is to parse the user's actions to identify basic task invocations, on the basis of information about the production rule part of an ETAG representation, which is provided by the ETAG translator. The second function of the Log Parser is to identify the type and number of concepts (tasks, objects, places, etc.) in the user's task specifications, and to check if they are in accordance with the type information in the ETAG representation. The third function is to identify the existence of the type instances (files, folders, etc.) themselves, on the basis of information from the System State Manager and the History Database.

Presently, two more projects concerning dynamic ETAG based help systems are going on, both focussing on answering user questions. In one (Schep, 1992) work is done on the formal parts of a question answering machine and, more specifically, the design of a FQL, a formal query language to interrogate the history database. The starting point of this project is an analysis of which kinds of questions involving dynamic information may logically be answered on the basis of the information in the History Database. Eventually, this system will be able to provide dynamic information to (formal equivalents of) questions such as: "when did the file MyFile exist?" or "which files are currently owned by MyName?".
To give an impression of the structure of the Dynamic help system, figure 6 presents the structure of its main modules and the relations between them.

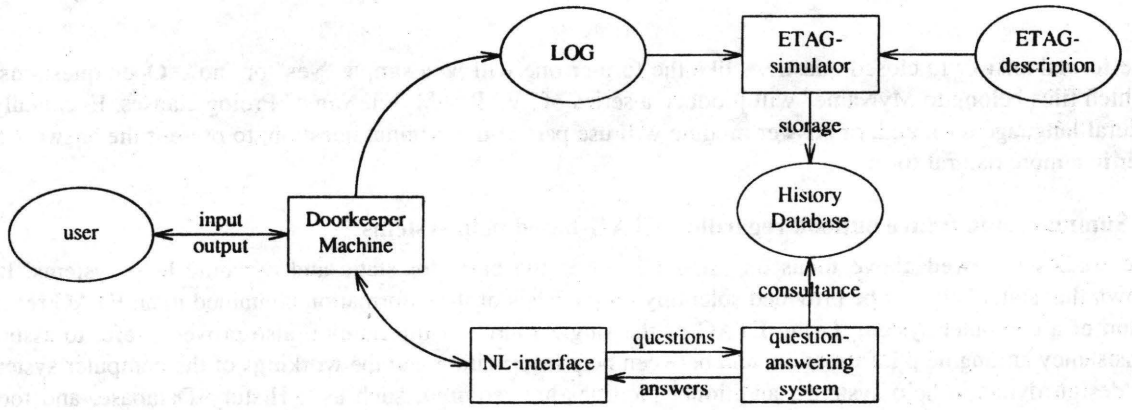


Fig. 6. the Structure of the Dynamic Help-system

Directly related to the formal question answering system two modules are required which address the translation of information between natural language and formal query language: the question module and the answer module. The purpose of the other project (van Hoof (1992) is to deliver a question module asking dynamic information. In general terms, this would involve a lexical, syntactic and semantic analysis of a question of a user, and presenting the formal result to the question answering machine. However, for practical purposes, the process is turned upside down.
Instead of building a huge natural language processing sub-system, and a system to determine whether the questions of the user can be answered indeed, it was considered more appropriate to build a question module asking the users to provide the necessary and sufficient information. In this way, after the user has indicated that help is needed, the question module will ask particular questions until enough information is collected to formulate a question that can be meaningfully answered by the formal question-answering machine.

Presently, the dynamic help system is not complete. It is not far from complete, but some of the features are missing, which are required to apply it for other purposes then demonstration and research. A main omission is the natural language back end. As such, only formal -Prolog- answers are produced. Furthermore, the help

systems for static and dynamic information live their lives independently. Both of them can actually be used, but integrated use is not possible yet.
To give an idea about the capabilities of the present dynamic help system, the following is an example of a part of the interaction between a user and the dynamic help system. In order to save (much) space and to show the main points, it has been extensively shortened, and somewhat simplified. Consider that a user wants to ask a question like:

"Is there a file called MyFile [now]?"

To answer this question, it has to be presented to the formal question-answering system in one of its question formats with the appropriate variable substitutions. In formal terms, the question can be rephrased as asking whether there is currently ("during now") a file called "MyFile" that existed "during now":

```
query( [state, exists, true,
    [[object, file, ref1, the,
        [[[state, has_val, true,
            [ref1, [attribute, name], [value, string, 'MyFile']]
            [context, ref2, the, point, [[[during, now]]]]
        ]]]
    ]],
    [context, ref3, the, point, [[[during, now]]]]
]).
```

The natural language front end, or rather, the question asking module is to build the Prolog clause above, by systematically asking questions, such as:

do you want to Check something, or to Solve a problem [C/S] ? :	C
do you want to know something about an Object or a Task [O/T] ? :	O
do you want to know about a Value, a State or a Place [V/S/P] ? :	S

The formal answer to closed questions like the former one will be a simple "yes" or "no". Open questions like: "which files belong to MyName" will produce a series of "VAR = MyFileName" Prolog clauses. Eventually, the natural language back end, or answer module will use part of the original question, to present the answers to the user in a more natural form.

5. Summary and future outlook regarding ETAG-based help systems

The studies reviewed above focus on using ETAG as the basis for static and dynamic help systems. It was shown that static help can be provided solemnly on the basis of the information contained in an ETAG representation of a computer system. Using ETAG as the single source of information also proved useful to assure the consistency among help information, and between help information and the workings of the computer system.
To design dynamic help systems, additional facilities are required, such as a History Database, and tools to manipulate that. Note that the advantage of using ETAG as the single source of information is not lost when introducing a database representation, as it only involves a direct and automatic translation from one formal representation to another, without any loss or addition.

The work reported above mainly involved implementing help systems. A typical and most important implementation problem encountered was how to deal with the -seemingly- ever changing ETAG notation, and build systems flexible enough to avoid having to rebuild modules. Another problem was that 'local' changes in software modules, motivated by the need to solve specific implementation problems in one project, did sometimes lead to problems elsewhere. With the increasing number of student projects and hence, software modules, the urge for something like system administration increases.
A first move towards flexibility was to use general purpose tools, such as Lex, Yacc and the C programming language, instead of a dedicated tools like SYNICS. Using these tools, changing the ETAG notation would in most cases only require having to create a new or changed BNF representation of ETAG, and a recompilation of the software.
A second decision to increase flexibility was the introduction of a kind of intermediate representation, the Prolog database to represent information apart from, and instead of, the ETAG model. This change was also required to allow dealing with dynamic help information.
Using a database and, in addition, Prolog's facilities to make logical inferences also provided the possibility to use the information in multiple ways. Not only to create help systems, but also, and this may prove important in the long range, to create tools for consistency checking and 'debugging' ETAG representations, and for using

ETAG as a language for prototyping purposes.

An important problem facing ETAG-based help systems is that they are not able to provide information which is not already present in either, the history of interaction or the ETAG representation. This problem concerns strategic help information. Tasks in ETAG are elementary tasks, which are presented to the user as indivisible units. Users however may also be interested in asking questions about tasks at a higher level, such as GOMS's unit tasks (Card, et.al, 1983) and TAG's (Payne and Green, 1986). These psychological unit tasks involve sequences of basic tasks and strategic decisions about how to combine them. Such decisions, however, fall outside the scope of ETAG, at least at present. This information has to added from a non-formal source, although it may be possible to use (parts of) the ETAG representation to do so in a systematic manner.

A final problem concerns the fact that ETAG representations do not seem to lend themselves easily for translation into a really natural English. It remains to be seen whether a quasi English message, like "the task CHOWN changed the attribute OWNER of the object YOURFILE of type file", which is directly translated from ETAG information is really useless. To create a more natural English would require far more powerful natural language modules then the ones which merely transform natural language sentences to and from their formal equivalents. As such, projects like this may be feasible only for the longer term.

At present, the modules of an ETAG based help system move towards a unified, but simultaneously flexible, system. What needs to be done in the near future is to implement the missing pieces, like parts of the doorkeeper machine, the natural language back-end, and perhaps a back-end to present help information by other means, such as, visually.

Finally, and for the longer term, it may also be worthwhile to consider extending the ETAG Simulator towards a real prototyping environment, and to create tools for analyzing ETAG representations and/or interaction histories as a first step towards building systems in which users do not need extensive help facilities.

5.1. Conclusions

In this paper, it was argued that formal representation techniques can and should be used as the basis for intelligent help systems. We think that the examples provided justify this argument. The examples also provide evidence supporting our second argument: that is possible to use a specific formal model, an ETAG representation of the competent user's knowledge of a computer system as the basis for help systems, and moreover, as the single source of information. ETAG was originally only meant to be a specification method for computer systems, and not meant for the purpose of creating help systems. However, the information contained in an ETAG description showed to be useful and sufficient to create help systems to provide all kinds of help. Strategic information cannot be provided by a model for mere descriptive purposes. But information can be provided about both, the computer system as well as the interaction with it. Elsewhere, it was shown that ETAG can be used fruitfully to specify computer systems in various application areas (de Haan and van der Veer, 1992). In this paper, it was shown that ETAG can also be used successfully for a quite different purpose. Thus, increasing the scope of its applicability and showing the validity of the approach. Finally, we argued that a modular structure of help systems would be advantageous. Regarding this point, it was shown that modules, modules which were developed as elements of help systems can be useful for quite different purposes. Finally, the help systems, created on the basis of ETAG are strictly application independent. To provide help information about whatever system, only requires an ETAG description of the particular system..

6. Acknowledgements

We would like to thank all of our students: Diderik Broos, Fred Bakker, Michael van Wetering, Mark Fokke, Michel Schep, Robert Smit, Erik Tamboer, Martin van Hoof, Willem Woudstra, and Vikas Kohli, who did the actual work of designing and implementing the modules of the help system. Without them, the systems would not exist, and ETAG would still be in its infancy. We also would like to thank Hans van Vliet for supervising these projects, and for the critical and clarifying comments he made about this paper.

7. References

Bakker, G.W.A & van Wetering, M.W. (1991). A Dynamic Help-system Based on the Execution of an ETAG-description. M.Sc. Thesis, Dept. of Computer Science, Free University, Amsterdam.

Broos, D. (1989). ETAG-based help: Generating Human readable Explanations from a Formal Description of the User Interface. M.Sc. Thesis, Dept. of Computer Science, Free University, Amsterdam.

Fokke, M.J. (1990). An interactive Question-Answering System based on a Formal Description of a User Interface. M.Sc. Thesis, Dept. of Computer Science, Free University, Amsterdam.

Go, N.T. (1992) De Natural Language front-end van een op ETAG gebaseerde Helpmachine. [The Natural Language front-end of an ETAG-based Helpmachine] M.Sc. Thesis in preparation, Dept. of Computer Science, Free University, Amsterdam.

Schep, M.S. (1992) Question Answering System based on the ETAG History Database. M.Sc. Thesis in preparation, Dept. of Computer Science, Free University, Amsterdam.

Smit, R.G. (1991) On the translating of ETAG into English. M.Sc. Thesis, Dept. of Computer Science, Free University, Amsterdam.

Tamboer, E. (1991) An ETAG-based Help-module for PMAIL. M.Sc. Thesis, Dept. of Computer Science, Free University, Amsterdam.

van Hoof, M.P.J. (1992). A Question Answering System based on ETAG: the question input module. M.Sc. Thesis in preparation, Dept. of Computer Science, Free University, Amsterdam.

Woudstra, W.M. & Kohli, V (1992). A Frond-end for the Simulator of a User Interface, using the Formal Description Language ETAG. M.Sc. Thesis, Dept. of Computer Science, Free University, Amsterdam.

Card, S.K., Moran, T.P. and Newell, A. (1983). The Psychology of Human-Computer Interaction, Lawrence Erlbaum Ass., Hillsdale, New Jersey.

De Haan, G. and Van der Veer, G.C. (1992). Analyzing User Interfaces: ETAG validation studies. Proc. of the 11th workshop on Informatics and Psychology: Task-Analysis in Human-Computer Interaction, Schaerding, Austria, 9-11 June, 1992.

De Haan, G., Van der Veer, G.C., and Van Vliet, J.C. (1991) Formal Modelling Techniques in Human-Computer Interaction. Acta Psychologica, 78, 26-76.

Guest, S.P. (1982). Software Tools for Dialogue Design. Int. Journal of Man-Machine Studies 14, 263-285.

Johnson, S.C. (1975). YACC: Yet Another Compiler-Compiler. Computer Science Technical Report No. 32. Bell Telephone Laboratories Inc., Murray Hill, New Jersey 07974.

Kieras, D. & Polson, P.G. (1985). An Approach to the Formal Analysis of User Complexity. Int. Journal of Man-Machine Studies 22, 365-394.

Lesk, M.E. (1975). Lex - A Lexical Analyzer Generator. Computer Science Technical Report No. 39. Bell Telephone Laboratories Inc., Murray Hill, New Jersey 07974.

Moran, T.P. (1981). The Command Language Grammar: A Representation for the User-Interface of Interactive Systems. Int. Journal of Man-Machine Studies, 15(1), 3-50.

