Mental Models and User–Centered Design

Workshop Report
Breckenridge, Colorado
January 13–15, 1988

Althea A. Turner, Editor

Technical Report No. 88–9

Institute of Cognitive Science
Techical Report 88–9

Alther A. Turner (Editor)

Mental Models and User–Centered Design

Workshop Report
Breckenridge, Colorado
January 13–15, 1988

List of Participants (Addresses appear in Appendix):

Participants from Outside Institutions:

- Stuart Card (Xerox PARC)
- Michael Drilling (ARI)
- Jonathan Grudin (MCC)
- Steve Goldberg (ARI)
- Julia Hough (Lawrence Erlbaum)
- Michael Kaplan (ARI)
- Dave Kieras (University of Michigan)
- R. Thomas King (Gesellschaft fuer Mathematik und Datenverarbeitung)
- Catherine Marshall (U S West Advanced Technologies)
- Jim Miller (MCC)
- Jerry Murch (Tektronix)
- Erich Neuhold (Gesellschaft fuer Mathematik und Datenverarbeitung)
- Don Norman (Univeristy of California at San Diego)
- Gary Olson (University of Michigan)
- Judith Orasanu (ARI)

Participants from the University of Colorado:

- Bernie Bernstein
- Gerhard Fischer
- Peter Foltz
- Walter Kintsch
- Andreas Lemke
- Clayton Lewis
- Suzanne Mannes
- Helga Nieper-Lemke
- Martha Polson
- Peter Polson
- Christian Rathke
- Paul Smolensky
- Althea Turner

This workshop was supported by Grant No. MDA903–86–C0143 from the Army Research Institure.

TABLE OF CONTENTS

ACTION PLANNING: ROUTINE COMPUTING TASKS

Suzanne M. Mannes & Walter Kintsch
Department of Psychology and Institute of Cognitive Science
University of Colorado, Boulder

The tasks we are concerned with here are routine operations involving the file and mail system. Our subjects are experienced computer users, for whom such tasks present no challenge. However, solutions for these tasks cannot be precompiled, because while the elements of each task may be familiar, and indeed overlearned, they are often put together in novel sequences. In other words, we are not dealing with fixed scripts which only need to be retrieved from memory, but with plans for action that are generated apparently effortlessly in the context of each specific task. The question addressed here is how these plans are generated.

Theoretical Background

The problem statement for each task is a piece of discourse, usually a sentence or two with which the experimenter instructs the subject what to do. (More generally, one can assume that computer users instruct themselves in a similar way in the course of their activities). The theory of discourse understanding, developed in a quite different context by Kintsch & van Dijk (1978), van Dijk & Kintsch (1983), and Kintsch (1988), will be used to model the way in which subjects comprehend this discourse.

We are interested in how the instructional text activates the knowledge a subject has about the tasks involved, resulting in a well-formulated plan for action. We are not studying the actions themselves: this is probably best done within a GOMS- or CCT-like framework (Card, Moran, & Newell, 1983, Kieras & Polson, 1985). What we do here is generate the GOAL statement, which is required to activate the high-level productions in these models. Once you have a PLAN – in the sense discussed below – you can then activate a sequence of productions for executing that plan, in the manner described by these models.

Understanding an instructional text means coming up with an appropriate plan for action. The approach we take to modelling this process is analogous to our previous work on word arithmetic (Kintsch & Greeno, 1985). Understanding a word arithmetic problem means generating a mental representation of the sets and set relations involved, on the basis of which the correct computations can be performed. This is achieved by activating both relevant general world knowledge and specific knowledge about arithmetic. Analogously, in the present case, we need to activate knowledge about the general nature of the task we want to perform, as well as knowledge about the specific computer system on which we are working. Thus, the same general theory of comprehension is being applied to two different domains.

Protocol Analyses

Three experienced computer users were given the following three tasks with standard think-aloud instructions:

(a) *Include an address that you know into a file on the computer.*

(b) *You got a manuscript from someone and you want to edit it, making modifications, and then return it to the person you got it from.*

(c) *You got a message from someone while in mail asking for a paragraph out of a manuscript of yours to be sent to them.*

Their actions as well as their verbal protocols were recorded and analyzed by means of a method described in detail in Kintsch & Mannes (1987). The resulting temporal sequences of action planning statements and elaborations provided the starting point for our analyses. They gave us some idea about the sort of knowledge that is activated when subjects perform these tasks, as well as about the sequence of planned and actual actions involved. Table 1 shows a greatly abbreviated summary of the verbalizations and actions of Subject 1 performing Task b.

GET INTO MAIL	TO GET TO THE FILE
SAVE MESSAGE TO FILE	ASKS FOR FILE NAME
EXIT MAIL	
EDIT FILE	I'D BE IN EMACS WITH MANUSCRIPT FILE
EXIT EDITOR	
SEND MAIL MESSAGE	ASKS WHO TO SEND TO

Table 1. Abbreviated Protocol: Subject 1, Task b (REVISE).

We note the evidence for backward reasoning in Table 1, which is typical for these protocols. It is the anticipation of things to come which often drives current actions. In Table 1, the subject recognizes that in order to be able eventually to send the revised file, she must first enter the mail system to get the file†. The backward reasoning observed here contrasts with the forward reasoning which is characteristic for physics experts (e.g. Larkin, McDermott, Simon, & Simon, 1980), probably reflecting differences between the tightly constrained domain of physics problems and the relatively flat and amorphous problem space of the present task domain. Our theoretical goal will be to simulate the sequence of planned and actual actions, as observed in our data (but not the accompanying verbalizations).

The Model

The main components of the construction - integration model proposed by Kintsch (1988) are a propositional textbase derived from the instructional text, an associative long-term memory, a knowledge activation process by means of which the textbase propositions activate information from the long-term memory, and, following these construction phases, an integration process which selects what fits together, and deactivates what is contextually irrelevant. Furthermore, these processes are cyclic, because the text is processed in sentence-like units rather than as a whole, and because when some action is performed, the state of the world changes. Thus, processing is repeated until the desired change in the state of the world has been achieved, or until it fails.

---

† The verbal protocols provide other instances of this reasoning, e.g.:

First I'd send my method section of the paper. It's the section... I'd get into the editor one way or another and – first I'd have to look at the paper you are talking about. So I would have to think through how I stored that in my directory structure... So I'd have to think through how I have organized my stuff and go find the paper and then bring it up in the editor.

I've been using the UNIX mail system and I'd read your letter and then I wouldn't want to reply right away. I'd want to go up – you know – figure out what was going on.

Long-term Memory

Forty-two propositions were taken from the protocols of subjects performing the three tasks described above to simulate a portion of a user's long-term memory network. Interconnections in this net were computed by finding all the instances where a given proposition shared an argument with another proposition, and where it was embedded in another proposition. Additive connection values of .5 were used for each case of argument overlap or embedding. A LISP program computes the interconnection values among all the items in long-term memory, creating a matrix of size n x n, called the Connectivity Matrix. This matrix approximates an association matrix: since we do not know the actual associative strengths among the propositions in long-term memory, we use this crude and fallible, but relatively simple and objective method to estimate these values. The second portion of the simulated long-term memory consists of 15 PLANS, as shown in Table 2.

Each plan has three fields, a plan name, preconditions, and outcomes. Preconditions correspond to propositions designating states of the world which must be fulfilled in the environment for the plan to be executable. Outcomes correspond to propositions which become states of the environment as a consequence of executing a plan. Plan names are propositions, thus consisting of a predicate and a number of arguments. Sets of plans may be mutually exclusive (e.g. the four ways to send mail in Table 2). In such fields, the most strongly activated plan inhibits the other members.

Connections between the propositions in the long-term memory net and the plans are computed by the same algorithm based on argument overlap and embeddings, based upon the name field of the plans only. An exception must be made, however, for certain types of propositions which we call REQUESTS and OUTCOMES.

Plan Name	Preconditions	Outcomes
(COPY TXT FL)	(@SYS) (IN FL DIR)	(IN TXT FL) (IN FL DIR)
(COPY FL)	(@SYS) (IN FL DIR)	(IN FL DIR)
(READ FL)	(@SYS) (IN FL DIR)	(READ FL)
(EDIT FL)	(@SYS) (IN TXT FL) (IN FL DIR)	(IN TXT FL) (IN FL DIR)
(FIND FL)	(@SYS)	(IN FL DIR)
(SEND TXT FL ML)	(@ML) (IN TXT FL) (IN FL DIR) (IN MSG ML)	(RECV TXT)
(SEND TXT ML)	(@ML) (IN MSG ML)	(RECV TXT)
(SEND TXT FL SYS)	(@SYS) (IN TXT FL) (IN FL DIR)	(RECV TXT)
(SEND TXT SYS)	(@SYS)	(RECV TXT)
(COPY MSG)	(@ML) (IN MSG ML)	(IN TXT FL) (IN FL DIR)
(READ MSG)	(@ML) (IN MSG ML)	(READ MSG)
(FIND MSG)	(@ML)	(IN MSG ML)
(SAVE MSG)	(@SYS) (IN MSG ML)	(IN TXT FL) (IN FL DIR)
(ENTER SYS)	(@ML)	(@SYS)
(ENTER ML)	(@SYS)	(@ML)

Table 2. List of PLANS.

Requests are the imperative verbs that are used in problem statements (requesting the user to do something - edit, send, etc). The relationships between these verbs and plans has to be more precise than the one afforded by argument overlap: each particular request must be associated

directly with an appropriate plan. Thus, the request EDIT is tied to the plan (EDIT FILE) by a value of +1, and has 0 connection strengths with all other plans. Each outcome proposition (i.e. a proposition in the last column of Table 2) is similarly connected by a value of +1 to plans that produce this outcome, by a value of -1 to plans which produce an incompatible outcome but which include task relevant objects, and by a value of 0 to plans which produce irrelevant outcomes (because they deal with objects that are not involved in the task at hand). Only ultimate goals are connected in this way: other plans which need to fire before the ultimate plan can be accomplished must be contextually activated through the network as a whole.

Textbases and Situation Models

From the text of a problem (such as the three tasks above) a propositional textbase is derived by hand-coding, in the manner of Kintsch (1985). Each text proposition then activates two other propositions which are associated with it in the long-term memory net. This is done by computing the argument overlap and embedding relations between that propositions and all other propositions in long-term memory and normalizing this vector (so that the sum of all interconnections is 1).The resulting connection strength values can then be treated as probabilities, and two associates of each propositions can be sampled with replacement. Once again, request propositions require special treatment: they do not randomly activate associated knowledge, but must be used to retrieve an expected outcome. Specifically, a proposition of the form (REQUEST X(Y)) together with (OUTCOME $) is used as a joint retrieval cue to retrieve W(Z) - the outcome of doing X(Y) - which is associated to both retrieval cues, as in Raaijmakers & Shiffrin (1981) and Kintsch & Mannes (1987).

Thus, we now have text propositions, plus their associates and outcomes. To these we add all 15 plans: when trying to solve a problem of the type considered here, all plan-knowledge must at least have the potential to be activated.

The interconnections among the nodes in the resulting matrix are computed in the same way as in the original long-term memory matrix. There are, however, two further complications, concerning the relations between outcomes and plans, and among the plans themselves. Any outcome proposition that describes an already existing state inhibits plans that produce this outcome (with a value of -10). Thus, if the location of FILE-X is already known, the plan (FIND FILE-X) is inhibited. Expected outcomes of plans, on the other hand, are connected to other plans in the same way as in the long-term memory matrix, i.e. they activate plans that produce these outcomes, and inhibit plans that produce incompatible outcomes.

Plans themselves are interconnected by a causal chaining mechanism. That is, a plan that requires condition X activates all plans that have X as an outcome. Thus, the interconnections among plans represent the user's knowledge about the causal relations in the system.

In this manner, a connectivity matrix of size n x n is obtained for each text, corresponding to the original m text propositions and to the (n-m) associates, outcomes, and plans that have been added to the text. An initial activation vector with n elements is then constructed, with activation values 1/m for the m original text propositions and 0 for everything else. This vector is then repeatedly post-multiplied with the connectivity matrix, and renormalized after each multiplication, to compute the spread of activation among the elements of the vector. When the activation vector reaches an arbitrary criterion, such as an average change after a multiplication of less than 0.0001, the activation pattern is considered to reflect the stable situation model formed by the user, given the particular task and the knowledge assumed here.

In terms of the van Dijk & Kintsch (1983) model, we have started with a small *textbase*, which activated both relevant and irrelevant *knowledge*. We then used an *integration* process to get rid of the irrelevant knowledge that had been introduced, thus arriving at a *situation model* that corresponds to the user's understanding of the situation and the task to be performed.

Plans and Action

Plans end up at different levels of activation in the situation model. We postulate an executive process that checks the plan with the highest activation level and executes it if its conditions are satisfied. If they are not, it goes to the next highly activated plan, and so on. If the plan executed has as its outcome the expected outcome (which was retrieved from the long-term memory by means of the original request), the problem is solved. If not, the consequences of the plan that was executed are added to the situation model, and the integration process is restarted, resulting in a new pattern of plan activation. This process is repeated until the desired outcome results (or is stopped when it goes hopelessly awry).

Three Examples

The results of a simulation with this model of the three problems shown above are partially depicted in Table 3. Only the plans actually considered by the executive process are shown, together with rankings indicating their relative activation values. For the *INCLUDE* problem, the (EDIT FILE-LETTER) plan receives the highest activation value in the first round. It cannot be executed because one of its preconditions is not fulfilled: the location of FILE-LETTER is unknown. Therefore, the second-ranked plan is considered. This happens to be (FIND FILE-LETTER), it is executed, and results in the the addition of the proposition (KNOW FILE-LETTER, LOCATION) to the situation model. After a new integration phase, the (EDIT FILE-LETTER) plan again is the most highly activated one. It is now possible to execute this plan, which provides the desired outcome (IN TEXT-ADDRESS, FILE-LETTER), completing the problem.

The *REVISE* example is segmented into two components by the linguistic cue *and then*: such temporal connectives tend to be used as signals of episode boundaries, and hence as segmentation cues (Kintsch & Mannes, 1987). The request which is acted upon first is to *revise the manuscript*: the models wants to EDIT, but cannot do it, so its executes the next highest plan, which results in a new proposition characterizing the state of the system, (KNOW FILE-MANUSCRIPT, LOCATION). After a new integration process, EDIT is still at the top, but now it is possible to execute this plan. As a result, the propositions (IN TEXT-MANUSCRIPT, FILE-MANUSCRIPT) & (KNOW FILE-MANUSCRIPT, LOCATION) get added to the textbase, the request to revise is dropped, and the request to *send* it is substituted. The integration process selects the plan (SEND TEXT-MANUSCRIPT, FILE-MANUSCRIPT, @SYSTEM), and the problem is solved.

The *SEND* problem is somewhat more complex and requires five construction-integration cycles before the solution is arrived at. From the very beginning, the model wants to SEND, but is unable to do so. All it can do in the first cycle is to leave the MAIL system. In the next cycle, it manages to locate the file it needs, so that it can copy the to be sent text into a FILE-TEXT. However, in order to send this file, it needs first to re-enter the mail system, because it prefers to reply to the message it received, rather than sending a new one.

Case-based Reasoning

In the simulations described above, the program used only its general knowledge about plans in arriving at solutions. What would happen, however, if these problems, or similar ones, were presented for a second time? Although we don't have any direct empirical evidence as yet, we would expect to see examples of case-based reasoning. It appears plausible that the to-be-comprehended text would remind subjects of having solved this problem, or a similar one, before.

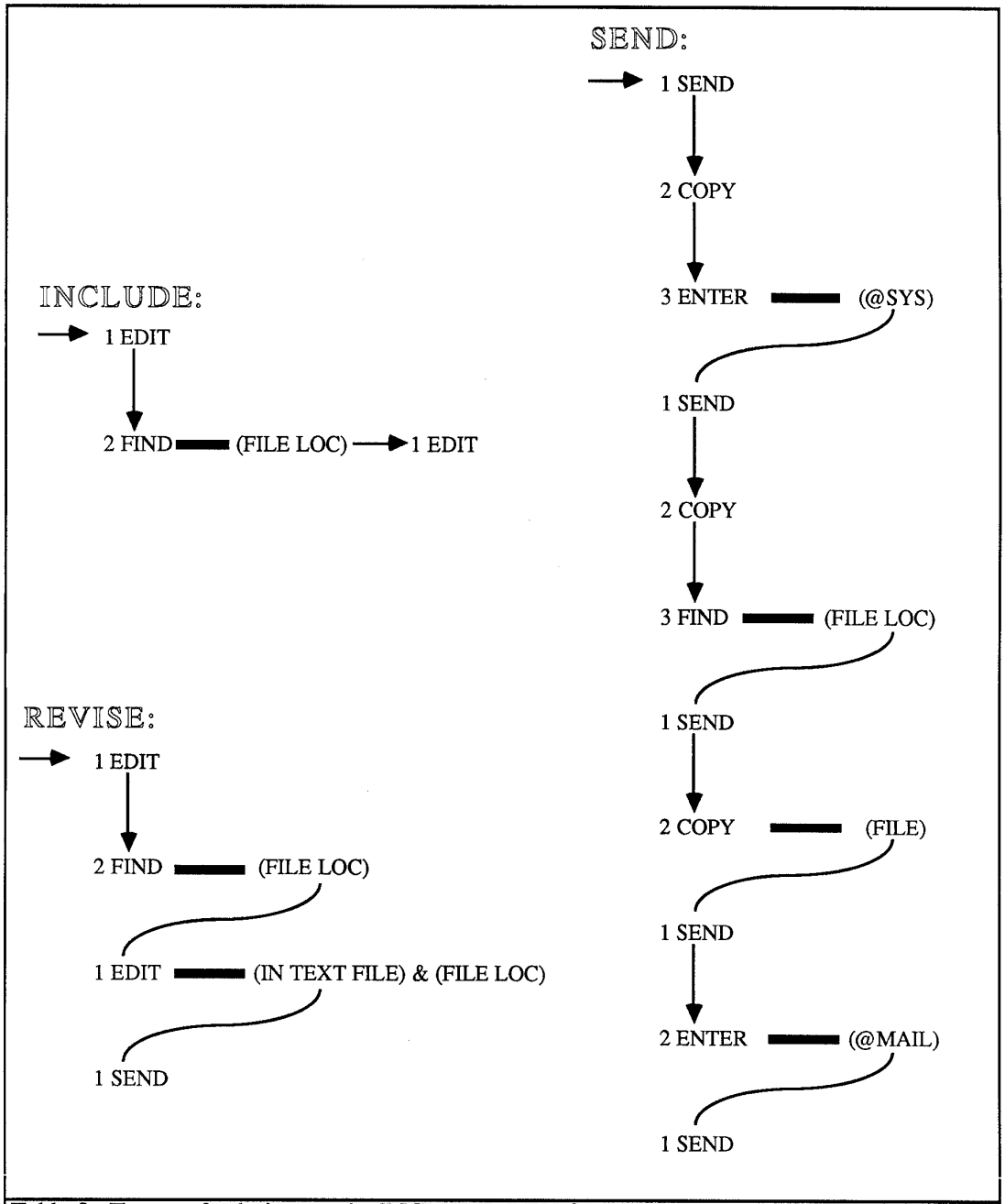


Table 3. Traces of solutions to the INCLUDE, REVISE, and SEND tasks.

There are many ways in which a case-based reasoning capability could be added to the present simulation. As an initial exploration, we decided to make some simple assumptions about memory for an episode. Specifically, we assumed that what would be remembered from an episode would be the three most highly activated text propositions, together with the plans that were executed in performing the task. Thus, four cases, each consisting of the three propositions and the corresponding plans, were added to our long-term memory. (There were four cases, because the Revise-Problem consisted of two episodes). The three problems were

then re-run. In each instance, the text propositions activated the earlier case, and sometimes even other cases. Thus, the Revise text reminded the system not only of the earlier Revise episodes, but also of the Send episode, which also involved a text and file called *manuscript.* In the comprehension network, each case was connected only to the plans that were actually executed in this episode. Not surprisingly, this had the effect of further strengthening these plans, and all problems were solved correctly, as before. Substantial portions of the activation of each plan, however, derived from its connection to earlier episodes: between 40% and 52% of the total activation of non-ultimate plans (i.e. plans that had to be executed before the ultimate goal could be achieved) derived from remindings of previous cases. For ultimate plans, this proportion was lower, ranging between 20% and 29%. Thus, cases helped particularly in the early part of the comprehension process, while the last step is well defined by the text anyway.