Norman, D.A. (1983). Some Observations on Mental Models. In: Gentner, D. & Stevens, A.L. (eds.) Mental Models, 7-14. Lawrence Erlbaum Ass., Hillsdale, New Jersey.

Payne, S.J. & Green, T.R.G. (1986). Task-Action Grammars: A Model of the Mental Representation of Task Languages, Human-Computer Interaction vol. 2, 93-133.

Tauber, M.J. (1988). On Mental Models and the User Interface. In: van der Veer, G.C., Green, T.R.G., Hoc, J.M. and Murray, D.M. (eds.) Working with Computers: theory versus outcome. Academic Press, London. 89-119.

Tauber, M.J. (1990). ETAG: Extended Task Action Grammar - a language for the description of the user's task language. In: Diaper, D., Gilmore, D., Cockton, G., and Shackel, B. (eds.). Proceedings Interact'90, pp. 163-168. Elseviers, North-Holland, Amsterdam.

Van der Veer, G.C., Guest, S., Haselager, P., Innocent, P., McDaid, E., Oestreicher, L., Tauber, M., Vos, U. & Waern, Y. (1988). An interdisciplinary approach to Human Factors in Telematic Systems. Computer Networks

and ISDN Systems 15, 73-80.

Van der Veer, G.C. (1990). Human-Computer Interaction: Learning, Individual Differences, and Design Recommendations. Ph.D. Thesis, Free University, Amsterdam.

Integration of Computer Tools Into the Structure of Human Activity: Implications for Cognitive Ergonomics

Victor Kaptelinin

Institute of General and Educational Psychology,
Russian Academy of Education,
Moscow, Russia

ABSTRACT

The psychological problems of human computer interaction are discussed in the present paper from the point of view of Russian activity theory. The computer is considered as a tool mediating human activity. The subject matter of the psychology of human computer interaction is defined as the integration of computer tools into the structure of activity. The paper begins with a discussion of the reasons for the recent interest in activity theory. Then it introduces the basic principles of activity theory by contrasting them with those of cognitive psychology. The notion of a computer tool as an extension of "internal plane of actions" is proposed and applied to the analysis of WIMP interfaces. A special section deals with the role of mental models in the process of the acquisition of system handling skills. The perspectives for applying activity theory to developmental aspects of human computer interaction are discussed in the last section. It is concluded that cooperation between cognitive ergonomics and activity theory can be beneficial for both of these approaches.

1. Introduction

In recent years there has been growing interest in higher level factors of human computer interaction (see Grudin, 1990). It is becoming more and more evident that the usability of computer systems is determined by their compliance not only with principles of human information processing but also with user's goals and strategies, social organization, and individual differences in education and development. The subtitle of the present conference can be considered as a further indication of this tendency.

The extension of the subject matter of cognitive ergonomics requires new methods and concepts, ones which will be adequate for solving new kinds of problems. At present such conceptual tools are actively looked for, either within the framework of the cognitive paradigm or beyond this tradition. Russian activity theory, the leading theoretical approach in what was formerly Soviet psychology, is considered by a number of specialists as one of the most promising alternatives to cognitive theories (Bødker, 1991; Grudin, 1991; Raeithel, 1992). The prospects for applying activity theory to the field of human computer interaction will be discussed in the present paper.

There are several versions of activity theory. In the present paper we mean by "activity theory" the approach developed by A.N.Leontiev (1959, 1975), on the basis of the cultural - historical conception of L.S. Vygotsky (1956).

The implicit assumption underlying most attempts to use activity theory is that two fields of application -- that of activity theory and that of the cognitive approach -- are separated. Cognitive analysis is supposed to be appropriate for studying mechanisms underlying individual human computer interaction, while activity theory is presumed to provide a framework for investigations of the impact of social, cultural, and organizational factors. From our point of view, the potential of activity theory is not limited to the contextual factors only. That is why special attention will be paid in the present paper to the

possibility of applying activity theory to individual human computer interaction. Activity theory was successfully used in the field of traditional ergonomics (Munipov, 1983; Zinchenko, Gordon, 1976; Zinchenko, Munipov, 1979) and it is reasonable to assume that it can also serve as a theoretical basis for modern studies in psychology of human computer interaction.

2. Activity theory vs. the cognitive approach.

Of course, it is not possible to give a full picture of activity theory within the space limits of the present paper. For our current purposes there is a good reason to introduce this theory by comparing it with cognitive psychology.

The subject matter of cognitive psychology can be defined as human information processing taking place between sensory input and motor output. The main task of cognitive studies is to build up formal models of this information processing. Activity theory makes quite different basic assumption about the nature of human mind. According to this assumption, the human mind emerges as a special "organ", which helps to survive a human being in the objective reality. So, the nature of the human mind can be described and understood adequately only within the context of subject's interaction with the outer world.

The basic unit of this interaction is activity driven by a motive. The motive is an object, material or ideal, which is of value in itself, i.e. an object fulfilling some human need. The human mind is considered as a special -- internal -- kind of activity.

It should be noted that there were several attempts to incorporate the notion of active nature of human cognition into the cognitive approach (see, e.g., Neisser, 1981; Norman, 1988). In these models the traditional "input - output" scheme is substituted by (or, rather, is complemented with) the opposite one, i.e. "output (human action) - input (feedback)". On the face of it, the adoption of this scheme seems to be quite enough for studying human activity while still being within the cognitive tradition. However, this scheme raises a number of rather hard problems.

To understand the nature of human actions, which are not completely determined by sensory inputs, well elaborated concepts of motivation and goal setting are needed. Then, the "output - input" scheme implies the existence of an intermediate link, namely the outer world changed by human actions. The inclusion of the environment into the scheme requires, in particular, an explicit conceptual description of the mechanisms underlying the influence of social factors on the human mind.

From our point of view, modern cognitive psychology lacks adequate conceptual tools for solving these problems (see also Velichkovsky, Zinchenko, 1979). The most radical attempt -- at least, according to the literature which is available to us -- to extend the scope of cognitive analysis and to apply it to human beings in their social environment is the concept of "knowledge in the world" introduced by D. Norman (1988). This approach provides very useful insights into the nature of human cognition. However, such fundamental aspects of human interaction with the outer world as motivation or the social determination of human mind are essentially not covered by this approach, yet these aspects have been profoundly elaborated within the tradition of activity theory.

According to activity theory, the influence of social factors on mental processes is not direct. These factors first determine the nature of activity, in two ways: (1) by the involvement of an individual into a shared (collective) activity and (2) by the usage by the individual of socially developed tools and means. Restructuring of external activity leads to restructuring of mental processes (internal activity) through internalization. In the process of internalization socially distributed (intersubjective) mental functions are transformed into internal (intrasubjective) ones (Vygotsky, 1956).

One of the major advantages of activity theory is a conceptual framework providing a coherent description of psychological phenomena at various levels, from interpersonal relations and motivation to perceptual and motor skills. This is allowed by the three level model of activity, worked out by A.N.Leontiev (1959, 1975).

This model distinguishes between activity itself and its subcomponents -- actions and operations (see Figure 1). Activity is composed of actions which are driven by goals. The difference between motives and goals is the difference between objects that impel human activity and objects that activity is directed at. Actions are, in turn, composed of operations which are determined by actual conditions. Usually subjects are not aware of their operations.

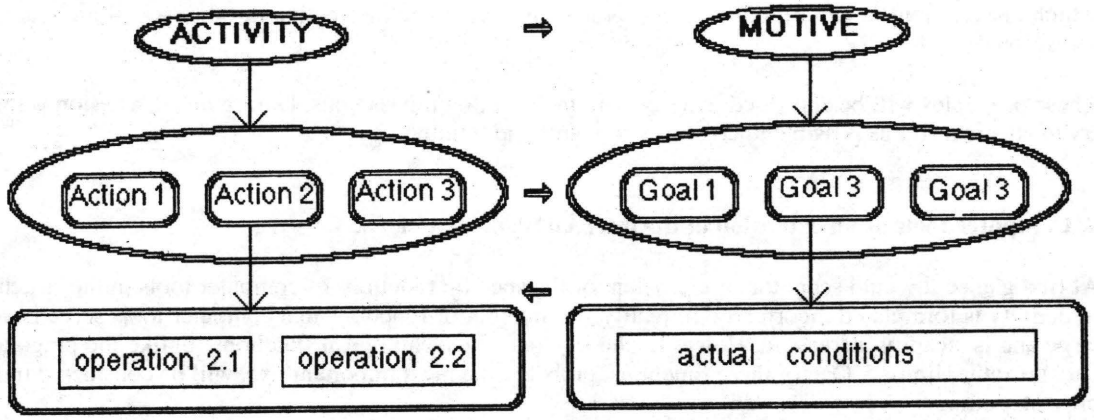


Figure 1. The structure of human activity put forward by A.N.Leontiev (1959, 1975).

The structure of activity is not invariable. Its components can be at different levels at different points in time. An action can metamorphose into an activity if its goal would begin to take on the properties of an object which is impelling in itself (the phenomenon of "shift of a motive to a goal"). Actions can also pass into operations through the process of automatization. When operations are not adequate to current conditions, they can turn to actions.

The hierarchical structure of activity just described makes it possible to analyze both low level and higher level mental processes within the general context of human interaction with his or her environment.

3. Human computer interaction as computer mediated activity.

Now we can formulate some points which are of central importance from the activity theory perspective but have not received enough attention, from our point of view, within the traditional cognitive approach. All these points emerge from the interpretation of computer as a tool which should be integrated into the structure of human activity in the most efficient way.

First, the specific psychological functions of computer tools in the structure of activity should be identified. It is necessary to understand the unique nature of computer tools compared to other kinds of artefacts in order to reveal the mechanisms underlying the integration of these tools into the structure of activity.

Second, the functionally equivalent processes of human computer interaction can actually be realized at different levels of activity structure. The psychological nature of these processes (i.e., whether they are activities, actions, or operations) should be taken into account because it can influence design decision or selection of an appropriate training strategy.

Third, it is necessary to analyze human computer interaction within the developmental context. Acquisition of new skills and abilities (the actualgenesis) establishes an important "novice - expert" dimension. The larger scale developmental processes (the ontogenesis) are to be considered in studying such problems as age differences in computer use or the impact of computer experience on cognitive development.

Fourth, human computer interaction is greatly influenced by social and cultural factors. Computer tools which are efficient in certain cultures and organizations can be ineffective in other social and cultural conditions.

These principles will be discussed more closely in the following sections. During this discussion we will try to adhere as far as possible to cognitive concepts and terminology.

4. Computer tools as an extension of the internal plane of actions.

At first glance it would seem that the problem of the specific functions of computer tools in the structure of activity is formulated incorrectly. In reality, the number of functions that computer tools serve is very large and is steadily increasing. However, the number of psychological functions (unlike the pragmatic ones) is rather limited. One of these functions, probably the most important one, will be considered in the present section.

Integration of a tool into the structure of activity means empowering the subject, enhancing some of his/her "natural" abilities. New tools serve as components of new "functional organs" (Leontiev, 1959). In another words, such a tool provides an extension of a pre-existing structure, be it a motor one or a cognitive one. Concerning computer tools, our point is that these tools can be considered as extensions of "internal plane of actions" (Kaptelinin,, 1991). This expression needs some clarification.

The "internal plane of actions" (or IPA) is a concept which has been actively used in Russian psychology recently (Ponomarev, 1975). It refers to the cognitive structure whose function is to perform actions "in the mind" before their actual realization. A lot of empirical (mainly developmental) studies dealing with IPA - related problems were conducted by Russian researchers. It was found, for example, that the IPA develops mainly in the primary school age. The concept of the IPA is somewhat similar, but not identical, to the concept of an explicit mental model (Carrol, Olson, 1988). Oversimplifying, we can consider the IPA as a basic structure underlying the functioning of mental models.

The extension of the IPA is not the only psychological function of computers. They are also used, for example, as extensions of long term memory or as communication tools. However, the special feature of computers which distinguishes them from other cognitive tools is their ability to perform knowledge manipulation, and not just to support its representation.

Computer models (the objects of "computer reality") and mental models have much in common, being placed, in a sense, between the subject and his/her environment. Computer models, while belonging to objective reality, are intended for human use. Mental models, while being a part of human mind, represent the structure and rules of outer world. Both of these kinds of models provide an opportunity for the subject to see the results of complex and reversible manipulations with modelled objects before applying these actions to real objects (see Figure 2).