Conclusions and Further Developments

The model's ability to simulate the comprehension processes for the three tasks considered here is promising, though of course hardly conclusive. We need to expand the size of our long-term memory, the number of plan alternatives, and the range of problems considered, and see how the present model will perform. We need to explore more systematically than we have done so far the parameter values used in this simulation. (Basically, we chose the first combination of values that worked for the present simulation). We need to explore various ways of making our simulation more elegant and efficient (e.g. it may not be necessary to introduce a separate executive component to check whether plans are executable, etc.). Equally importantly, we need to find a way in which the details of the model predictions can be evaluated against empirical protocols. All we have done so far, is to note a generally satisfactory overall correspondence.

In spite of the fact that this work is as yet quite incomplete, our progress so far warrants some optimism. From the standpoint of problem solving work, what we have done is not spectacular: we do a means-end, backward problem solving that probably would not have challenged the General Problem Solver. However, we do this within a framework of a general theory of discourse comprehension. We are not proposing a model of problem solving in the computing domain *de novo* but merely adapt a comprehension model to this domain by representing the domain knowledge as plans. Our description of problem solving in this domain as basically a comprehension process - more perception-like than problem solving in the traditional sense - has great intuitive appeal. We of course do not doubt that problem solving *per se*, in the sense of conscious consideration of alternatives and decisions among them, plays a role in these tasks - but that happens when the basic, automatic comprehension processes we are describing here fail. Most of the time, they will suffice to handle the simple, routine problems we are concerned with here.

Should we be able to work out and test more fully the model sketched here, we plan to use the model to explore the question how novice users who do not have the kind of knowledge upon which behavior is based in our simulations can be helped to perform routine computing tasks.

References

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

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

Kintsch, W. (1985). Text processing: A psychological model. In T. A. van Dijk (Ed.) *Handbook of Discourse Analysis*, Vol. 2 (pp. 231-244). London: Academic Press.

Kintsch, W. (1988). The use of knowledge in discourse processing: A construction-integration model. *Psychological Review*, in press.

Kintsch, W., & van Dijk, T. A. (1978). Toward a model of text comprehension and production. *Psychological Review*, **85**, 363–394.

Kintsch, W., & Greeno, J. G. (1985). Understanding and solving word arithmetic problems. *Psychological Review*, **92**, 109-129.

Kintsch, W., & Mannes, S. M. (1987). Generating scripts from memory. In E. vanderMeer & J. Hoffmann (Eds.). *Knowledge aided information processing* (pp. 61-80). Amsterdam: North-Holland.

Larkin, J. H., McDermott, J., Simon, D. P., & Simon, H. A. (1980). Models of competence in solving physics problems. *Cognitive Science*, **4**, 317-345.

Raaijmakers, J. G. & Shiffrin, R. M. (1981). Search of associative memory. *Psychological Review*, **88**, 93-134.

van Dijk, T. A., & Kintsch, W. (1983). *Strategies of discourse comprehension.* New York: Academic Press.

AN EMPIRICAL EVALUATION OF RETRIEVAL BY REFORMULATION ON HELGON

Peter W. Foltz and Walter Kintsch
Department of Psychology and Institute of Cognitive Science
University of Colorado, Boulder

Description of HELGON and Retrieval by Reformulation

Retrieval by Reformulation

Retrieval by reformulation is a method of extracting information from a database. Using retrieval by reformulation, a user interactively refines partial descriptions of his or her target item by criticizing successive examples. To do this, a user initially makes a query by constructing a partial description of the item in the database for which he or she is searching. The system then produces a description of an example instance from the database which matches the user's partial description. The user can then refine the query by choosing any attributes shown in the example instance and incorporating those descriptions, or variations of them, into the partial query, thereby reformulating the initial query. This process of reformulating the query continues until the desired information is retrieved.

Retrieval by reformulation is based on psychological theory of human memory retrieval in which people retrieve information from human memory through an iterative process of constructing partial descriptions of what they want to retrieve (e.g. Norman and Bobrow, 1979). The RABBIT system (Williams, 1984; Tou et al., 1982), implements this procedure in a hierarchical database retrieval system. The advantages of such a system include that users do not need to have a complete concept of what they want to retrieve, but can interactively refine their query until they find what they need. ARGON (Patel-Schneider, Brachman, and Levesque, 1984) is a similar information retrieval system based on the RABBIT system. It uses a frame based knowledge representation language to implement the database.

HELGON

HELGON (Nieper, 1987) is an extension of ARGON incorporating several changes to facilitate usability. These changes include a graphical display of the frame hierarchy of information. The frame hierarchy is organized as a tree with the actual information in the database stored under the subnodes of categories. A sample tree for the HELGON system is shown in Figure 1. This display permits users to see how the information is organized in the database thereby facilitating navigation through information.

An evaluation of HELGON was done in order to determine what types of difficulties users would encounter when using query by reformulation.

Method

Subjects

Four subjects, all without prior experience with HELGON implemented on a Symbolics, were used in the study. One of the subjects was an experienced Symbolics user, while the other

three were not familiar with the Symbolics, but were experienced using similar types of systems.

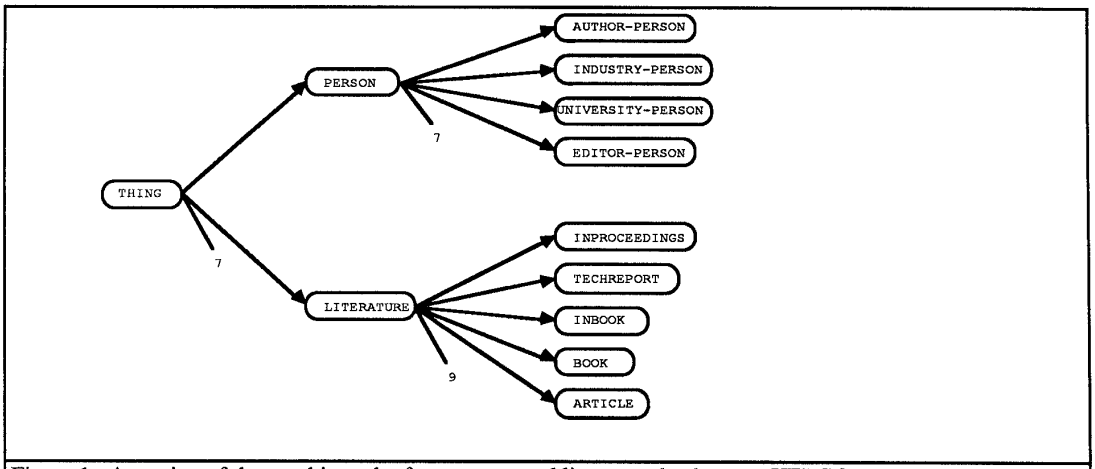


Figure 1. A portion of the tree hierarchy for a person and literature database on HELGON.

Tasks

Seven tasks that would be considered typical HELGON retrieval tasks were used for the evaluation. All tasks involved retrieving information about people or literature that were contained in a database. An example tasks is:

> Task 5: You are looking for a paper which has the word LISP in its title. You happen to know that it was published in the proceedings of a conference you attended. You know that it was not in 1984 because you didn't go to any conferences that year, and it was not in 1987 because you would remember that.

Procedure

Subjects were given minimal instructions on how to use the HELGON system, but some information about the theory of how to perform retrieval by reformulation. Subjects were given the seven tasks which were ordered from easier tasks to the more complex tasks. Before attempting each task, subjects had to provide a description of how they planned to complete the task. While the subjects worked on the tasks, they had to provide verbal protocols and were videotaped.

Results

The performance of the subjects was compared to that of an experienced HELGON user. This permitted a comparison of the commands chosen and the proper sequence of those commands to perform each task.

Overall Error Rates

Subjects learned to complete the tasks on HELGON fairly quickly. Overall, 92 percent of the tasks were solved correctly. Nevertheless, only 25 percent of the tasks were solved without errors. The average number of errors per task was 2.8, with a range from 0 to 12 errors per task.

Categorization of the Errors

A task analysis of each subject's performance on each task was done. The analysis included both the steps the subject performed, based on the videotape, and the subject's hypotheses, based on the verbal protocol as best as possible. This analysis made it possible to see the errors the subject made and the hypotheses that may have caused the subject to err. The performance for each subject on each task was graphed showing the steps the subject took and the subject's hypotheses as he/she performed the steps. A sample graph from the task analysis for a subject on task 5 is shown in Figure 2.

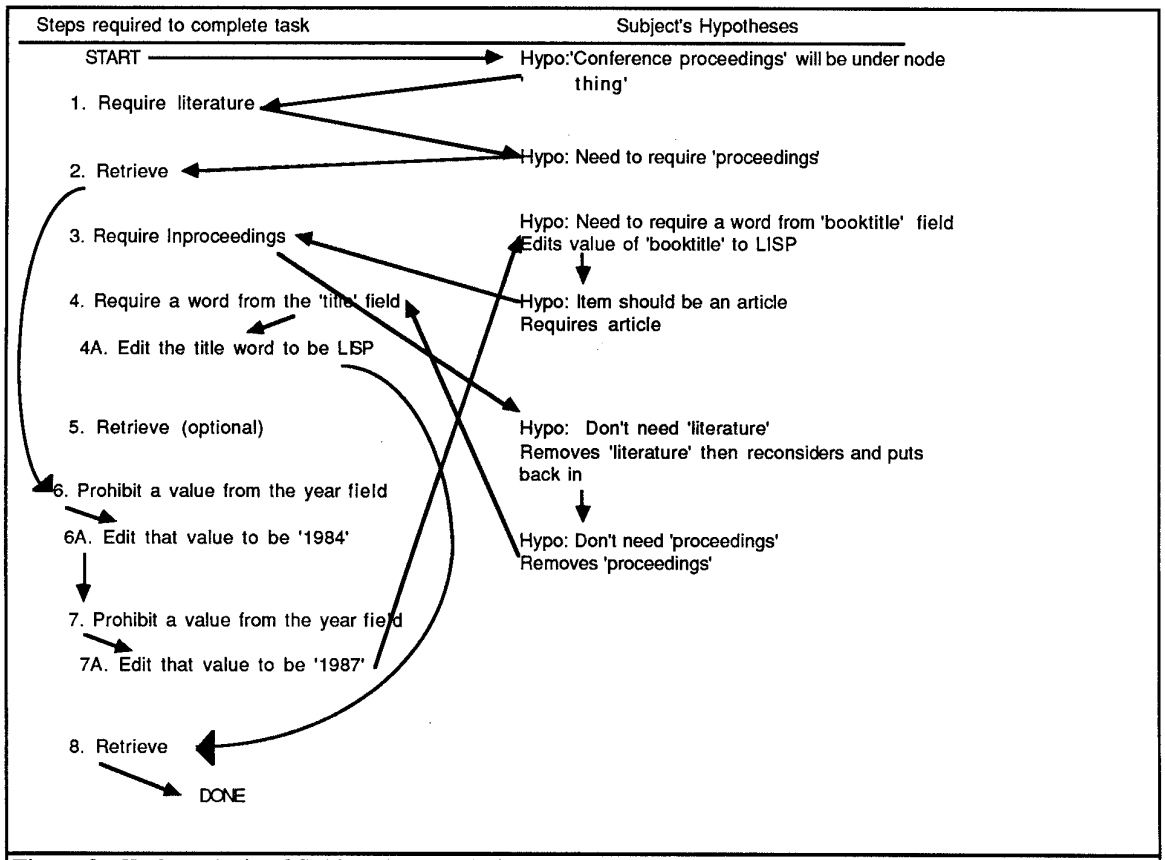


Figure 2. Task analysis of Subject 4 on Task 5.

The errors made by the subjects on HELGON can be roughly categorized into two classes; errors due to misunderstanding the procedures to use HELGON, and errors due to misunderstanding the contents of the database. The errors due to misunderstanding the contents of the database can be further subdivided into two classes; errors due to misusing the terms in the database, and errors due to misunderstanding the hierarchical structure of the database. A count of the errors showed that 43 percent of all errors were due procedure misunderstanding, 18 percent of all errors were due to misuse of database terms, and 39 percent of all errors were due to misunderstanding the hierarchy of the database.

For procedure misunderstanding errors, subjects were not fully aware of how to use the system to retrieve the desired information. These errors encompassed such things as choosing wrong items or menus and directly typing commands and retrieval cues rather than choosing them off the screen. The subjects had been given very little information about how to use the system, so a good portion of time was spent using a trial and error method of determining the proper interaction techniques. Often subjects knew exactly what they wanted to retrieve, but

were unsure about how to go about retrieving the information on the system. Based on their pre-task protocols, they often fully understood the retrieval by reformulation process for a task and had a fairly well constrained retrieval cue that they thought ought to be successful. Because of this, they would attempt a direct retrieval strategy of entering their retrieval cue instead of attempting to modify an exemplar.

A common problem with database retrieval systems is that users are not aware of the names of the database fields which they must specify in order to perform a retrieval. The HELGON system does avoid this problem, since through the process of retrieval by reformulation, the system presents examples of matching instances with their associated fields. This permits users to select the field directly from the matching instances. Nevertheless, there is still a problem of precision of application of terms for field names in the database. Subjects often confused the field name *proceedings* (which referred to a proceedings from a conference) with the field name *inproceedings* (which referred to an article within a proceedings). Landauer, Galotti and Hartwell (1983) showed that the precision of application of a term must be very well constrained and, in the case of editing commands, novices and system designers have different views on what are good terms for commands. Thus, a single term for each field does not provide a good enough idea for the user of what information is contained within that field.

While HELGON does present a graphical representation of the database hierarchy, the 39 percent of the errors being hierarchy errors shows that there were still a lot misunderstandings of the hierarchical structure of the database. Subjects often searched for nodes or fields down the wrong branch of the hierarchy or tried to use two nodes that were mutually exclusive. An example of this type of error occurred when a subject who, trying to find the address of the author of *The Architecture of Cognition,* used the retrieval cues of both *person* and *literature.* Since no item in the database could be both a person and a piece of literature, no information was retrieved. Instead of staying within the confines of the hierarchy, the subject tried to use two cues that were highly associated for the task in order to retrieve the relevant information.

This type of retrieval attempt suggests that subjects used retrieval cues that would be similar to those used in human semantic memory. Research on semantic memory retrieval, starting with Collins and Quillian (1969) has shown that semantic memory does not have a hierarchical organization, but one in which there are rich cross-classifications of information. Thus, it is organized in an associative structure rather than in an ISA hierarchy as in HELGON. The use of a retrieval cue in a retrieval attempt is also similar to the memory retrieval model of Raaijmakers and Shiffrin (1981). In this model, retrieval cues are used as probes operating on an associative memory structure. The probes are then compared with each item in memory to determine the strength of relationship between the probes and each item. The errors that occurred from retrieval cues selected by the subjects, may be due to the subjects trying to use the same type of retrieval cues used in their own retrieval processes.

Conclusions and Extensions to HELGON project

Conclusions

While using HELGON subjects appeared to have the greatest problems in three areas: the direct retrieval of information when they have a constrained retrieval cue, the use of the database terms, and the use of the database hierarchy.

For the direct retrieval problem, subjects should be able to perform a direct retrieval as soon as they have a well constrained retrieval cue. The RABBIT-HELGON strategy of modifying exemplars is used when no plausible retrieval cue is available. It is an extremely powerful strategy, and the heart of the system that differentiates it from other retrieval systems, but it should not be the only strategy.

The problem of selecting precise and appropriate terms for database fields is a familiar problem in human-computer interaction (e.g. Furnas, Landauer, Gomez & Dumais, 1987). Furnas et al. suggest a solution of "unlimited aliasing" in which many alternative access words would be available for each item. This would raise the probability that users would be able to access the correct term when making retrieval attempts.

The difficulty subjects encountered with the database hierarchy suggests that subjects may be using associative retrieval cues to perform their retrieval tasks. A possible solution to this problem is to design a retrieval system that works in the same way as human retrieval.

A Human-Retrieval-Like HELGON simulation

One extension to the HELGON project is that we are developing a simulation of the way subjects actually tried to retrieve information in HELGON. The simulation incorporates an associative retrieval mechanism similar to the Raaijmakers and Shiffrin model. This simulation will thus incorporate two facets of human retrieval; the retrieval by reformulation process, as currently used in HELGON, and an associative retrieval mechanism. This should provide clues to how users conceptualize the retrieval process when using a database retrieval system. We expect that this simulation will then provide guidelines for the redesign of HELGON.

Additional Subjects

A second extension to the project is to run additional subjects on HELGON. While the types and numbers of errors are consistent across subjects for the four subjects, twice the number of subjects are being run to check on the reliability of the results obtained.

References

Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior*, **8**, 240–248.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Communications of the ACM*, **30**, 11, 964–971.

Landauer, T. K., Galotti, K., & Hartwell, S. (1983). Natural command names and initial learning: A study of text editing terms. *Communications of the ACM*, **26**, 7, 495–503.

Nieper, H. (1987). Information retrieval by reformulation: From ARGON to HELGON. In G. Fischer & H. Nieper (Eds.) *Personalized Intelligent Information Systems* (Chapter 19). Technical Report 87-9, Institute of Cognitive Science, University of Colorado, Boulder, May 1987.

Norman, D. A., & Bobrow, D. G. (1979). Descriptions: An intermediate stage in memory retrieval. *Cognitive Psychology*,**11**, 107–123.

Patel-Schneider, P. F., Brachman, R. J., & Levesque, H. J. (1984). ARGON: Knowledge representation meets information retrieval. Fairchild Technical Report 654, Schlumberger Palo Alto Research, September 1984.

Raaijmakers, J. G. W., & Shiffrin, R. M. (1981). Search of associative memory. *Psychological Review*, **88**,2, 93–134.

Tou, F., Williams, M. D., Fifkes, R. E., Henderson , D. A., & Malone, T. (1982). RABBIT: an intelligent database assistant. *Proceedings of the AAAI conference*, Pittsburgh, Pennsylvania, August, 1982.

Williams, M. D. (1984). What makes RABBIT run? *International Journal of Man-Machine Studies*, **21**, 333–352.

MENTAL MODELS (MM) – A COMPUTER SCIENTIST'S POINT OF VIEW

Gerhard Fischer
Department of Computer Science and Institute of Cognitive Science
University of Colorado, Boulder

Summary

In our research we are interested in the design and understanding of high functionality computer systems (HFCS). These systems contain thousands of objects (e.g., functions, classes, operations, ...) and they are not completely mastered by users. How can mental models (MM) be used to help users to take greater advantage of such systems?

HFCS are *designed* systems (i.e., "artifacts"). To use designed systems, it is more important to understand the goals, the functions and the adaptive capabilities which they can be used for or associated with than their internal structure. One important design criteria for such systems should be: is the acquisition of a MM of them possible or at least facilitated (e.g., by constructing these systems with a layered architecture)?

In order to enhance and/or reduce the learning process of HFCS and to allow users to achieve their goals with them, we have constructed a number of intelligent support systems. These support systems should enhance the acquisition of a MM and they should reduce the amount of knowledge which needs to be accumulated in the MM to operate the systems effectively.

We are pursuing a number of methodologies to break the "conservation law of complexity" (i.e., the description of a complex system does not need to be equally complex). Many of these methodologies (e.g., differential descriptions of a systems with respect to another system, custom-tailored views of reduced complexity, custom-tailored explanations in the situation model of the user) require that the system has some understanding of the MM of an individual user. Which mechanisms are required for systems to get some conceptualization of different MM and to be able to acquire this information from the user?

Our goal (as Computer Scientists) to develop design principles for comprehensible systems raises a lot of interesting questions about MM – and we are convinced that answers to these questions will help us to build more user-centered computer systems.