An extension of a mental structure can be provided in two ways: (1)by external support of the mental structure, i.e. by delegating to a tool only functional subcomponents of the structure, the delegation which makes the functioning of the structure more efficient, or (2)by simulation, i.e. by delegating to the tool all the functions of the mental structure. Therefore, there are two ways of efficient integration of computers into the structure of activity. First, to create and to use tools that support functional components of the IPA, namely, retrieval and selection of relevant information, building up explicit models, manipulation with these models, presentation of these models' states to subject, comparison between the current state and the goal state. Second, to design systems that will substitute for the IPA,

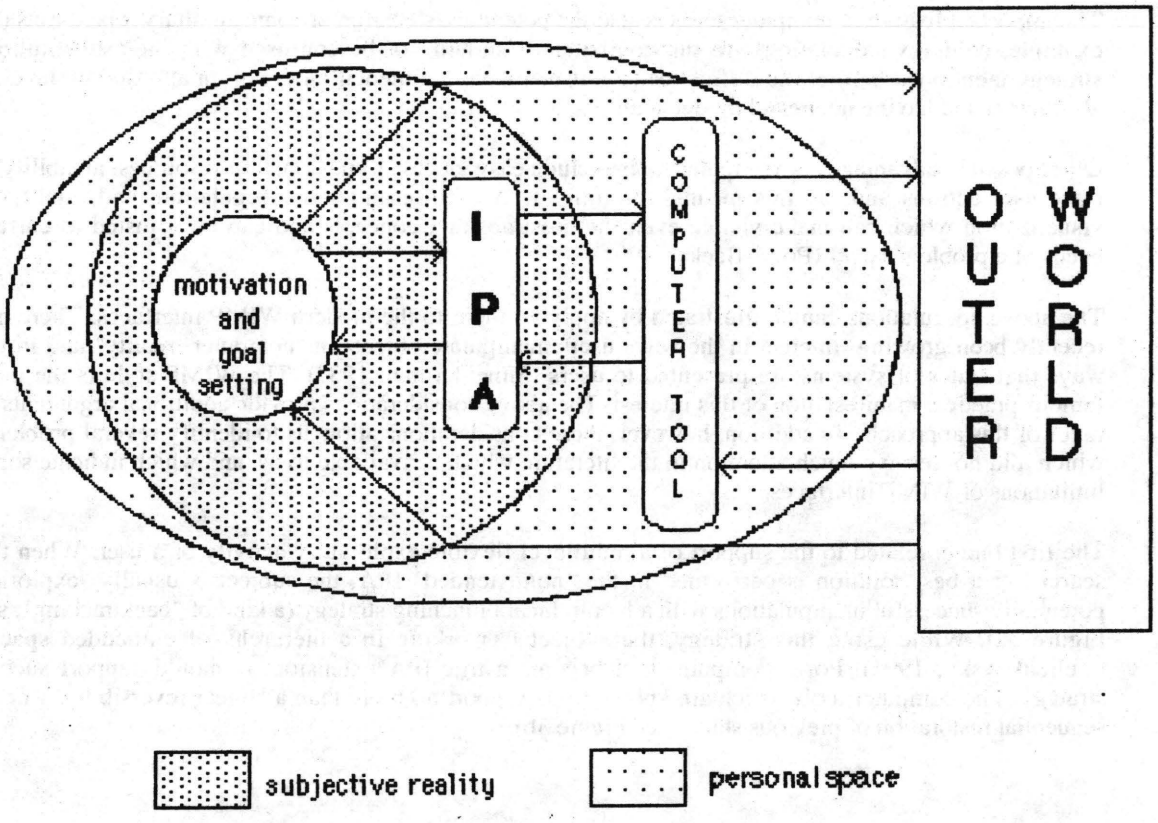


Figure 2. A computer tool as an extension of internal plane of actions.

i.e. systems which can be addressed with questions of the "what, if ...?" kind related to possible states of an object and can provide answers to such questions.

The potentiality of the second ("substitution") strategy seems to be rather limited. This strategy implies that the human being does not need to have a mental model of the system in question at all. In most cases, however, the user does need a mental model even if a tool's performance is perfect. Any use of a tool usually needs some preliminary knowledge. To avoid an ineffective trial and error strategy users have to decide what are the best questions to address to the system and whether it is necessary to use it at all. For example, the easy access to computers and calculators makes arithmetic skills rather insignificant, but if a subject has a kind of arithmetic intuition, he/she can not only correct obvious mistakes but also avoid calculations, if a preliminary evaluation of the result is enough.

The first ("support") strategy appears to be more promising. It is oriented to the optimal use of computer tool advantages for enhancing IPA potential. This brings up another point: What are the possible advantages of computer support of IPA? From our point of view, the following forms of this support can be identified. The most important one is providing the basis for knowledge accumulation. The user can add the necessary information when it is necessary; the flexible, "soft" nature of computer representations allows step by step development of an integrated body of knowledge necessary for solving the current problem. A computer based extension of the IPA is much less vulnerable to interference caused by distraction. It lessens user memory load and gives an opportunity for the user to focus comfortably on details, to try various perspectives for looking at the problem, etc.

The important feature of computer tools is also the potential delegation of some auxiliary operations (for example, complex calculations) to the computer. It should not be confused with the "substitution" strategy mentioned before: the difference is between having a mental model of an appropriate level of abstraction and having no mental model at all.

Other possible advantages of computer tools include easy access to information resources, an ability to trace user actions and, by this means, to stimulate reflection and self - regulation, and intelligent visualization which can make visible even the most abstract concepts and can be adjusted to current needs of a problem solver (Price, Baeker, 1991).

The above speculations can be illustrated by applying them to the modern WIMP interfaces. There has recently been growing interest in the ways users manipulate objects in "computer reality" and in the ways that states of systems are presented to users (Shneiderman, 1987). The WIMP style is the most famous practical manifestation of this interest. The above considerations provide additional arguments in favor of this approach. In addition, however, these considerations allow us to identify several problems which did not receive much attention in the literature which is available to us, and which indicate some limitations of WIMP interfaces.

The first one is related to the support of a multilevel flexible exploratory activity of a user. When the search for a best solution is performed in the "nonextended" IPA, the subject is usually exploring potentially successful manipulations with a hierarchical branching strategy (a kind of "backtracking", see Figure 3a). While using this strategy, the subject is working in a hierarchy of embedded spaces (Velichkovsky, 1991). For a computer tool become a true IPA extension, it should support such a strategy. The computer tools which are known to us support no more than a "linear reversibility", i.e. a sequential restoration of previous states (see Figure 3b).

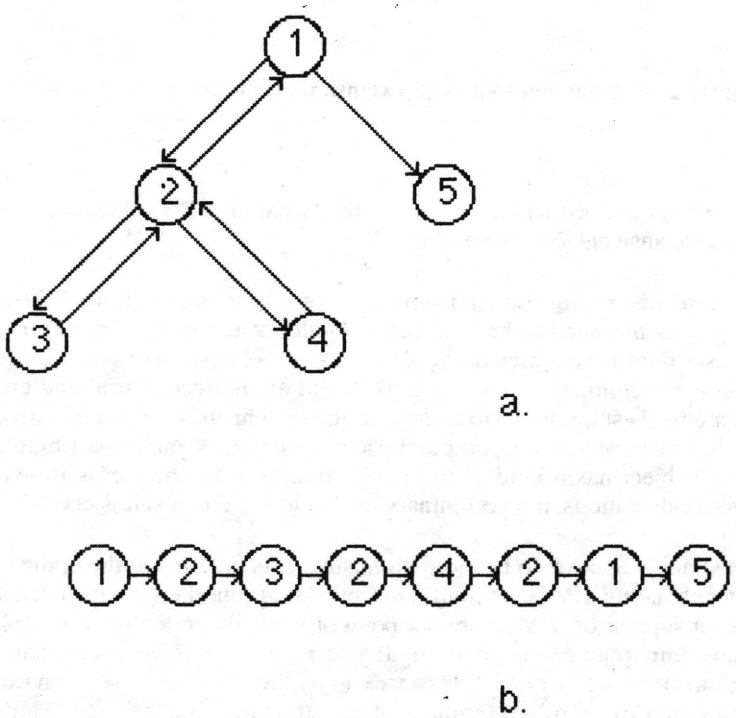


Figure 3. The difference between the multilevel exploratory strategy (a) and the linear reversibility provided by some computer tools (b).

The support of flexible nonlinear exploration strategies needs an appropriate functionality which can be provided by the modern hypertext and hypermedia systems. The main problem seems to be the design of appropriate user interface.

Another essential limitation of modern WIMP interfaces is the lack of "direct retrieval" of information which is potentially important but do not appear on the screen. Much of human cognitive ability stems from the very effective integration of central, peripheral, and unconscious processes. The desktop metaphor lacks this advantage: natural switching of attention from one piece (or source) of information to another is only possible if this information is presented on the screen, for example, as an icon. So the easy access to relevant information is limited to the use of perceptual features presented in the visual field. Therefore, the greater the amount of relevant information, the more distractors are in the field of attention, and the more complicated, confusing, and ill-organized is the working area.

Access to the information not presented on the screen is usually system - specific and far from natural. If a computer is intended to provide a general purpose working environment, these disadvantage is essential. I believe that one of the most promising lines of development in this field is the use of the ability of human beings to accumulate knowledge about natural (i.e. 3D) environments in the form of "images of the world" (Leontiev, 1979). The principles of "direct manipulation" (Shneiderman, 1987) and "transparency" (Rutkowski, 1982) should probably be complemented with the principle of "direct retrieval".

Third, the benefits of visibility and easy control provided by WIMP interfaces are coupled with the loss of some powerful features of "old fashioned" command language interfaces. An opportunity to operate on a class of objects and to integrate several applications in a configuration appropriate for a given task (batch files, pipes) is potentially useful not only for computer experts but for any expert user as well. We believe that there are very exiting opportunities to combine the benefits of direct manipulation interfaces with those of command language ones. For this purpose it is necessary to understand the natural ways of presenting to the user either abstract system objects or higher level procedural units relevant to the task structure.

5. Mental models, transparency, and acquisition of the system handling skills.

As mentioned in the section 3, one of the most fundamental principles of activity theory states that mental processes should be analyzed within the context of their development. The importance of this principle can be demonstrated by problems related to the mental models of interactive systems. There are various approaches to mental models (see van der Veer, 1990). In the present paper by mental models we mean explicit ones, which are different from simple rules and methods (Carrol, Olson, 1988).

Dialogue metaphor is, actually, misleading. Human computer interaction is not an interaction between two partners but rather a human interaction with the outer world "through the interface" (Bødker, 1991). So the role of mental models of interactive systems is rather uncertain.

It is generally accepted that mental models facilitate human computer interaction. This notion seems to be absolutely correct when the learning situation is considered, i.e. when the goal of the user is to understand and master the system. The real use of the system, however, implies a quite different structure of activity. Actions in the IPA are performed with objects from the task space; the computer use is taking place at the level of operations (which are usually not conscious), and an active use of mental models of a system can contradict the principle of user interface transparency.

We assume that two kinds of transparencies should be distinguished: the learner's transparency, based on mental models, and the expert's transparency, based on long term knowledge including both the procedural and the declarative (i.e. rules and "functional cognitive maps"). Mental models are important to the extent to which they facilitate efficient skill acquisition and backup strategy formation. If the model is too detailed it could make the learning of the system handling skills more difficult.

Data obtained in our study (Kaptelinin, 1992) are in accordance with this assumption. In this study two groups of students were learning a simple user interface of a version of a LOGO - based environment in two different manners. One group was taught with the use of state - transition diagram, which was supposed to support the formation of mental model of the system. A set of rules describing the behavior of the system was exposed to other group. The results indicated that providing students with an explicit state-transition diagram of the interactive environment interfered with acquisition of the system handling skills.

So, the nature of mental models and their role in human computer interaction can be adequately understood only within the general context of a user's skill automatization. A closer look at the mechanisms of skill automatization in users and further investigations into the efficiency of various teaching methods are needed to realise the full potential of mental models in the field of human - computer interaction.

6. Human computer interaction in developmental and social contexts.

In this section we will consider the general aspects of human computer interaction -- the developmental and (very briefly) social contexts of this interaction. While doing this, we will heavily rely on the notion of computer tools as an extension of the IPA.

First of all, it is necessary to identify new features of the IPA determined by the integration of computer tools into the structure of activity. Our point is that as a consequence of this integration the IPA becomes potentially shared and self - realizing.

The product and the process of manipulation with a model, even if they are intended solely for private use, are available for others. While the results of solving a problem in the "nonextended"

IPA can only be applied to reality by the subject him/herself, a solution found with the extension of the IPA can often be realized automatically, without any participation of the subject. So the use of computers makes the border between internal and external phenomena more fuzzy, bringing subjective reality closer to the world of people and the word of things.

It is reasonable to suppose that the most profound impact of the IPA extensions is on the "nonextended" IPA. The nature of this influence is, however, not clear enough yet. On the one hand, the delegation of IPA functions to computer tools can result, as discussed previously, in some negative consequences. On the other hand, computer tools can stimulate IPA development. For example, the appropriate organization of computer based learning activities can add a new dimension to the developing IPA through providing an opportunity for a student to model not only objects but processes as well (Kaptelinin, Stetsenko, 1989).

Empowering people means that goals are reached with less time and/or effort. The question is, what are the ways people use to invest the reserved resources? This question is not purely theoretical. It is sometimes necessary, in order to solve very practical problems (for example in marketing), to anticipate the transformation of human behavior resulting from wide use of a computer tool.