Mental Models

Mental Models – Some Definitions

During the workshop the participants spent considerable time trying to come up with a precise definition for a MM[†]. The group did not succeed in coming up with a single agreed-upon definition – but we defined a number of useful terms such as "How-to-do-it-knowledge",

---

[†] An similar observation was made by another workshop (see page 6 in (Carroll, Olson & Anderson, 1987)): *["Much of the discussion at the workshop centered on sifting through the many definitions of the term 'mental model'"].*

"How-it-works-knowledge" and "How-it-behaves-knowledge." We should not be bothered by the fact that the term MM is not precisely defined. It shares this feature with other interesting notions such as intelligence, understanding, creativity, etc.. Despite this lack of a precise definition, these terms have proven to be useful.

Here are a number of definitions for MM from different sources:

- *"Some kind of understanding of how the device works in terms of its internal structure and processes (Kieras & Bovair, 1984)."*
- *"Knowledge of how the system works, what its components are, how they are related, what the internal processes are, and how they affect the components. It is this conceptualization that allows the user not only to construct actions for novel tasks but also to explain why a particular action produces the result is does (Carroll, Olson, & Anderson, 1987)."*
- *"People form internal, mental models of themselves and of the things and people with whom they interact. These models provide predictive and explanatory power for understanding the interaction (Norman, 1986)."*

For our own approach, the following working definition seems to be useful: *"Mental models are individuals' organized body of knowledge (including beliefs) for a domain."* The emphasis in this definition is on "organized body"; the organization is imposed by our goals and by the structure of the specific domain knowledge. We will try to illustrate our notion by comparing it with other kind of models in the next section (similar distinctions have been made by Norman, (1982).

Relationship of Mental Models to Other Kind of Models

In our research we have tried to understand the implications of *usage patterns of complex systems* as shown in Figure 1.

The different domains correspond to the following:

D_1: the subset of concepts (and their associated commands) that the users know and use without any problems.

D_2: the subset of concepts which they use only occasionally. Users do not know details about them and they are not too sure about their effects. Descriptions of commands (e.g., in the form of property sheets), explanations, illustrations, and safeguards (e.g., UNDOs) are important so that the user can gradually master this domain.

D_3: the mental model of the user, i.e., the set of concepts which he/she thinks exist in the the user) is necessary for users to communicate their plans and intentions to the system†

D_4: represents the actual system. *Passive help systems* are of little use for the subset of D_4 which is not contained in D_3, because the user does not know about the existence of these system features. *Active help systems* and *Critics* which advise and guide a user similar to a knowledgeable colleague or assistant are required so that users can incrementally extend their knowledge to cover D_4. One question is: where and how is

† The relevance of MM in our lives in general is illustrated in the writings of Oscar Wilde ("The Portrait of Dorian Gray") and Max Frisch ("Diary"), who show that in many situations, models overwrite reality and the *perceived reality* dominates.

D_4 documented? Through the code? Through documentation (which is in almost all cases neither complete nor up-to-date)? In the heads of the design team?

D_3, the mental model of the user (called "user model" by Norman (Norman, 1986)), is one instance of a class of systems which may be called the *conceptual models*, C_m, of the actual system, D_4. D_3 is a naturally evolving model, acquired through observation, instruction, inference and exploration. There are many of them to fit different situations. They are constrained by users' technological backgrounds, previous experiences, and by the structure of the human information processing system. Mental models are also held by the designer (called "designer model" by Norman (Norman, 1986)).

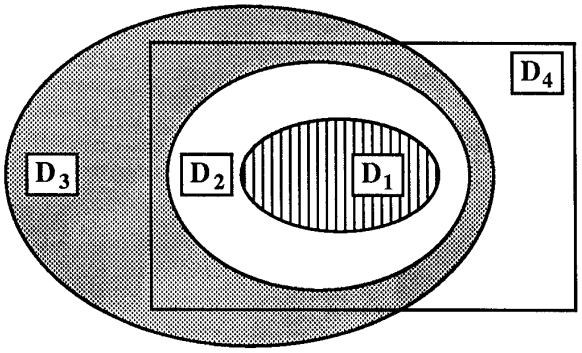


Figure 1. Levels of System Usage.

Conceptual models of systems are invented by teachers, designers, scientists and engineers to assist them in their understanding of a system. The boundaries of conceptual models are not clearly defined: should the conceptual model include *situation models* (i.e. models how users might think about the system) or is it only an conceptualization of the system (i.e. a system model) (Fischer & Kintsch, 1986).

There are a number of other models which deserve further investigation:

- *the scientist's conceptualization $C_S(D_3)$* of the mental model of a user (the question here is what kind of experiments can be conducted to characterize mental models adequately?);
- *the designer's model of the (anticipated) user population* (this model will impact whether a system will support different users or just the "generalized typical user");
- *the system's model of the user* (needed to acquaint the user with system functionality in the part of D_4 which is not contained in D_3).

A last interesting aspect to be mentioned here is the role of *mental models in sports*. How successful can we be learning to ski or to play tennis by lying on a sofa and doing it in our heads? Mental training has been found useful in sports and some people attribute a very important role to it (see the books about "Inner Tennis" and "Inner Skiing" by Timothy Gallwey). Sports also clearly indicates the different errors which can be made in doing something, namely *specification errors* (where the mental model of the task is wrong) and *implementation errors* (where the mental model of the task is correct but its execution is wrong). Papert (Papert, 1980) argues that "people need more structured ways to talk and think

about the learning of skills. Contemporary language is not sufficiently rich in this domain." He concludes that programming is an important source for a new class of descriptive devices which serve as a means of strengthening languages. This descriptive power is needed to describe and articulate general conceptual models as the basis for individual mental models.

Mental Models and the Sciences of the Artificial

We believe that there is a crucial difference between MM for the Natural Sciences (dealing with *natural systems*) compared to the Sciences of the Artificial (Simon, 1981) (dealing with *designed systems*).

In the Natural Sciences, MM have to describe existing phenomena. We cannot change physics to make it easier for the user to construct a MM. The MM has to be based on reality; to achieve a high degree of "cognitive comprehensibility", simplification may play a crucial role (e.g., worlds without air resistance and worlds for which Newtonian physics gives an adequate description).

Computer systems (especially HFCS) belong to the Sciences of the Artificial. We claim that one of the major design requirements for a system should be that it should be designed in a way that it is easy to construct a MM for a system. This requirement leads to many consequences, e.g., to provide tools to facilitate a mental model, to simplify descriptions by introducing layers of abstractions (see Figure 2), and to represent programs in order to increase their "cognitive comprehensibility" (e.g., the LISP-CRITIC (Fischer, 1987a) has a few hundreds rules which transform an arbitrary LISP program into a different representation which is cognitively more efficient).

Brown (Brown, 1986) argues for the separation of *physical fidelity* from *cognitive fidelity*, in order to recognize that an <accurate> rendition of the system's inner workings does not necessarily provide the best resource for constructing a clear mental picture of its central abstractions. In the Natural Sciences, we (as designers) can only change the cognitive fidelity of a system, whereas in the Sciences of the Artificial, we have the freedom to shape the "physical fidelity" (as a product of our design) according to our design requirements.

At the workshop, there was an extensive discussion about a *coffeemaker* (an example introduced by Don Norman). The critical question was: what is a good conceptual model for a coffeemaker? Is it good enough to achieve our goal: to make coffee – without any need to know the internal workings of it? Most users (I claim) do not care about the internal mechanisms, they want to achieve their goals. So what happens if the coffeemaker breaks down (i.e., when our experiential model is insufficient)? Well, as far as I am concerned, I would bring it back to get it fixed or I would buy a new one.

I believe that this example can be interpreted from a debate in knowledge-based systems – namely the differentiation between "shallow" and "deep" expert systems[†] . Obviously, it would be nice to have all kinds of reasoning methods in expert systems; but the successful use of shallow expert systems has shown that we can get (at least in certain domains) quite far with them. There may be another useful comparison to knowledge-based systems: the structural view of a system is usually expressed with frame-based, object-oriented systems whereas the experiential view is usually expressed with rule-based systems.

Breckenridge 1987 and Mental Models

The Breckenridge workshop in 1987 about "Personalized Intelligent Information Systems" (which preceded the one about mental models) was less focused. Both workshops had the

---

[†] It is also related to different educational strategies: should we teach case studies or general principles?

ultimate goal to increase our understanding of user-centered systems. A number of issues were raised in the 1987 workshop which are directly relevant to the mental model workshop (for details the reader should consult Fischer & Nieper (1987)). To justify the existence of HFCS, it was observed that "reality is not user-friendly", or in other words: there will always be a need for complex systems. But it was also noted that there is no "conservation law of complexity." The issues enumerated to reduce complexity are in many ways relevant to mental model research (and several of them will be discussed further in the next chapter):

- exploiting what people already know (one way to do so is supporting "human problem-domain communication" (Fischer & Lemke, 1988));
- using familiar representations, based on previous knowledge and analogous to known situations (this can be done by supporting differential descriptions, which requires that we relate the presentation of new information to the information contained in the mental models of individual users);
- exploiting the strengths of human information processing, e.g., the power of our visual system (this requires visualization techniques to overcome the opacity of many computer systems);
- segmenting information into microworlds (by supporting the architecture of "Increasingly Complex Microworlds" which has the potential to increase the cognitive fidelity of a system);
- eliminating learning through "better" systems and intelligent support systems (where "better" could be defined to facilitate the acquisition of a mental model).

A important discussion centered around the topic: "Do we want to be explicit all the time (i.e. put knowledge in the world)"? This issue seen from a mental model perspective implies that people do not necessarily want to communicate their mental models and make them explicit. There may be different reasons why this is the case: people may not have MM which they can communicate, they may not know how to communicate them, the effort to communicate them is too large or they just want to keep the knowledge to themselves.

The Role of Mental Models in the Design of Comprehensible Systems

Limitations of Previous Research

Previous research about mental models has been oriented mostly towards rather simple systems (i.e., systems requiring operation only or systems which could be learned in a few hours or less (Kieras & Bovair, 1984)).

Our main research interest is how people understand and successfully use HFCS. Norman Norman (1986) claims that MM are easy to construct for single tasks, used by one set of users (e.g., specialized tools such as spreadsheets) whereas MM are difficult (perhaps impossible) to construct for general purpose systems with a open-ended set of users and power[†].

With HFCS we encounter a dilemma: on the one hand, these are systems where good MM are most urgently needed – but on the other hand, it is less clear which conceptual models could be developed for these systems so users would be able to build MM for them. It is quite obvious

---

[†] One may argue that systems which do not allow users to construct a MM for them should not be constructed.