In other words, what is the influence of computer tools on goal setting processes? An answer to this question can be only found in studies based on the analysis of human activity as a whole, because goal setting is determined jointly by a motive and the means available to reach this motive.

Concerning the role of social interactions surrounding computer use, it should be noted that conceptual tools for such an analysis are, unfortunately, rather poorly elaborated in activity theory. For many years specific, objective, unbiased studies of social factors were not allowed by the political system in the former USSR. Some very interesting results were, however, obtained by proponents of activity theory outside the USSR. From our point of view, the most promising approach is proposed by A. Raeithel (1992). His notion of "figurations", which referred to the units of interpersonal relations, seems to be

helpful in overcoming some difficulties related with the traditional activity theory interpretation of interpersonal relations as components of a motive - driven activity of a so called "collective subject" (Andreeva, 1980).

7. Conclusions.

The major problems of cognitive ergonomics can be formulated as "what is a usable computer system and how can it be used in the most efficient way?". From the point of view of activity theory, to solve these problems we should consider human computer interaction within the context of human interaction with his/her environment. Activity theory proposes a paradigm for description and understanding of this interaction, one which provides "vertical" integration of different levels of psychological analysis and stresses some points that are usually missed by the cognitive approach.

These points, discussed in the present paper, include: (a)the goal - determined nature of human actions, their structure and performance, (b)the role of mechanisms underlying the integration of a computer tool into the structure of activity, (c)the importance of developmental aspects of human computer interaction, (d)the significance of social interactions surrounding computer use.

From the above it might be assumed that a psychological enquiry into human computer interaction should cover the following aspects: functions of computer tools in the structure of human activity (i.e., the psychological nature of functional organs which emerge from the use of these tools), the process of acquiring interaction skills and the impact of this process on the cognitive and communication functions of a user, the conformity of computer tools to the user's goals and tasks, and to the social context. In the previous sections two problems of cognitive ergonomics -- the limitations of WIMP interfaces and the role of mental models in human computer interaction -- were considered from the point of view of activity theory.

It can be concluded that cooperation between cognitive ergonomics and activity theory can be useful for both of these approaches. On the one hand, activity theory, which is a well elaborated conceptual system providing coherent descriptions of processes at various levels, as well as analysis of contextual factors, can add some important perspective to the traditional cognitive approach. On the other hand, the use of cognitive models and powerful experimental methods, as well as assimilation of the empirical data gained in cognitive studies, seems to be one of the most promising lines of activity theory development.

Acknowledgment.

I would like to thank Maurice Naftalin for improving the English in this paper.

References.

Andreeva G.M. (1980). Object - activity principle and development of the system of social - psychological knowledge. Vestnik MGU. Psikhologia, #4. (in Russian)

Bødker S. (1991). Through the interface: a human activity approach to user interface design. Hillsdale, NJ.

Carroll J.M., Olson J.R. (1988). Mental models in human - computer interaction. In: Helander M. (ed.) Handbook of Human - Computer Interaction. Amsterdam, etc..

Grudin J. (1990). The computer reaches out: the historical continuity of interface design. In: Proceedings of CHI'90. Seattle, WA.

Grudin J. (1991). Utility and usability: Research issues and development contexts. In: Proceedings of the 1st International Moscow HCI'91 Workshop. Moscow.

Kaptelinin V.N. (1992). Can mental models be considered harmful? In: Proceedings of CHI'92. Short talks and posters. Monterey, CA.

Kaptelinin V.N. (1991). Logo, computer literacy, and cognitive development. In: Proceedings of the Third European Logo Conference (EUROLOGO 91). Parma, Italy.

Kaptelinin V.N., Stetsenko A.P. (1989). Computer aided cognitive development: microworlds and reality. In: Proceedings of the 3rd International Conference "Children in the Information Age". V.1. Sofia.

Leontiev A.N. (1959). Problems of development of mind. Moscow. (in Russian)

Leontiev A.N. (1975). Activity. Consciousness. Personality. Moscow. (in Russian)

Leontiev A.N. (1979). The psychology of image. Vestnik MGU. Psikhologia, #2. (in Russian)

Munipov V.M. (1983). The contribution of A.N. Leontiev to the development of engineering psychology and ergonomics. In: A.N. Leontiev and modern psychology. Moscow. (in Russian)

Norman D. (1988). The Psychology of Everyday Things. NY.

Niesser U. (1981). Cognition and reality. Moscow. (in Russian)

Ponomarev Ya. A. (1975). The psychology of creativity. Moscow. (in Russian)

Price B.A., Baeker R.M. (1991). The automatic animation of concurrent programs. In: Proceedings of the 1st Moscow International HCI'91 Workshop. Moscow.

Raeithel A. (1992). Activity theory as a foundation for design. In: Floyd ea.. (eds.) Software Development and Reality Construction. Berlin, etc.

Rutkowski C. (1982). An introduction to the human applications standard computer interface. Part 1: theory and principles. Byte, v.7.

Shneiderman B. (1987). Designing the user interface: strategies for effective human - computer interaction. Reading, Mass., etc.

van der Veer, G. C. (1990). Human-Computer Interaction. Learning, Individual Differences, and Design Considerations. Offsetdrukkerij Haveka, Amsterdam.

Vygotsky L.S. (1956). Selected psychological works. Moscow. (in Russian)

Velichkovsky B.M. (1990). Cognitive science and psychological problems of studying intelligence. In: Computers and Cognition. Moscow. (in Russian)

Velichkovsky B.M., Zinchenko V.M. (1979). Methodological problems of modern cognitive psychology. Questions of Philosophy, #7. (in Russian)

Zinchenko V.P., Gordon V.M. (1976). Methodological problems of psychological analysis of activity. In: System Studies 1975. Moscow. (in Russian)

Zinchenko V.P., Munipov V.M. (1979). The basics of ergonomics. Moscow. (in Russian)

A Method of a Quantitative Measurement of Cognitive Complexity

M. Rauterberg

Work- and Organizational Psychology Unit
Swiss Federal Institute of Technology (ETH), Zurich, Switzerland

ABSTRACT

A theoretical framework to conceptualize measure of behaviour complexity (BC), system complexity (SC) and task complexity (TC) was developed. From this framework cognitive complexity (CC) is derived as CC = SC + TC - BC. In an empirical study to investigate different measures of cognitive complexity 6 beginners and 6 experts solved 4 different tasks with an interactive database management system. To analyze the empirical data recorded during the interactive sessions a special program was developed. The program's purpose and the program architecture are described. 4 different approaches from the literature to measuring complexity in a quantitative way were considered and discussed. Application of these 4 approaches were compared and tested against the empirical results of the experiment. The measure of McCabe (1976) proved to be the most effective.

1. Introduction

This study was carried out to support the formal analysis of studying user keystroke behaviour. The normal design cycle used to construct a formal model is a top down approach. In this paper we present an automatic bottom up approach to construct a formal description of user behaviour. The formalism we have selected to model the user's knowledge with finite state/transition systems, is the Petri net theory.
There are different formalisms for constructing user models the context of human computer interaction; TAG (Payne & Green 1986), CLG (Moran 1981), GOMS (Card, Moran & Newell 1983), CCT (Kieras & Polson 1985), and different kinds of grammars BNF (Reisner 1981), EBNF (Reisner 1984), etc. Using any of these formalisms the investigator is obliged to design the pure (="error free") user model in a top down approach. Then he or she can try to prove the model with "error free" empirical data. This is difficult, expensive and insufficient, and one of the consequences is that most of the formal models exist only as paper versions and have not been implemented as executable simulations. For a more detailed critique of the mentioned formalism see Greif & Gediga (1987), Sutcliffe (1989), Karat & Bennett (1991) or Benyon (1992).
If user models can be constructed using an automatic, bottom up approach, then the handling of formal models becomes easy. To achieve this, we first of all develop a theoretical framework, which fits our empirical approach. Based on this framework we are able to test four different measures of complexity according to their "discriminating power".

2. Cognitive complexity in man computer interaction

Cognitive complexity has been defined as "an aspect of a person's cognitive functioning which at one end is defined by the use of many constructs with many relationships to one another (complexity) and at the other end by the use of few constructs with limited relationships to one another (simplicity)" (Pervin 1984, p. 507). Transferring this broad definition to the man computer interaction could mean: the complexity of the user's mental model of the dialog system is given by the number of known dialog contexts ("constructs") on one hand, and by the number of known dialog operations ("relationships") on the other hand. The scope of this paper does not include approaches based on questionnaires (Scott, Osgood & Peterson 1979, McDaniel & Lawrence 1990). Comparing the traditional approaches to measure cognitive complexity Grote and James (1990, 419-420) come to the following conclusion, "it appears that the most basic problem concerning the concept of cognitive complexity is its lack of an unambiguous and uniformly used definition and operationalization".

2.1. Definition of cognitive complexity

Observing the behaviour of people solving a specific problem or task is our basis for estimating "cognitive complexity (CC)". The cognitive structures of users are not direct observable, so we need a method and a theory to use the observable behaviour as one parameter to estimate CC. A second parameter is a description of the action or problem solving space itself. The third parameter is an "objective" measure of the task or problem structure.

We call the complexity of the observable behaviour the "behaviour complexity (BC)". This behaviour complexity can be estimated by analysing the recorded concrete task solving process, which leads to an appropriate task solving solution. The complexity of a given tool (e.g. an interactive system) we call "system complexity (SC)". The last parameter we need is an estimation of the "task complexity (TC)".
The necessary task solving knowledge for a given task is constant. This knowledge embedded in the cognitive structure (CC) can be observed and measured with BC. If the cognitive structure is too simple, then the concrete task solving process must be filled up with a lot of heuristics or trial and error strategies. Learning how to solve a specific task with a given system means that BC decreases (to a minimum = TC) and CC increases (to a maximum = SC). We assume, that the difference (BC–TC) is equal to the difference (SC–CC).
To solve a task, a person needs knowledge about the dialogue structure of the interactive software (measured by SC) and about the task structure (measured by TC). SC is an upper limit for TC (SC≥TC); this aspect means, that the system structure constrains the complexity of the observable task solving space. We can now state with the constraints (BC≥TC) and (SC≥CC), that:

$$BC - TC = SC - CC \qquad \text{(Formula 1)}$$

Transforming this Formula 1 to get CC alone, results in Formula 2:

$$CC = SC + TC - BC \qquad \text{(Formula 2)}$$

The parameter SC is given by the concrete system structure, so we have to apply a given complexity measure to this system structure. The parameter TC can be estimated either in a theoretical or in an empirical way. All apriori descriptions of task structures are usable to calculate a theoretical TC. If we have empirical data of different task solving processes of different persons, then we can estimate TC using the minimum of all observed BCs per task. Given a sample of different complete task solving processes, the best approximation for TC seems to be the minimal solution regarding a specific complexity measurement. One plausible consequence of this assumption is that CC is equal to SC in the case of "best solution" (TC=BC). This is the approach we are presenting here.

2.2. Quantitative measurements of complexity

The cognitive complexity (CC) is defined by Formula 2. To measure the system complexity (SC) we need a description of the interactive system "behaviour". In the context of this work we are using a state/transition matrix to describe all possible dialog actions the user can use to change from one dialog state to another. Then we need to carry out an empirical investigation to observe and record user behaviour solving a set of tasks with the interactive system.
To measure complexity we introduce four different metrics. The first simple metric we use is given by Stevens, Myers and Constantine (1974). They count the number of dialog states (S: states) to measure the "absolute structural complexity of a net" (see Formula 3).

$$C_{state} = S \qquad \text{(Formula 3)}$$

The "relative structural complexity" of a net is the ratio of the number of connections between dialog states (T: transition) to the total number of dialog states (S: state) (see Formula 4). The measure C_{fan} represents the average number of connections per state, so we call this complexity measure the "fan degree" of the net.

$$C_{fan} = T / S \qquad \text{(Formula 4)}$$

The third metric we use is published by McCabe (1976). His complexity measure was created in the context of software development, to analyse large programs. The mathematical background of this measure is graph theory, which offers several procedures to describe different aspects of net structures. If we have two dimensional net structures, then we can calculate the number of basic cycles ("holes") in the net with this complexity measure (see Formula 7). The complexity is defined by the difference of the total number of connections (T: transition) and the total number of states (S: state). The parameter P is a constant to correct the result of Formula 5 in the case of a sequence (S = T + P); the value of P in our context is 1. The semantic of C_{cycle} can be described by the number of "holes" in a net. C_{cycle} is a metric to calculate the number of linear independent cycles of a plane and coherent net.

$$C_{cycle} = T - S + P \qquad \text{(Formula 5)}$$

The fourth metric is given by Kornwachs (1987). His concept was developed for a general system with elements and connections. His complexity measure was explicitly designed for man-computer interaction. The idea of this measure is to estimate the actual net density compared to the maximal possible net density.