(see Figure 1) that MM for HFCS cannot be complete. MM cannot be deduced merely from experience – because there are just too many experiences to go through. Therefore future research efforts should be oriented not towards perfect models, but towards models which are "good enough." Because the learning of these systems is an incremental, indefinite process, we have to understand how MM operate and evolve in naturalistic settings over long periods of time. Another question is: can a good MM help us to keep D_1 (see Figure 1) fairly small and extend it on demand (Fischer, 1987b)?

Mental Models and our Research

Better Prescriptive Conceptual Models

Conceptual models are invented and developed by designers, teachers, psychologists and engineers. The quality of these models may have a great impact on how easy it is for users to develop their own mental model and understand something. For example, *recursion* has for long time been considered a difficult concept to master. Research and experiences (primarily in connection with LOGO projects (Papert, 1980)) has shown that the existence of a good model to think about recursion (i.e. the "little men model") has enabled children to successfully use recursion in their programming.

Increasing the Cognitive Comprehensibility of Systems

Informational systems are often opaque. But computer systems offer substantial power to make the *invisible visible*†. Visualization techniques (such as our "software oscilloscope" (Boecker, Fischer, & Nieper, 1986)) provide (automatically generated) representations and cues towards the understanding of data and control structures. In other systems, we have reduced memory requirements for the user (e.g., the CATALOG in the WLISP system and the property sheet representations of commands such as in WLISPRC (Fischer & Lemke,1988))†. Debugging systems can be understood from this perspective, in that they offer a progressive disclosure of a system's internal working.

Norman (Norman, 1988) distinguishes between "knowledge in the world" and "knowledge in the head." We will equate "knowledge in the head" with MM. One of the major challenges we see for innovative design of computer systems is to *redraw the borderline* between the two. Many techniques should be oriented towards putting more "knowledge in the world" so we have to keep less "knowledge in our head." Examples are:

- in design artifacts: do not only retain the product, but also the process;
- make opaque systems observable and inspectable (through visualization, explanation, and problem-domain semantics);
- the interface of the STAR and the Macintosh reduced the demands on MM by making more information in the external world available.

But putting more knowledge into the world will not solve all our problems by itself. Norman states: *"Whenever information needed to do a task is easily available in the world, the need for*

---

† The real challenge is to make not everything visible, but to make the information visible which is of importance for understanding a system and for the goals of a specific user.

† `Append` versus `nconc` Boecker, et. al., 1986, for example, shows that designers of systems have different options to reduce the demand on a mental models: they can omit `nconc` (as it is the case in LOGO), they can provide an explanation in the conceptual framework of copying versus destructive functions; or they can provide a support tool such as KAESTLE to visualize it.

a person to learn it diminishes". But how do pilots cope with 450 alert instruments on a BOEING 747 (Chambers & Nagel, 1985)? Even if we put knowledge in the world, it can be represented in many different ways. And unless we have the right kind of media, our ability to increase the cognitive comprehensibility may be limited (e.g., unless we have dynamic media, we have problems illustrating dynamic processes). Visualizations by themselves are not necessarily easier to understand: visual and form-based representations (just like languages) require a shared understanding (i.e. the right kind of background knowledge on the side of the observer).

Support for the Acquisition of Mental Models

HELGON (Nieper, 1987) is a system to assist a user in establishing a mental model incrementally. The system was specifically designed to support incremental query specification which allows users to develop a model of an information store. The multiple specification techniques supported by HELGON require a less elaborated MM by allowing the user to access the information store from different starting points.

Similar goals are pursued with our design environments (Fischer & Lemke, 1988). They allow multiple specification techniques in creating a program. They link internal objects and the external behavior and appearance, they provide animated examples and guided tours – again supporting the incremental development of a MM of a functionality-rich construction kit.

Exploiting Existing Knowledge and Reducing the Transformation Distance

Mental models should help us to understand the world and successfully use the artifacts given to us. Over the last few years, we have pursued two closely related research efforts relevant to these goals: *layered architectures* and *human problem-domain communication.* Both of these try to exploit existing knowledge and to reduce the transformation distance between a real-world object and its corresponding description within an informational system.

Layered architectures can increase the cognitive comprehensibility by providing a descriptive level which facilitates the understanding of a complex system. Instead of bridging many layers at once, it is often sufficient enough to understand just the layer below the one in which we want to operate.

The FINANZ system (Fischer & Rathke,1988) is an example which is built upon a layered architecture (see Figure 2). Each layer in the architecture provides a reference point for understanding.

Human problem-domain communication (Fischer & Lemke, 1988) allows us to redraw the borderline between the amount of knowledge coming from computer systems versus coming from the application domain. Systems built to support human problem-domain communication establish an "application-oriented" vocabulary which is essential for effective reasoning and communication in the domain and which bring the system model closer to the situation model (Fischer & Kintsch, 1986).

Issues and Questions about Mental Models

How Do People Form Mental Models?

Mental models can be acquired in a number of different ways. Experience, self-exploration, training, instruction, observation (apprenticeship model) and accidental encounter are some of the possibilities. I have asked colleagues and students in our department to document their *great ahas* which were defined as events of finding (not by actively searching) a piece of

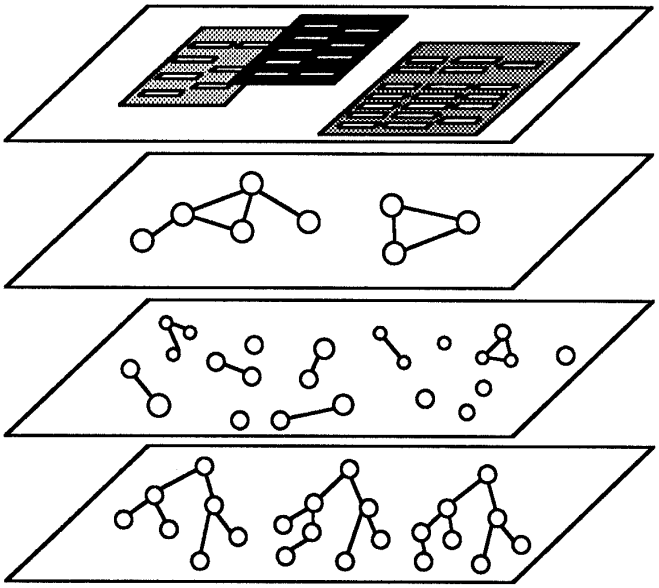
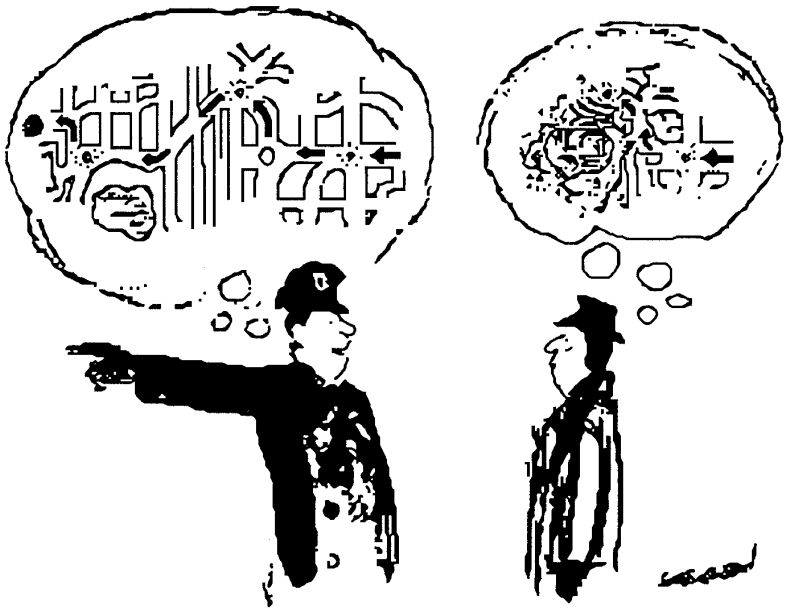


Figure 2. A Layered Architecture for FINANZ. Layers are problem specific at the top and general at the bottom. The bottom layer (first layer) represents LISP and OBJTALK. The second layer contains elements, primitive constraints and frames for the construction of knowledge-based systems. The third layer is a kit-building kit for financial planning systems. The top layer contains kits for specific application systems. The layered architecture of FINANZ allows the employment of spreadsheet technology for bridging the gap between the two topmost layers.

information that solves a long perceived problem. The answers indicated that the following factors play an important role:

- the existence of information alone is not enough;
- the role of context is crucial;
- the applicability conditions of pieces of information have to be known;
- accidental encounters often provide more important information than planned incidents;
- to appreciate the relevance of a command, prerequisite knowledge structures must exist;
- users live happily with safe but suboptimal strategies (i.e. they do not necessarily want to update their mental model to correspond more closely to the system model even if they can).

Another interesting question arises from Simon's analysis of complex systems (Simon, 1981). He argues, and provides evidence, that complex systems evolve much faster if they are able to develop stable subsystems. What are (or can be, or should be) the stable subsystems in a mental model for a certain problem domain?



Drawing by Stevenson: © 1976 The New Yorker Magazine, Inc.

Figure 3. Mental Models and Mutual Understanding. The picture characterizes a number of issues: the visitor cannot build up a mental model; more of the information provided should have been put in the world (e.g., by drawing a map). The structural model provided by the policeman is too detailed; it may be possible to avoid this by tailoring the explanations more to the goals and objectives of the visitor. The policeman may have also enhanced the understanding by using layers of abstractions (see the discussion of main versus side streets in (Ehrlich & Walker, 1987)).

Which Information Do and Should Mental Models Contain?

This question contains a descriptive and a prescriptive part. To assess someone's understanding of her/his knowledge is known to be difficult (see the problem of figuring out the knowledge of human experts to be used in expert systems). Several techniques that have been used to analyze this knowledge are experiments, think-aloud protocols, and interviews.

On the prescriptive side, we should make sure that people not only know the correct procedure ("action"-part in a production) but also know the applicability condition ("condition"-part in a production) and the things that can go potentially wrong. The development of good conceptual models (see my previous remarks about recursion) can make a big difference how easy it is for someone to develop an adequate mental model.