The maximal possible net density increases proportional to the square of the number of states. Let S be the number of all elements ("states") in a given system I; the matrix of all realized connections c is given by: $c_{ij}=0$, if element $i \in I$ is disconnected with element $j \in I$, and $c_{ij}=1$, if element $i \in I$ is connected with element $j \in I$. The number of all realized (= really existing) connections is $t= \Sigma\Sigma\ c_{ij}$. All possible directed connections s of the system I can be calculated by $s=S(S-1)$. The structural density d is given by $d=t/s$. The "structural degree of complexity" $C_{density}$ is now defined as the structural density d.

$$C_{density} = T / (S*(S-1)) \qquad \text{(Formula 6)}$$

With C_{state}, C_{fan}, C_{cycle}, and $C_{density}$ we have four different metrics to measure complexity. We shall discuss the advantages and disadvantages of these four quantitative metrics in the context of an empirical investigation below.

2.3. Representations of cognitive complexity

Measurement of complexity of a system described with a state/transition matrix in a quantitative way is one central issue; the other central issue is to transform the structure of a given system in an "appropriate form". One qualitative approach to figure complexity is drawing the "net structure" of the system. If we use Petri-Nets (Petri 1980) instead of the equivalent state/transistion formalism (Wasserman 1985), we can simulate the user's mental model in an executable form with Petri-Net simulators.
A Petri-Net is a mathematical structure consisting of two non-empty disjoint sets of nodes, called S-elements and T-elements, respectively, and a binary relation F, called the flow relation. F connects only nodes of different types and leaves no node isolated. Nets can be interpretated by using a suitable pair of concepts for the sets S (signified by curved brackets "( )") and T (signified by square brackets "[ ]") and a suitable interpretation for the flow relation F (signified by an arrow "->"). The means/activity interpretation allows one to describe the static structure of a system with several active and passive functional components: means (S) = real or informational entity, and activity [T] = (repeatable) action of a system. The flow relation F means: [a] -> (m), the activity a (e.g. a dialog operation) produces means m (e.g. a dialog state); (m) -> [a], activity a uses means m. The main operations (relations) between two nets are abstraction, embedding and folding (Genrich, Lautenbach & Thiagarajan 1980). Folding is the most important operation in our context.

3. The Automatic Mental Model Evaluator (AMME)

What is the main concern of a user interacting with a software system? The user must build up a mental representation of the system's structure and gain knowledge about the functions of this system with respect to a set of tasks. Furthermore, he must learn the language, i.e., a set of symbols, their syntax, and operations connected to them, to evoke interaction sequences (the interactive "processes") related to task and subtask functions. So, the user's representations of the system structure are models of a virtual machine. A "virtual machine" is defined as a representation of the functionality of a system (functional units and their behaviour). The most important point for the user is the relation between task and machine, and not so much the internal structure of the machine's system. Consequentely, the task for the human factors engineer is to model a suitable interface as a representation of the virtual machine which can serve as a possible mental representation for the user.
The symbolic representation of the machine system consists of the following elements: 1. objects (things to operate on), 2. operations (symbols and their syntax), and 3. states (the "dialog states"). The mental model of the user can be structered in representing objects, operations, states, system structure, and task structure.

3.1. The "idea" of AMME

If a user interacts with a dialog system, he or she produces a sequence of states and transitions (s') -> [t'] -> (s") -> [t"] -> (s') -> [t'] -> (s") -> [t"] -> ... Each state corresponds to a dialog context, and each transition corresponds to a dialog operation. This sequence is called a "process". Measurable facts in the process are for example number of states and transitions, time per transition, etc. This measurements can be easily done based on a protocol of the user's behaviour automatically recorded by the dialog system in a "logfile" (the logfile recording technique; Crellin, Horn & Preece 1990).
To measure the complexity of the mental model which generates the actual process, we first need a mapping procedure from the observable process to the embedded structure of this process. This mapping procedure can be done with the folding operation in the context of Petri nets. Folding a process means to map S-elements onto S-elements and T-elements onto T-elements while keeping the F-structure. The result is the structure of the performance net. The result of a folding operation of our example sequence above is a loop (s') -> [t'] -> (s") -> [t"] -> {back to (s')}. This simple loop with two different states and two different transitions is the whole net structure we need to produce the process given above.

The prime idea of our approach is based on the actual observation of users performing a specific task. The key to the interpretation of the protocols is a map of the complete task solving domain, on which the behaviour of individual processes is drawn. This task solving domain in our approach is the whole dialog net structure of the interactive system. A sequence of keystrokes can be contemplated as a sentence derived from a defined grammar or as a process derived from a Petri net. The state transition net, as a complete description of the software the user is interacting with, can be used to identify the equal states in the keystroke sequence. All parts of the user's keystroke sequence between two dialog states are elementary processes. All elementary processes can be combined to form a Petri net (the "folding" operator). The "folded" Petri net is a formal description ("model") of the procedural knowledge of the users behaviour.

3.2. The program structure of AMME

The tool AMME consist of four different programms: (1) the dialog system with the logfile recording feature (ADIMENS 1988); (2) a transformation program, which translate the binary logfile in a readable ascii file; (3) the analysing program PACEGEN (Hofmann & Rudnik 1991), which extracts the net of the task and user specific process and calculates different quantitative aspects of the generated net; (4) the Petri net simulator PACE (Dähler 1989), which reads the extracted net given in a special syntactical format. The analysing program PACEGEN version 1.0 is programmed in MacMETH-Modula on Macintosh computers. The Petri net simulator runs in the SmallTalk runtime environment on Macintosh computers.

3.3. Operating and usage of AMME

First, we shall describe the dialog structure of ADIMENS 2.21. This description is written as an ascii file in a special state/transition syntax, defined by Hofmann and Rudnik (1991). This dialog system description is one of the neccessary inputs to the analysing program PACEGEN. The other input to PACEGEN is the logfile after translation in an ascii format. PACEGEN produce two output files: a protocol file with different quantitative measures of the extracted net and a net description file in a readable form for PACE.
The generation of the file with the dialog description is an iterative procedure. With each new logfile analysed by PACEGEN in most cases the description file must be updated. After analysing the whole set of logfiles in the first trial all logfiles must be reanalysed in a second trial. This procedure guaranties, that the calculation of the quantitative measures of each generated net is correct.

3.4. Limitations of AMME

The actual version of PACEGEN is restricted to logfiles only generated by ADIMENS 2.21. In the next version of PACEGEN the syntax of the logfiles could be defined by the researcher given a special description language to express the specific syntactical structure of the logfiles. The more general restriction comes from the type of the dialog structure (the "task domain") itself: only dialog structures, which can be expressed with a context-free grammar, are describable.

4. Analyzing user behaviour recorded in logfiles

4.1. The empirical investigation

We tested the dialog system ADIMENS 2.21 with 12 users (Ulich et al. 1991). These 12 users had have to solve 10 different benchmark tasks in the context of operating the data base system ADIMENS. We present the results of the first four different benchmark tasks.

4.1.1. Subjects

Two different user groups took part in this experiment: a beginner group (N=6: 4 women, 2 men; average age of 27 years), and an expert group (N=6: 0 woman, 6 men; average age of 38 years).

4.1.2. Experimental setting

First, the experience with computers was measured with a 115-item questionnaire and with interviews, then the beginner group was instructed for 1.5 hours in handling the database system. The expert group had 1,740 hours of experience in handling the database system ADIMENS. Their total computer experience of about 7,500 hours was the result of their daily work using different types of computers and software systems. The duration of the actual task solving session was about 30 minutes. Each keystroke with a timestamp was recorded in a logfile. Each user needed about 2 hours for the whole experiment (10 tasks, individual sessions).

4.1.3. System description

The dialog system was the relational data base system ADIMENS version 2.21 with a character oriented user interface (CUI) running on standard IBM PC's with standard keyboard. The whole dialog structure is strictly

hierarchical organized with three levels: (1) the main menu has 7 dialog operations (ordinary ascii characters chosen from a menu) to go down to 7 different modules, and 5 function keys with specific semantics; (2) at the module level each module has exactly 4 different dialog operations to change to routines and on average 4.1 (±1.7; range: 0-5) function keys with specific semantics; (3) at the routine level the user has only on average 3.7 (±2.9; range: 0-10) different function keys to control the dialog (additionally all ascii keys and the 4 cursor keys are usable).
The number of all ordinary dialog contexts (main menu, modules, routines) is 1+7*4=29. But to describe the complete dialog structure with all help, error and additional dialog states we need at least 144 different states. To change from one state to the other the system offers overall 358 different dialog operations (transitions).

4.1.4. Task description

In the experiment all users had to play the role of a camping place manager. This manager uses a database system with a data base consisting of three data files: PLACE, GROUP, and ADDRESS. All users had to solve the following four different tasks operating the database system:

Task 1: "How many data records are in the file ADDRESS, in the file PLACE, and in the file GROUP? Find out, please."

The user has to activate a specific menu option ("Datafile" in module "Info" of the menu interface) and to read the file size (solutions: PLACE = 17 data records, GROUP = 27 data records, ADDRESS = 280 data records).

Task 2: "Delete only the last data record of the file ADDRESS, the file PLACE, and the file GROUP (sorted by the attribute 'namekey')."

The user has to open (sorted according to the given attribute), select and delete the last data record (file: PLACE, GROUP, ADDRESS).

Task 3: "Search and select the data record with the namekey 'D..8000C O M' in the file ADDRESS, and show the content of all attributes of this data record on the screen. Correct this data record for the following attributes:

State:	Germany
Place number:	07
Remarks:	Database system dealer can give a demonstration."

The user must select a certain data record (file: ADDRESS), update the data record with regard to the three attributes: State, Place number, Remarks.

Task 4: "Define a filter for the file PLACE with the following condition: all holidaymakers arrived on date 02/07/87. Apply this filter to the file PLACE, and show the content of all selected data records in the mask browsing mode on the screen."

The user must define a filter for the attribute "arrival date", apply the filter to the data file PLACE, and display the content of each data record found on the screen.

4.1.5. Dependent measures

One of the most important dependent measures is the task solving time. This variable and the results of the computer experience questionnaire are needed for validating the different complexity measures. With the analysing program PACEGEN we obtained the following measures per user and per task: the number of all different transitions (T vector: "# of transitions") and the number of all different dialog states (S vector: "# of states").
We do not assume a massive learning process during the task solving period. So, a good measure of CC must differentiate between beginners and experts, but not overall between the four tasks.

4.2. Experimental results

Let us begin with the performance measure "task solving time". The means of the two groups are different (see Fig. 1). The experts are significantly faster than the beginners (see Tab. 1). This expected result is fundamental for validating the complexity measures. The other important results are the significant differences between the four tasks. [Two beginners were not be able to solve task 4, so there are two missing data. These missing data are estimated by group means and replaced.]
If the performance measure is a coarse estimation of complexity, then the order of the four tasks according to their complexity is: lowest is task 1 ("info"), followed by task 4 ("filter"), task 2 ("delete") and task 3 ("edit"). A close look at task 3 shows, that the user needs the knowledge to handle 15 different states just for the editor required in the dialog context of the routine "update"; to change from one editor state to another the user needs at least 45 different transitions (e.g. different semantics of the cursor keys). In the following analysis we evaluate the logfiles only on the routine level, so we are not able to measure the complexity below the routine level.

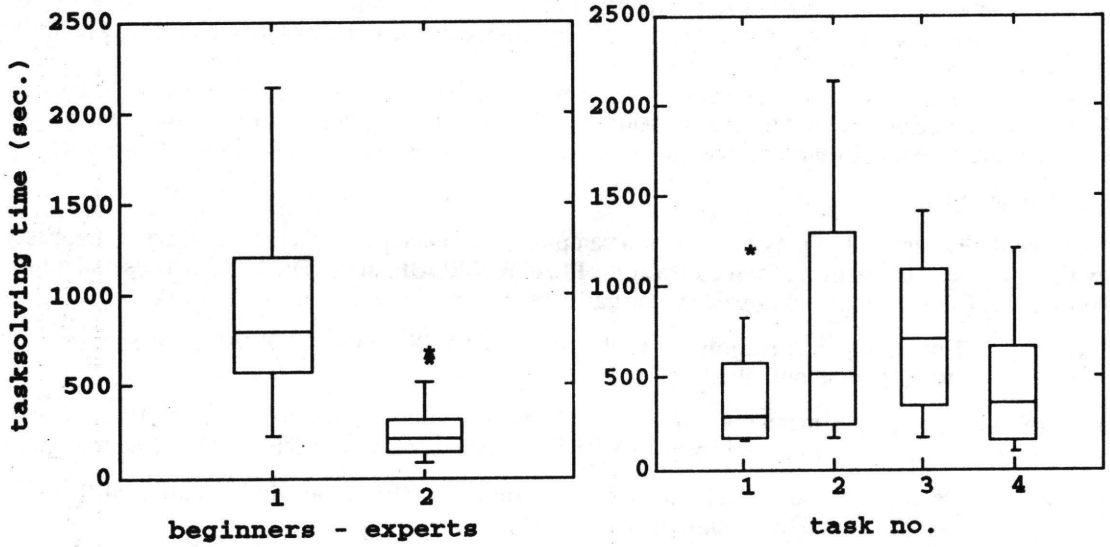


Figure 1 These two Box-and-Whisker Plots show the task solving time seperated for beginners-experts and tasks 1-4 (N=48). The central box of each condition covers the middle 50% of the data values, between the lower and upper quartiles. The "whiskers" extend out to the extremes, while the central line is at the median.

Table 1 Analysis of variance of the variable "task solving time" (N=48).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	4987786.021	1	4987786.021	45.283	0.001
tasks	1326184.229	3	442061.410	4.013	0.014
exp. x tasks	489222.229	3	163074.076	1.481	0.234
ERROR	4405844.500	40	110146.113		

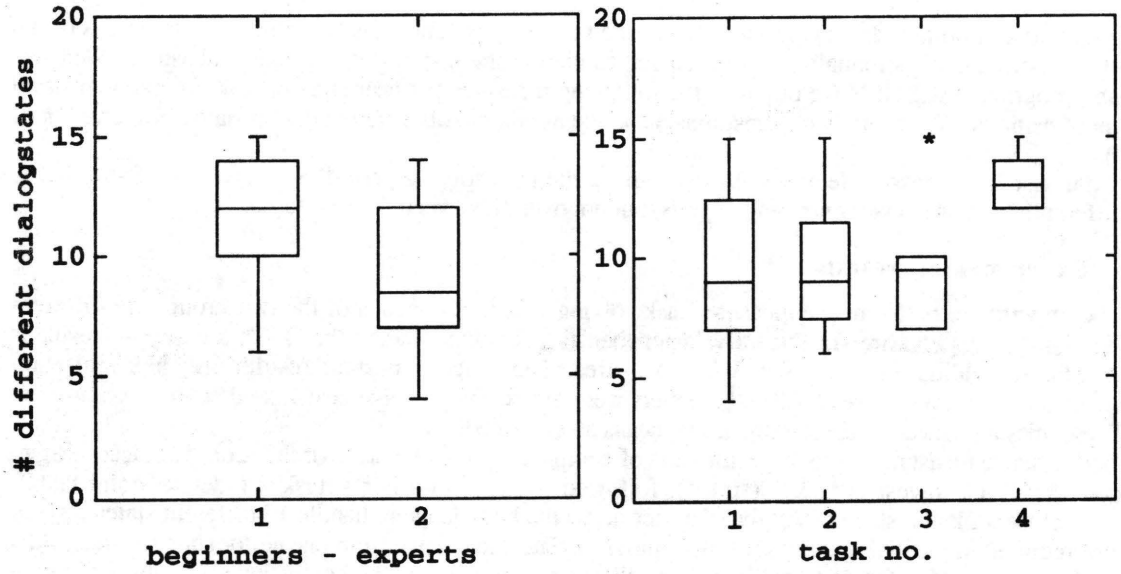


Figure 2 These two Box-and-Whiskers Plots show the results of the "number of different dialog states" (N=45, 3 outliers excluded).

Table 2 Analysis of variance of the "number of different dialog states" (N=45, 3 outliers excluded).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	54.028	1	54.028	8.830	0.005
tasks	106.271	3	35.424	5.789	0.002
exp. x tasks	15.260	3	5.087	0.831	0.485
ERROR	226.400	37	6.119		

The next estimation of complexity is the number of different dialog states. We need 144 different dialog states to describe the dialog structure of the interactive system. Solving a given task the user is navigating through this dialog structure. The average number of different dialog states the beginners need to solve all tasks is significantly higher than the average of the expert group (see Fig. 2, and Tab. 2). This result can be easily explained by the different experience of operating the system of these groups: the beginners need more heuristic search strategies to solve the tasks than the experts. There is also a significant difference between task 4 and the other three tasks.

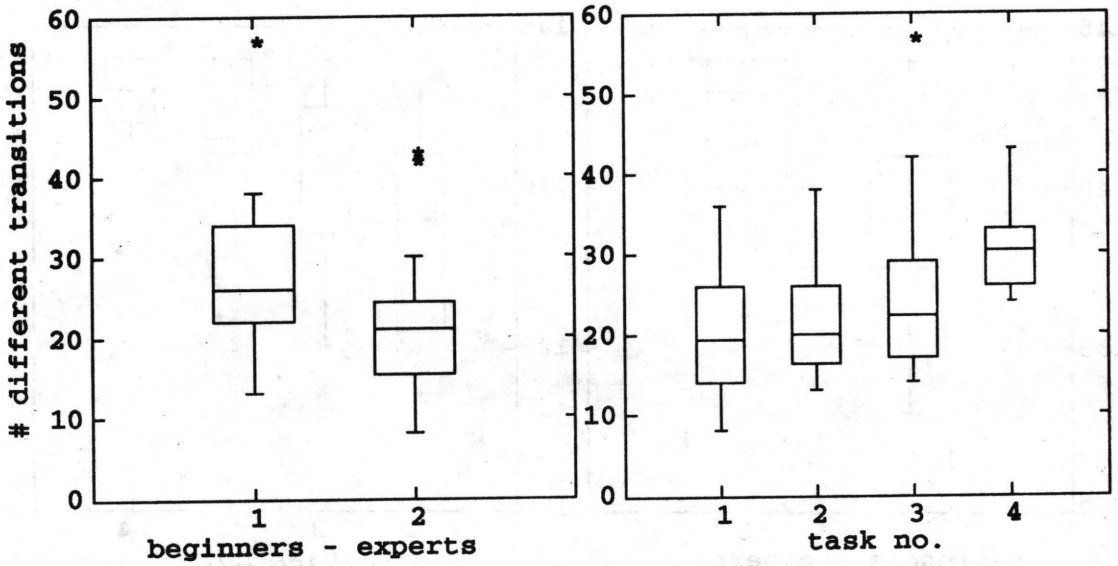


Figure 3 These two Box-and-Whiskers Plots show the results of the "number of different transitions" (N=46, 2 outliers excluded).

Table 3 Analysis of variance of the "number of different transitions" (N=46, 2 outliers excluded).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	508.255	1	508.255	6.682	0.014
tasks	654.711	3	218.237	2.869	0.049
exp. x tasks	88.974	3	29.658	0.390	0.761
ERROR	2890.500	38	76.066		

The beginners also use significantly more different transitions than the experts (see Fig. 3, and Tab. 3). This result support the interpretation, that the beginners need more heuristics to solve the tasks than the experts. In solving task 4 all users need the most transitions. The main effect "tasks" in the analysis of variance is significant at the 5% level (see Tab. 3).
These results of the three dependent measures are the basis for validating the 4 different complexity measures.

5. Validation of the four different complexity measures

To estimate the cognitive complexity we must first calculate the behaviour complexity ("BC") of each user and each task. Then we estimate the task complexity ("TC") of each task by searching for the minimum of the 12 empirical values of the behaviour complexity (the "best" solution). The system complexity is given by the system description, and is always constant in the context of a specific complexity measure.

5.1. "Cognitive Complexity" based on the measure of "different dialog states"

,First we present the results of the measure of cognitve complexity CC_{state} of Stevens et al. (1974) according to the number of different dialog states (S: states). The results for the behavioral complexity BC_{state} are shown above (see Fig. 2, and Tab. 2). The following estimations of the behavioural, system, and task complexity are set:

Behaviour Complexity:	$BC_{state} = S$	
System Complexity:	$SC_{state} = 144$ (number of total states)	
Task Complexity:	TC_{state}:	$\min[BC_{state}]_{task\text{-}1} = 4$,
		$\min[BC_{state}]_{task\text{-}2} = 6$,
		$\min[BC_{state}]_{task\text{-}3} = 7$,
		$\min[BC_{state}]_{task\text{-}4} = 12$
Cognitive Complexity:	$CC_{state} = SC_{state} + TC_{state} - BC_{state}$	

The global equation to calculate the cognitive complexity CC is given by Formula 2 above. The TCs are in the expected direction: task 1 low complex, task 4 high complex.

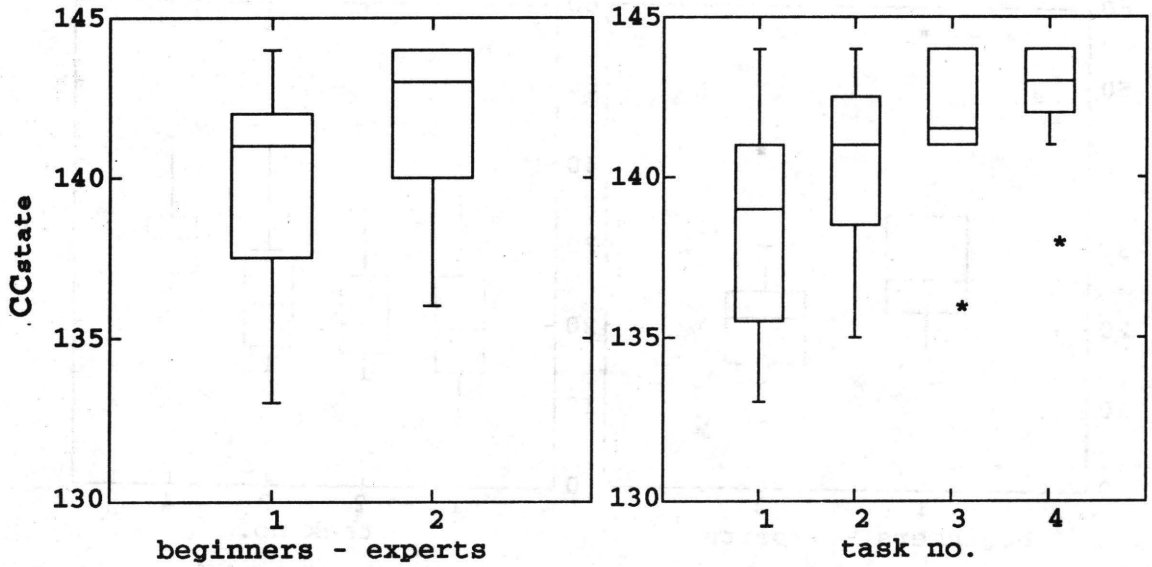


Figure 4 These two Box-and-Whiskers Plots show the results of the cognitive complexity "CC_{state}" measure according to Stevens et al. (N=46, 2 outliers excluded).

Table 4 Analysis of variance of the cognitive complexity "CC_{state}" measure according to Stevens et al. (N=46, 2 outliers excluded).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	44.579	1	44.579	6.757	0.013
tasks	107.853	3	35.951	5.449	0.003
exp. x tasks	24.230	3	8.077	1.224	0.314
ERROR	250.700	38	6.597		

This measure CC_{state} is good enough to differentiate between beginners and experts (see Fig. 4, and Tab. 4). As one can see in Figure 4, the cognitive complexity CC_{state} of the performance model of the experts is significantly higher than the cognitive complexity CC_{state} of the beginners. This outcome is valid and meaningful. But the result, that the average complexity CC_{state} of the cognitive model of task 3 and 4 is higher than the complexity CC_{state} of task 1 and 2, is counter intuitive. This result would enforce the interpretation, that the users had more knowledge of the "more complex" tasks than of the "less complex" tasks. This outcome can only be explained, if there was a massive learning process during the task solving period. This assumption could be plausible for beginners, but not for the experts, so, if this interpretation is correct, then we must have a significant interaction "exp. x tasks" in the analysis of variance. But there is no significant interaction (see Tab. 4). Let us see, how this aspect will be handled by the other 3 measures of complexity.

5.2. "Cognitive Complexity" based on the measure of "fan degrees"

Now we present the results of the measure of cognitve complexity of Stevens et al. (1974) according to the average fan degree of each dialog state. The total number of all possible dialog states (S) is 144, and of all transitions (T) is 358 (given by the system description).

Behaviour Complexity:	$BC_{fan} = T/S$	
System Complexity:	$SC_{fan} = 358/144 = 2.486$	
Task Complexity:	TC_{fan}:	$\min[BC_{fan}]_{task\text{-}1} = 1.875$,
		$\min[BC_{fan}]_{task\text{-}2} = 2.077$,
		$\min[BC_{fan}]_{task\text{-}3} = 2.000$,
		$\min[BC_{fan}]_{task\text{-}4} = 2.000$
Cognitive Complexity:	$CC_{fan} = SC_{fan} + TC_{fan} - BC_{fan}$	

The average fan degree of all 144 dialog states is 2.486 (SC_{fan}). This means that the user can choose on average between 2 and 3 alternatives to go on navigating in the dialog structure. If we take the fact that the 29 main dialog contexts described above have between 3.7 and 12.0 transitions alternatives (see section on "system description"), then we can derive from the system description, that the most dialog states in the system description are of simple decision quality: "go on" ($SC_{fan} = 1$; 48%), "yes" or "no", "ok" or "cancel" ($SC_{fan} = 2$; 22%), ($SC_{fan} = 3$; 11%) and ($SC_{fan} > 3$; 19%). The TC_{fan} measure does not differentiate very well between the four tasks.