The development of intelligent support systems which simplify and reduce learning (Fischer & Lemke, 1988) may reduce the amount of information required to be available in the mental model, because the support systems will allow us to derive the information when needed (learning on demand).

How do we Assess the Quality of a Mental Model?

Mental models can be characterized along the following dimensions:

- structure (e.g., state space models, ATN);

- utility (e.g., input and output characteristics, related to goals, procedures);
- extensibility;
- generalizability;
- predictive power and explanatory power (they should let us figure out what happens in novel situations).

Proposed requirements for good mental models are:

- *accuracy* – but see the previous remark about cognitive fidelity versus physical fidelity.
- *consistency* – but at what level? If we have a system using defaults for the version numbers of files, the command `delete file` should default to the oldest file, whereas the command `print file` should default to the newest file. Systems should be consistent only with respect to the underlying semantics; although they may be inconsistent at the syntactic level. Jonathan Grudin pointed out in his presentation at the workshop that there is no unique best strategy to achieve consistency. Instead the dimension along which a system should behave consistently is dependent on the goals one wants to achieve (for example, in abbreviating commands, the following strategies can be used: (a) truncation: delete → de; good choice for: to reconstruct the abbreviation from the name; (b) vowel deletion: delete → dlt; good choice for: to reconstruct the name from the abbreviation; (c) single letter strategy: delete → d; good choice for: overlearned tasks). He also argued for that predictability is more important than consistency.
- *completeness* – but (as I argued before) for HFCS there are no complete models.
- *task-orientation* – but general purpose computer system support an arbitrary number of tasks.
- *short derivational paths* – but this requires higher level of abstractions (and a shared understanding and mutual intelligibility) which do not exist in many systems.

For which Tasks are Mental Models Irrelevant?

Tasks which require highly skilled behavior (e.g., some aspects of flying an airplane, playing a game of tennis, typing on a keyboard) should not be driven by consulting the mental model. These tasks have to be trained that the knowledge and skill to execute them will exist in compiled form and their execution becomes increasingly independent of the mental model. It should not be necessary to use first order problem solving principles, but the execution of the commands and actions should be driven by a large number of compiled activation patterns.

What would we (as Computer Scientists) like to know about Mental Models

We (as Computer Scientists in cooperation with Cognitive Psychologists) would like to develop design principles for comprehensible systems. We are convinced that mental models (and the other models mentioned on page 19) play an important role in this research effort. Independent of what is given to users, they will form some mental model of their artifacts. We are interested in mental models, because of their potential to facilitate this acquisition process and to provide users with powerful cognitive tools.

The following is a short (and incomplete) list of the questions which we hope mental model research will eventually provide answers to:

- Which kinds of mental models are possible for high functionality computer systems (where they are most needed)?
- How can mental models support incremental learning and learning on demand?
- Is our architecture of "increasingly complex microworlds" an adequate architecture for mental models (if we try to structure systems in this way, don't we have to do the same thing with mental models?)?
- Should systems have MM of themselves (self-images; self-explicating systems); how do we create them? what do they need to contain?
- Users are "moving targets" in using systems over long periods of time. What happens to their mental models in such a process? How do we scale up from laboratory experiments to assess mental models to longitudinal studies in naturalistic settings?
- How are systems able to assess the mental models of users (a prerequisite to support differential descriptions, custom-tailored explanations, etc.)?
- Which new kind of difficulties will users encounter if they have to develop mental models for systems which consist of meta-structures which must be interpreted by the users (e.g. Hypertext systems and tailorable systems).

References

Boecker, H.-D., Fischer, G., & Nieper, H. (1986). The enhancement of understanding through visual representations. In *Proceedings of CHI '86 Human Factors in Computer Systems*, (Boston), ACM, New York, 44–50.

Brown, J. S. (1986). From cognitive to social ergonomics and beyond. In D. A. Norman & S. W. Draper (Eds.), *User Centered System Design:; New Perspectives on Human-Computer Interaction* (pp. 457–486). Hillsdale, NJ: Erlbaum.

Carroll, J. M., Olson, J. R., & Anderson, N. (1987). *Mental models in human-computer interaction: Research issues about what the user of software knows*. Technical Report 12, Cognitive Science and Machine Intelligence Laboratory, University of Michigan, June 1987.

Chambers, A. B., & Nagel, D. C. (1985). Pilots of the future: Human or computer? *Communications of the ACM*, **28**, 1187–1199.

Ehrlich, K., & Walker, J. H. (1987). High functionality, information retrieval, and the document examiner. In G. Fischer & H. Nieper (Eds.), *Personalized Intelligent Information Systems* (Chapter 5). (Technical Report 87-9), Institute of Cognitive Science, University of Colorado, Boulder, May 1987.

Fischer, G. (1987a). A critic for LISP. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, J. McDermott (Ed.), (Milan, Italy), Morgan Kaufmann Publishers, Los Altos, CA, 177–184.

Fischer, G. (1987b). *Learning on demand: Ways to master systems incrementally.* (Technical Report). Department of Computer Science, University of Colorado, Boulder, CO, 1987.

Fischer, G., & Kintsch, W. (1986). *Theories, methods and tools for the design of user-centered systems.* (Technical Report). Department of Computer Science, University of Colorado, Boulder, CO, 1986.

Fischer, G., & Lemke, A. C. (1988). Construction Kits and design environments: Steps toward human problem-domain communication. *Human-Computer Interaction,* **3**, 179–222.

Fischer, G., & Nieper, H. (Eds.) *Personalized Intelligent Information Systems.* (Technical Report 87-9), Institute of Cognitive Science, University of Colorado, Boulder, May 1987.

Fischer, G., & Rathke, C. (Submitted for publication, 1988). Knowledge-based spreadsheet systems. Department of Computer Science, University of Colorado, boulder, CO.

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

Nieper, H. (1987). Information Retrieval by reformulation: From ARGON to HELGON. In G. Fischer & H. Nieper (Eds.), *Personalized Intelligent Information Systems* (Chapter 19). (Technical Report 87-9), Institute of Cognitive Science, University of Colorado, Boulder, May 1987.

Norman, D. A. (1982). Some observations on mental models. In D. Gentner & A. L. Stevens (Eds.), *Mental models* (pp. 7–14). Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D. A. (1986). Cognitive Engineering. In D. A. Norman & S. W. Draper(Eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction* (pp. 31–62). Hillsdale, NJ: Erlbaum.

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas.* New York: Basic Books.

Simon, H. A. (1981). *The sciences of the artificial.* Cambridge, MA: The MIT Press.

Acknowledgments

The author would like to thank Walter Kintsch, Andreas Lemke, Helga Nieper-Lemke, Christian Rathke, Curt Stevens, and Thea Turner for providing me with insightful comments to an earlier draft of this paper. The research was supported by grant No. MDA903-86-C0143 from the Army Research Institute.

THE ROLE OF HOW-IT-WORKS KNOWLEDGE IN THE ACQUISITION OF HOW-TO-DO-IT KNOWLEDGE

Peter G. Polson
Department of Psychology and Institute of Cognitive Science
University of Colorado

This note is a brief report of a study done in the early 80's at at the University of Colorado by David Kieras, myself, and two students, George Engelbeck and Nancy Lee Willer. Our goal was to understand the role of how-it-works knowledge in the acquisition of operating procedures, i.e., how-to-do-it knowledge. We were attempting to replicate an experiment that had recently been completed in David Kieras's laboratory (Kieras and Bovair, Experiment 1; 1984). They showed that how-it-works knowledge facilitated the acquisition of how-to-do-it knowledge for a simple control panel device. The experiment described here was designed to show that we could scale up the Kieras and Bovair (1984) results by showing that how-it-works knowledge could facilitate the acquisition of how-to-do-it knowledge for a more complex system like a word processor.

The Experiment

The basic design and procedures for this study were similar to Kieras and Bovair's (1984) first experiment. The experimental group received instructions containing how-it-works information about both the hardware and software for the word processing system. The experimental instructions included a description of each component and its functions and descriptions of the interrelationships between each component in terms of control and information flow. In addition, subjects were told how the document filing system was organized. These instructions were about 10 pages long, and the average study time was about 35 min.

The Common Procedures

Otherwise, both groups were treated identically. Subjects were given detailed instructions on the low level details of interacting with the system's hierarchical menu interface. They were then trained on a series of five utility tasks like check spelling, duplicate diskette, and change document name. Each task was learned to a strict criterion. During training, it was not possible to leave the correct path. Feedback was given after any error with transitions off the correct path blocked by the tutoring subsystem. More information about the training procedure and the system can be found in Polson, Muncher, and Engelbeck (1986).

After a 24 hour delay, subjects were given a series of increasingly rigorous retention and transfer tests. The first test involved relearning the five utility tasks using the initial training procedure. Then, there were three transfer tests. The transfer test procedure did not constrain subjects to the correct solution path; subjects were interacting with a high fidelity simulation of the menu interface for the actual system. Incorrect menu choices took subjects off the correct path onto menus that that they may not have seen during training. Subjects were given 15 minutes to complete each test task.

The three transfer tests were retention, near transfer, and far transfer. The retention test required subjects to perform the training tasks with the unconstrained, high-fidelity simulation. The near transfer tests were tasks that were closely related to the original training tasks. The

far transfer test was a series of office tasks which were versions of the near transfer tasks presented in the contexts of scenarios in an office. Table 1 presents examples of training, transfer, and office tasks.

Training Task 4 Load Work Diskette "Denver" into the right slot of the diskette unit. Then: Duplicate the work diskette named "Denver" onto work diskette name "Ernest". Transfer Task 4 Load Diskette "Key" into the right slot of the diskette unit. Then: Condense the work diskette named "Key" onto the work diskette named "Label" Office Task 4 You have been typing a manuscript (diskette "zeiko") when a message appears on the screen that you have run out of disk space. "Zeiko" has several deleted documents on it. Diskette "algo" has been initialized and can be used if needed. What will you do so that you can finish entering the manuscript?

Table 1. Examples of Training, Transfer, and Office Tasks.