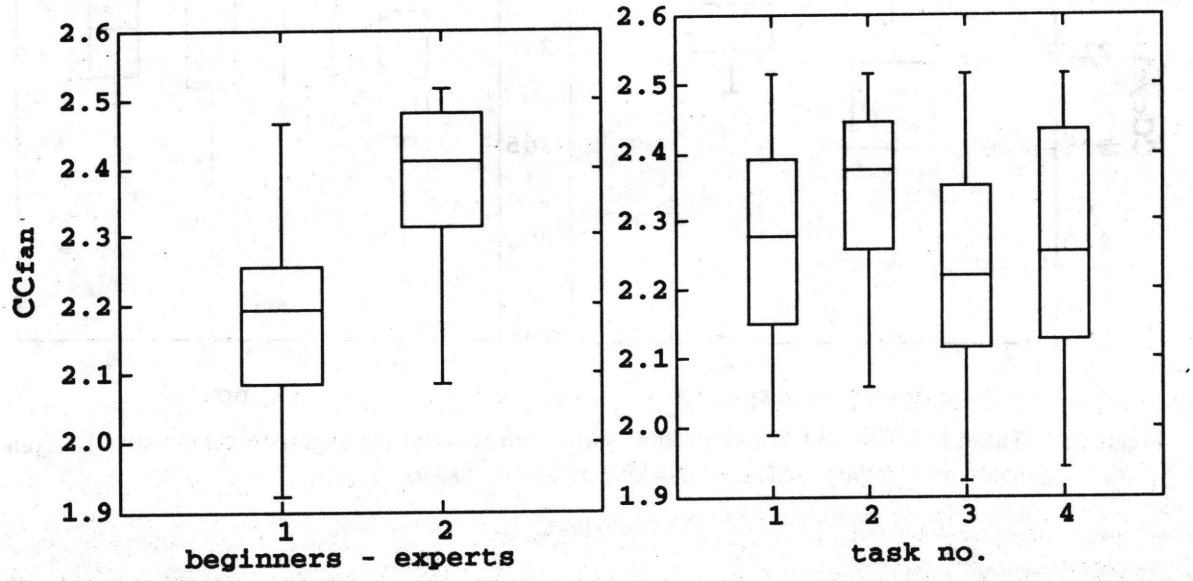


Figure 5: These two Box-and-Whiskers Plots show the results of the cognitive complexity "CC_{fan}" measure according to Stevens et al. (N=48)

Table 5 Analysis of variance of the cognitive complexity "CC_{fan}" measure according to Stevens et al. (N=48).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	0.494	1	0.494	29.026	0.001
tasks	0.085	3	0.028	1.667	0.189
exp. x tasks	0.022	3	0.007	0.423	0.738
ERROR	0.680	40	0.017		

This measure CC_{fan} is able to differentiate between beginners and experts in the expected direction: the complexity of the cognitive models of the experts is significantly higher than the complexity of the beginner models (see Fig. 5, and Tab. 5). The average cognitive complexity CC_{fan} of all users does not differ in the four tasks.

5.3. "Cognitive Complexity" based on the measure of "net cycles" (McCabe)

The empirical values of the number of states (S) and transitions (T) of each user per task are given by the vectors S and T.

Behaviour Complexity: $BC_{cycle} = T - S + 1$

System Complexity: $SC_{cycle} = 358 - 144 + 1 = 215$

Task Complexity: TC_{cycle}: $\min[BC_{cycle}]_{task\text{-}1} = 5$, $\min[BC_{cycle}]_{task\text{-}2} = 8$, $\min[BC_{cycle}]_{task\text{-}3} = 8$, $\min[BC_{cycle}]_{task\text{-}4} = 13$

Cognitive Complexity: $CC_{cycle} = SC_{cycle} + TC_{cycle} - BC_{cycle}$

It is important to notice, that the task complexity TC_{cycle} of the different tasks is in the expected direction: task 1 low in complexity and task 4 is highly complex.

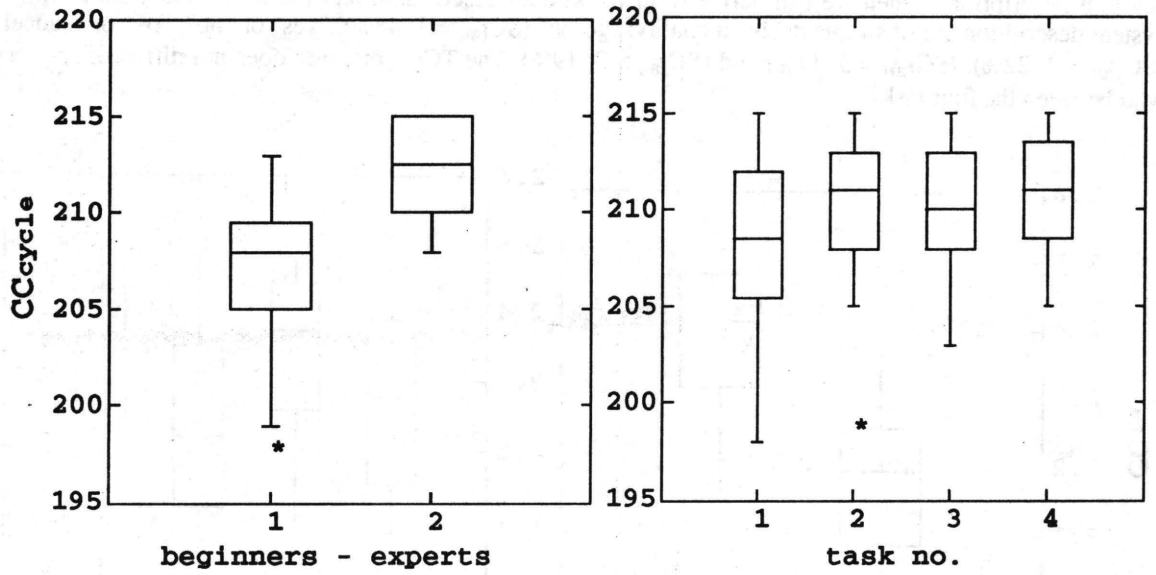


Figure 6: These two Box-and-Whiskers Plots show the results of the cognitive complexity "CC_{cycle}" measure according to McCabe (N=45; 3 outliers excluded).

Table 6 Analysis of variance of the cognitive complexity "CC_{cycle}" measure according to McCabe (N=45, 3 outliers excluded).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	272.596	1	272.596	21.547	0.001
tasks	53.326	3	17.775	1.405	0.257
exp. x tasks	10.813	3	3.604	0.285	0.836
ERROR	468.100	37	12.651		

The cognitive complexity CC_{cycle} of the mental models of beginners is significantly less complex than the cognitive complexity of the mental models of the experts. This result is in accordance with our starting point. CC_{cycle} does not differ in the four tasks (see Fig. 6); so, we do not find a significant difference for the factor "tasks" in the analysis of variance (see Tab. 6). The maximum of CC_{cycle} is SC_{cycle} (= 215). This maximum is reached by the expert group (see Fig. 6).

5.4. "Cognitive Complexity" based on the measure of "net density" (Kornwachs)

The empirical values of the number of states (S) and transitions (T) of each user per task are given by the vectors S and T.

Behaviour Complexity: $BC_{density} = T/(S*(S-1))$

System Complexity: $SC_{density} = 358/(144*(144-1)) = 0.017$

Task Complexity: $TC_{density}$: $\min[BC_{density}]_{task\text{-}1} = 0.667$,
$\min[BC_{density}]_{task\text{-}2} = 0.433$,
$\min[BC_{density}]_{task\text{-}3} = 0.333$,
$\min[BC_{density}]_{task\text{-}4} = 0.182$

Cognitive Complexity: $CC_{density} = SC_{density} + TC_{density} - BC_{density}$

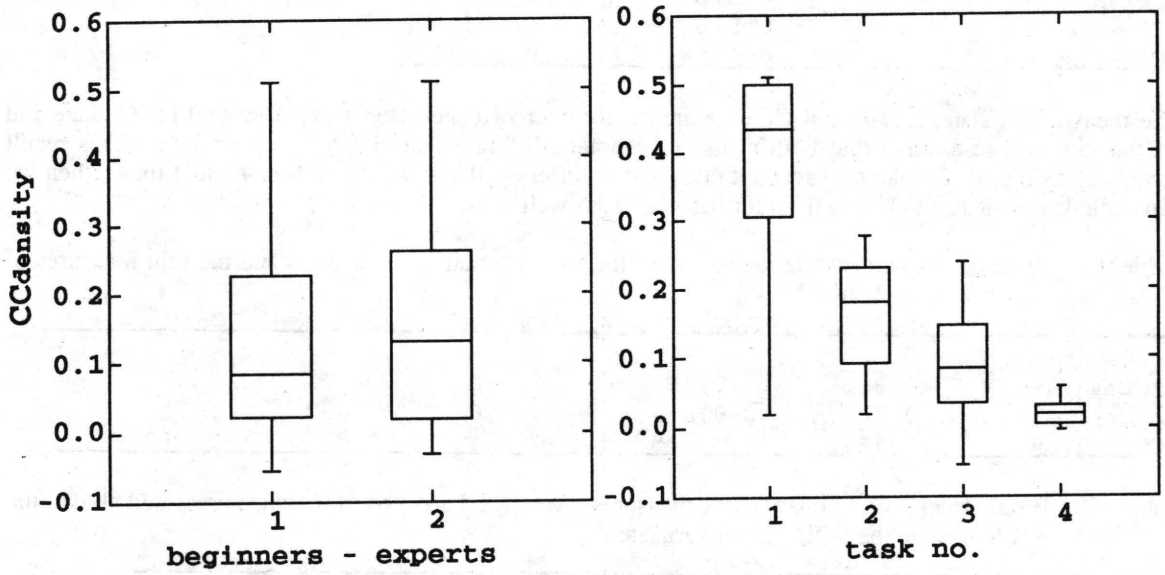


Figure 7: The two Box-and-Whiskers Plots show the results of the cognitive complexity "$CC_{density}$" measure according to Kornwachs (N=48)

Table 7 Analysis of variance of the cognitive complexity "$CC_{density}$" measure according to Kornwachs (N=48).

SOURCE	SUM-OF-SQUARES	DF	MEAN-SQUARE	F-RATIO	P
experience	0.042	1	0.042	4.355	0.043
tasks	0.916	3	0.305	31.621	0.001
exp. x tasks	0.042	3	0.014	1.438	0.246
ERROR	0.386	40	0.010		

The estimations of task complexity $TC_{density}$ are counter intuitive: task 1 is highly complex and task 4 is low in complexity! Also, the system complexity $SC_{density}$ is lower than each task complexity $TC_{density}$. This fact leads to negative complexity values (see Fig. 7). We did not include in the construction of $CC_{density}$ the aspect of the structural complexity of hierarchies (see Kornwachs 1987), so, perhaps, this surprising result comes from excluding this aspect.

The measure $CC_{density}$ differentiates poorly between beginners and experts, but very well between the four different tasks (see Tab. 7). The average values of $CC_{density}$ of the cognitive performance models according to the tasks are in a surprising direction: the users loose their knowledge during the task solving period (see Fig. 7). This outcome is not plausible.

6. Discussion and conclusions

Based on the assumption, that our theoretical model of cognitive complexity given by Formulas 1-4 is valid and meaningful, we are able to test and validate the four different measures of complexity. After the first test trial presented above, we bring the four different complexity measures (CCs) in relation to other aspects of our experimental "reality". First, we calculate the product moment correlation of the CCs with "task solving time". As one can see from Table 8, there is a significant negative correlation between CC_{state}, CC_{fan}, and CC_{cycle} with the task solving time. This is a valid and plausible outcome. Only $CC_{density}$ correlates positively (insignificant) with task solving time (see Tab. 8) and with number of dialog states (see Tab. 9), so we can exclude the measure $CC_{density}$ from further considerations.

Table 8 Product moment correlation matrix of "task solving time" with the different quantitative measures (N=48).

	CORRELATION	PROBABILITIES
# of states	0.356	0.013
# of transistions	0.461	0.001
CCstate	-0.427	0.002
CCfan	-0.576	0.001
CCcycle	-0.576	0.001
CCdensity	0.126	0.394

The measues CC_{state}, CC_{fan}, and CC_{cycle} are highly intercorrelated (see Tab. 9), except for CC_{state} and CC_{fan}. So we can assume that both measures estimate different qualities of a net structure. This result sounds plausible. If we take into account that CC_{state} differs with the tasks (see Fig. 4 and Tab. 4), then we can exclude the measure CC_{state} from further studies, as well.

Table 9 Product moment correlation matrix of the two quantitative net aspects and the four measures of cognitive complexity (N=48).

	#STATES	#TRANSITION	CCstate	CCfan	CCcycle
CCstate	-0.685	-0.675			
CCfan	-0.363	-0.527	0.338		
CCcycle	-0.792	-0.853	0.900	0.657	
CCdensity	0.162	0.115	-0.724	-0.066	-0.463

Table 10 F-ratio matrix of all quantitative measures given by all analysis of variance presented above (the 3 columns are the SOURCE of variance).

	"experience"	"tasks"	"exp.x tasks"
task solving time	45.283	4.013	1.481
# of dialog states	8.830	5.789	0.831
# of transitions	6.682	2.869	0.390
CCstate	6.757	5.449	1.224
CCfan	29.026	1.667	0.423
CCcycle	21.547	1.405	0.285
CCdensity	4.355	31.621	1.438

We introduce the idea of "discriminating power" of the measures above. This discriminating power can be expressed as an F-ratio. Our definition of "discriminating power" is positively correlated with the F-ratio value given in an analysis of variance. We presume that the measure CC is a more or less stable attribute of the user in the scope of our study. CC is changing only during a learning process. We do not assume that the experts in our investigation acquire fresh knowledge of operating the interactive system during the task solving period. This assumption is valid, because the experts are highly skilled over several years of operating the database system ADIMENS.

To compare the discriminating power of both of the remaining measures CC_{fan} and CC_{cycle} we give an overview of the F-ratio of the three sources of variance given by the analysis of variance (see Tab. 10). Overall, the measures CC_{fan} and CC_{cycle} discriminate sufficiently between beginners and experts and not between the four tasks.

If we take into account that the measure TC_{fan} is not really appropriate to differentiate between the tasks, we can accept that the measure CC_{cycle} meets all demands. Overall, the best quantitative measure of complexity in our context seems to be CC_{cycle}. This measure estimates the average number of basic cycles in a net structure. Most empirical values of this measure lie between 195 and 215 near to the maximum of 215, which indicates that only some expert users had a complete knowledge of the system structure (see Fig. 5). To reach the maximum value of SC is only possible, when BC is equal to TC.

We can conclude that the four different quantitative measures of complexity we tested are of different value in measuring task and cognitive complexity. The measure of McCabe (1976) seems to be the most appropriate measure.

Acknowledgments

We gratefully acknowledge the fruitful discussions with Gudela Grote, the valuable remarks of Andrew Shepherd and the great support in developing the different programs by Thomas Greutmann, Jens Hofmann, Jack Rudnik and Martin Roth.
The preparation of this paper was supported by the BMFT (AuT programme) grant number 01 HK 706-0 as part of the BOSS "User oriented Software Development and Interface Design" research project.

References

ADIMENS (1988). A relational database system Version 2.21d. ADI Software GmbH, Hardeckstrasse 5, D-7500 Karlsruhe, Germany.

Benyon D. (1992). The role of task analysis in systems design. Interacting with Computers, 4(1):102-123.

Card S.K., Moran T.P. & Newell A. (1983). The psychology of human computer interaction. Erlbaum.

Crellin J , Horn T., Preece J. (1990). Evaluating Evaluation: A Case Study of the Use of Novel and Conventional Evaluation Techniques in a Small Company. Human-Computer Interaction - INTERACT '90 (Diaper D et al.; eds.) Elsevier, 329-335.

Dähler J. (1989). PACE user's manual. Pace Inc., Neptunstrasse 16, CH-8280 Kreuzlingen, Switzerland.

Genrich H.J., Lautenbach K. & Thiagarajan P.S. (1980). Elements of general net theory. Lecture Notes in Computer Science vol. 84 "Net Theory and Applications" (Bauer, W.; ed.) Springer, 21-163.

Greif S. & Gediga G. (1987). A critique and empirical investigation of the "one-best-way-models" in human-computer interaction. Psychological Issues of Human-Computer Interaction in the Work Place (Frese M., Ulich E. & Dzida W.; eds.) Elsevier, 357-377.

Grote F.G. & James L.R. (1990). A policy capturing approach to the measurement of cognitive complexity. European Perspectives in Psychology vol. I (Drenth P.J.D., Sergeant J.A. & Takens R.J.; eds.) Wiley & Sons, 417-434.

Hofmann J. & Rudnik J. (1991). PACEGEN: ein automatisches Logfile-Auswertungsprogramm zur Generierung von Petri Netzen. unpublished technical report, Work and Organizational Psychology Unit, Swiss Federal Institute of Technology (ETH).

Karat J. & Bennett J. (1991). Modelling the user interaction methods imposed by designs. Mental Models and Human-Computer Interaction (Tauber M.J. & Ackermann D.; eds.) Elsevier, 257-269.

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

Kornwachs K. (1987). A quantitative measure for the complexity of man-machine interaction process. Proceedings of Human-Computer Interaction - INTERACT'87 (Bullinger H.-J. & Shackel B.; eds.) Elsevier (North-Holland). 109-116.

McCabe T. (1976). A complexity measure. IEEE Transactions on Software Engineering, SE-2(6):308-320.

McDaniel E. & Lawrence C. (1990). Levels of cognitive complexity: an approach to the measurement of thinking. Springer.

Moran T.P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. International Journal of Man-Machine Studies, 15:3-50.

Payne S.J. & Green T.G.R. (1986). Task-action grammars: a model of the mental representation of task languages. Human Computer Interaction, 2:93-133.

Pervin L.A. (1984). Personality. Wiley.

Petri C.A. (1980). Introduction to general net theory. Lecture Notes in Computer Science vol. 84 "Net Theory and Applications" (Bauer, W.; ed.) Springer, 1-19.

Reisner P. (1981). Formal grammar and human factors design of an interactive graphics system. IEEE Transactions on Software Engineering, SE-7(2):229-240.

Reisner P. (1984). Formal grammar as a tool for analyzing ease of use. Human Factors in Computing Systems (Thomas J.C. & Schneider M.L.; eds.) Ablex, 53-78.

Scott W.A., Osgood D.W. & Peterson C. (1979). Cognitive structure: theory and measurement of individual differences. Wiley.

Stevens W.P., Myers G.J. and Constantine L.L. (1974). Structured design IBM System Journal, 13(2):115-139.

Sutcliffe A. (1989). Task analysis, systems analysis and design: symbiosis or synthesis? Interactng with Computers, 1(1):6-12.

Ulich E., Rauterberg M., Moll T., Greutmann T. & Strohm O. (1991). Task orientation and User-Oriented Dialog Design. International Journal of Human-Computer Interaction, 3(2):117-144.

Wasserman A.I. (1985). Extending state transition diagrams for the specification of human-computer interaction. IEEE Transactions on Software Engineering, SE-11(8):699-713.

List of Contributors

Beth Adelson, Department of Psychology, Rutgers University, Camden NJ 08102, USA

Carl Martin Allwood, Department of Psychology, University of Göteborg, P.O. Box 14158, S-40020 Göteborg, Sweden

Miklòs Antalovits, Technical University of Budapest, Egry J.u.l. "E". bldg. III. 11.,1111 Budapest, Hungary

Sebastiano Bagnara, Università di Siena & Istituto di Psicologia del CNR, via Roma 47, I-53100 Siena, Italy

Yvan Burmistrov, Department of Psychology, Moscow State University, Prospekt Marksa 18-5, K-9 Moscow 103009, Russia

Paul F. Byerley, Terminal Research Centre, Standard Elektrik Lorenz AG, Postfach 1760, D-7530 Pforzheim, Germany

L. Candy, LUTCHI Research Centre, Loughborough University of Technology, Loughborough, Leicestershire LE11 3TU, United Kingdom

Cristiano Castelfranchi, Istituto di Psicologia - CNR, Viale Marx 15, I-00137 Roma, Italy

Amedeo Cesta, Istituto di Psicologia - CNR, Viale Marx 15, I-00137 Roma, Italy

Elizabeth Churchill, Department of Psychology, The University of Nottingham, University Park, Nottingham NG7 2RD, United Kingdom

Rosaria Conte, Istituto di Psicologia - CNR, Viale Marx 15, I-00137 Roma, Italy

I. Crevits, Laboratoire d'Automatique Industrielle et Humaine, Université de Valenciennes et du Hainaut Cambrésis, Le Mont Houy, BP 311, 59304 Valenciennes Cedex, France

Simon P. Davies, Department of Psychology,The University of Nottingham, University Park, Nottingham NG7 2RD, United Kingdom

S. Debernard, Laboratoire d'Automatique Industrielle et Humaine, Université de Valenciennes et du Hainaut Cambrésis, Le Mont Houy, BP 311, 59304 Valenciennes Cedex, France

Ian Denley, Ergonomics Unit, University College London, 26 Bedford Way, London WC1H 0AP, United Kingdom

Giorgio De Michelis, Istituto RSO, Via Leopardi 1, I-20123 Milano, Italy

Alice Dijkstra, Faculty of Social and Behavioural Sciences, Unit of Experimental and Theoretical Psychology, RU Leiden, Wassenaarseweg 52, P.O.Box 9555, 2300 RB Leiden, The Netherlands

Paolo Donzelli, ISD European Space Agency (ESA/ESRIN), Frascati, Roma, Italy

E.A. Edmonds, LUTCHI Research Centre, Loughborough University of Technology, Loughborough, Leicestershire LE11 3TU, United Kingdom

Donatella Ferrante, Dipartimento di Psicologia, Università di Trieste, viale dell'Università 7, I-34123 Trieste, Italy

Thomas R.G. Green, MRC Applied Psychology Unit, 15 Chaucer Road, Cambridge CB2 2EF, United Kingdom

Corinne Grusenmeyer, Service Ergonomie et Psychologie Industrielle, Avenue de Bourgogne, BP no27, 54501 Vandoeuvre Cedex, France

Geert de Haan, Department of Computer Science, Free University, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands

Sibylle Hermann, Terminal Research Centre, Standard Elektrik Lorenz AG, Postfach 1760, D-7530 Pforzheim, Germany

Carla Huls, Nijmegen Institute for Cognition and Information, KU Nijmegen, Montessorilaan 3, P.O.Box 9104, 6500 HE Nijmegen, The Netherlands

C.M.M. Hurts, Dept. of Behavioral Computer Science, Leiden University, P.O. Box 9555, 2300 RB Leiden, The Netherlands

Lajos Izsó, Technical University of Budapest, Egry J.u.l. "E". bldg. III. 11., 1111 Budapest, Hungary

Tomas Kalén, Department of Psychology, University of Göteborg, P.O. Box 14158, S-40020 Göteborg, Sweden

Victor Kaptelinin, Institute of General and Educational Psychology, Russian Academy of Education, Moscow, Russia

Paolo Legrenzi, Dipartimento di Psicologia, Università di Trieste, viale dell'Università 7, I-34123 Trieste, Italy

Lena Linde, National Defence Research Establishment, Department of Human Studies (FOA 5), S-172 90 Sundbyberg, Sweden

Mario Malfatti, Università di Siena & Istituto di Psicologia del CNR, via Roma 47, I-53100 Siena, Italy

Jon May, MRC Applied Psychology Unit, 15 Chaucer Road, Cambridge CB2 2EF, United Kingdom

P. Millot, Laboratoire d'Automatique Industrielle et Humaine, Université de Valenciennes et du Hainaut Cambrésis, Le Mont Houy, BP 311, 59304 Valenciennes Cedex, France

S.M. O'Brien, LUTCHI Research Centre, Loughborough University of Technology, Loughborough, Leicestershire LE11 3TU, United Kingdom

Reinhard Oppermann, GMD FIT, Postfach 1316, D-5205 Sankt Augustin 1, Germany

Marion Petre, MRC Applied Psychology Unit, 15 Chaucer Road, Cambridge CB2 2EF, United Kingdom

Matthias Rauterberg, Work and Organizational Psychology Unit, Swiss Federal Institute of Technology (ETH), ETH-Zentrum, Nelkenstrasse 11, CH-8092 Zurich, Switzerland

Antonio Rizzo, Università di Siena & Istituto di Psicologia del CNR, via Roma 47, I-53100 Siena, Italy

N.P. Rousseau, LUTCHI Research Centre; Loughborough University of Technology, Loughborough, Leicestershire LE11 3TU, United Kingdom

Thomas Schael, Istituto RSO, Via Leopardi 1, I-20123 Milano, Italy

F. Vanderhaegen, Laboratoire d'Automatique Industrielle et Humaine, Université de Valenciennes et du Hainaut Cambrésis, Le Mont Houy, BP 311, 59304 Valenciennes, CedexFrance

Gerrit C. van der Veer, Department of Computer Science, Free University, De Boelelaan 1081 A, 1081 HV Amsterdam, The Netherlands

Arnold P.O.S Vermeeren, Delft University of Technology, Jaffalaan 9, 2628 BX Delft, The Netherlands

Ulla-Britt Voigt, Terminal Research Centre, Standard Elektrik Lorenz AG, Postfach 1760, D-7530 Pforzheim, Germany

Karl-Gustaf Waern, Department of Psychology, Stockholm University, S-10691 Stockholm, Sweden

Yvonne Waern, Department of Psychology, Stockholm University, S-10691 Stockholm, Sweden

Charles C. Wood, School of Cognitive and Computing Sciences; University of Sussex Falmer CB2 1AB, Brighton, United Kingdom

Buni Zeller, Istituto RSO, Via Leopardi 1, I-20123 Milano, Italy.

Artigiana Multistampa snc
Via Ruggero Bonghi, 36 - 00184 Roma