Duplicate Diskette and Condense Diskette are very similar disk copy utilities. The system did not maintain a list of free space on a diskette. Thus, the space released by deleted documents was not available for the storage of new documents. Condense Diskette enabled a user to recover space freed by deleted documents by copying an old diskette and performing garbage collection. Duplicate Diskette copied the intact disk structures along with the inaccessible space freed by deleted documents onto a new diskette. This later copy utility was much faster. The experimental group was told how the system handled space freed by deleted documents, and the purpose of the condense function was briefly described. There were similar interrelations between pairs of training and transfer tasks for the remaining four training tasks. However, the experimental instructions did not contain information that was relevant to all of the corresponding office tasks.

Results

The times to complete the original training, relearning, and three transfer tasks were averaged over subjects and tasks for each measure and group and are presented in Table 2. There were 22 subjects in each group and each subject performed five tasks. The differences between experimental and control groups show in Table 2 are small and none even approached significance. Clearly how-it-works works knowledge had no effects on any of these measures, and thus, we did not succeed in scaling up the Kieras and Bovair (1984) results to the acquisition of how-to-do-it knowledge for a word processor.

Test	Experimental	Control
Original Learning	10.8	11.2
Retraining	3.6	3.3
Retention Test	2.1	2.4
Near Transfer Test	2.8	3.0
Office Tasks Test	5.7	6.5

Table 2. Mean Task Completion Times (in minutes).

Another measure computed for the last three tests was the percentage of tasks completed without making any errors. The time data above is a mixture of three categories of performance: 1) tasks completed without errors, 2) tasks subjects completed while making one or more errors, and 3) task not completed before the 15 minute criterion. Subjects were give a score of 15 minute if they failed to complete a transfer task.

Test	Experimental	Control
Retention Test	73%	75%
Near Transfer Test	62%	63%
Office Tasks Test	41%	23%

Table 3. Percentage Of Tasks Completed Without Making Any Errors.

The only significant difference was found on the percentage of tasks completed without making any errors measure for the office task test. Furthermore, there was a highly significant groups by tasks interaction. For example on Office Task 4 shown in Table 1, 73% of the tasks for the experimental group were completed without any errors and 36% for the control group.

Discussion

Halasz and Moran (1983) found a very similar pattern of results for the task of evaluating complex expressions with a simple stack of calculator. Their experimental group received detailed how-it-works information on the internal structure of the stack calculator and how to use the stack to evaluate embedded arithmetic expressions. Their control group was taught a procedure dealing for with embedded expressions but given no information about the internal structure of the calculator. Halasz and Moran found no differences for training time and for near transfer test performance.

However, when subjects were required to use the calculator's registers to store intermediate results in the process of solving multi-step word problems, there was a large facilitatory effect observed for the experimental group. Halasz and Moran showed that information about the registers was relevant to solving the problem of storing the partial results, and subjects who successfully used the registers to store intermediate results showed improved performance on word problems equivalent to far transfer problems.

The office tasks, which were in essence word problems, required subjects to solve a problem from a description that only hinted at which procedure was to be used. Referring to the three tasks shown in Table 1, the experimental group's how-it-works instructions contained information necessary to select the correct procedure from the description of the problem in the scenario. In particular, for the condensed diskette function, the how-it-works instructions

described the circumstances under which you would want to use the condense function. Our results show that how-it-works knowledge does not facilitate the learning of normal operating procedures or the execution of near transfer tasks when these transfer tasks are described using terms that appear in menus. The only effects we found were for tasks with large problem solving components, like the office tasks.

Conclusions

We draw two morals from this study. One, how-it-works knowledge is useful to the extent that is directly relevant to the kinds of problem solving operations involved in learning to use a system without explicit instruction, in error recovery, and in generating novel operating procedures. Two, obviously users must be able to acquire and retain such material. This later requirement presents presents a dilemma since there's a large amount of such knowledge, and it contains difficult to comprehend material because of the high intensity of new concepts for the non-technically trained reader. Today's systems require that users acquire a good deal of how-it-works knowledge in order to recover from errors or generate novel procedures. Learning this knowledge is a major source of difficulty for non-technically users.

References

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

Halasz, F. G., & Moran, T. P. (1983). Mental models and problem solving in using a calculator. In *Proceedings of the CHI 1983 Conference in Human Factors in Computing*. New York: ACM, pp. 212–216.

Polson, P.G., Muncher, E., & Engelbeck, G. (1986) A test of a common elements theory of transfer. In *Proceedings of the CHI 1986 Conference in Human Factors in Computing*. New York: ACM, pp. 78–83.

Acknowledgement

This research was supported by IBM on a contract to David Kieras and Peter Polson. The conclusions expressed in this paper are the authors and not necessarily those of IBM.

MENTAL MODELS FOR ENGINEERED SYSTEMS AND COMPUTERS

David Kieras
Technical Communication Program
University of Michigan

The purpose of this paper is to discuss some puzzles concerning mental models for systems such as computers and other complex systems (that here will be termed "engineered" systems). This work is based on experiments and simulation modeling done in my laboratory over the last several years. However, despite the intense interest in mental models, the issues are not yet clear, and so this paper will be focused more on posing questions than presenting answers.

Three Puzzles

There are three puzzles that will be discussed in this paper. The first is a *theoretical* puzzle. Just what is a mental model for a computer system or other complex system? The term *mental model* is used in an incredibly vague way. The second is an *empirical* puzzle. Experimenting on mental models for computer systems or other systems is in fact quite difficult, and no empirical consensus has yet emerged on the value of mental models in interacting with systems such as computers. Third, there is a *practical* puzzle of how should mental models be chosen, both in the design of systems and for training people how to use systems.

Theoretical Puzzle

Just what is a mental model? We need a good characterization. Some distinctions and definitions have to be made, followed by empirical work to support that characterization. Below are some candidate ideas that may help clarify the theoretical situation.

Basic Content of a Mental Model

In the domain of how people interact with computers or other devices, the most central aspect of a mental model is that it has to contain knowledge specifically about the system or device in question. It is of no benefit to define *mental models* as synonymous with knowledge in general. If the situation in question involves a human interacting with a piece of equipment, then the only useful and intelligible sense of mental model is one in which we are concerned with the human's understanding or knowledge of the equipment itself.

Human knowledge of equipment is by no means simple. Kieras (1982) had subjects give descriptions of pieces of equipment both when the equipment was physically present and could be manipulated, and also from memory. There are quite a few different kinds of information that people manifested, one of which is information about the internal structure and mechanisms of the device. However, it should be pointed out that this kind of knowledge is by no means the most salient knowledge that subjects displayed in this task. Much of their descriptions seemed to focus on how to operate the device, its behavior, and its external features. Of particular importance for this discussion are three kinds of knowledge: (1) the *how-it-works* knowledge; (2) knowledge of *procedures* for operating the device, both to accomplish goals and to repair or maintain it, and (3) a knowledge of the *behavior* of the device, that is, the input/output function. These kinds of knowledge were clearly distinguished in the subjects' descriptions; for example, they would often describe the behavior of an alarm

clock quite separately from information about procedures for operating it, and such descriptions were also quite distinct from descriptions of how the clock worked internally.

In a discussion of what mental models for devices could consist of, these three categories of information are potential candidates. However, I would argue that the procedures for operating the device do not constitute a useful characterization of a mental model for a device, because there are adequate rigorous theoretical descriptions of procedural knowledge in the context of people interacting with equipment (e.g. Kieras & Polson, 1985; Bovair, Kieras & Polson, 1988), and these descriptions contain nothing of the other two types, suggesting that a mental model is something else.

How the device works inside is perhaps the best single category of knowledge to use as the definition of a mental model. This is indeed a characterization of what people know about the device that is above and beyond simply the procedures for operating it. This is also relevant to both empirical and practical issues because most experiments that attempt to manipulate mental models for equipment have involved teaching people how the device works, and the importance of training on such information for equipment is of considerable practical concern to organizations such as the Defense Department.

The knowledge of how the device behaves is intriguing, because we know from our own experience that many devices behave in very regular and consistent ways, and thus can be interacted with in a fairly intelligent and powerful manner without much training, and without deep how-it-works knowledge. For example, classes of devices such as VCRs or digital alarm clocks share patterns of behavior, and thus the user can rely on this knowledge of stereotypical behavior to infer procedures for operating a novel member of this class. This concept of a mental model that is based on knowledge of regularities in behavior of devices, is an intriguing one and potentially quite important. However, other than perhaps the work of Clayton Lewis (this volume), it seems that even less work has been done on this concept of a mental model than on the how-it-works model concept. For this reason, the rest of this paper will focus on how-it-works knowledge as being the basis of a mental model.

In this context, the basic content of a mental model for a system can be defined as:

- Knowledge of the system structure
- Knowledge of system principles
- Knowledge of strategies for using the structure and principles to perform tasks

Knowledge of the system structure consists of knowledge of the components of the system and their connections with each other. Knowledge of the system principles is knowledge of how the individual components behave. By combining the knowledge of principles with the knowledge of structure, a person can predict things such as what effect an input to the system will have on the internal state of the system, or how different internal states are manifested externally.

The mental model content of structure and principles is normally hierarchical; any given device can be represented as a hierarchy of levels of analysis, with system principles applicable to the components described at that level of analysis. For example, electronic equipment is often described at a high level in a block diagram, in which the blocks represent major subsystems. Usually these blocks exchange only information, such as signals or control data, rather than voltages or currents. These block diagrams can then be represented at a lower level of analysis involving actual electronic devices, until eventually the system is represented in terms of a *conventional lowest-level set of primitive components*, such as individual resistors, transistors, and so forth. In electronic equipment, the level of discrete components and their behavior

principles is normally the lowest level of description required. These principles describe how the behaviors of the components can be combined to get more complex behaviors. It is important to notice that in any technical domain involving real equipment, this lowest useful level of analysis is a convention in the technical literature of the domain, and explanations of a system at this level are normally taken to be adequately accurate and complete explanations. Thus, for example, practical electronics textbooks do not go into the quantum mechanics of the solid state physics involved in the behavior of a transistor to explain how a transistor radio-frequency amplifier operates. Thus any description of a how-it-works mental model will necessarily have made a commitment to some level of analysis and level of detail involved in the model. Because real devices have many different possible levels of analysis, it can be important to be clear on what level is under discussion.

The third category of mental model content, the knowledge of strategies, at first glance may not seem part of the mental model for a device because it is not specifically about the device, in the same way as knowledge of the system structure and system principles. However, in constructing cognitive simulations of mental model reasoning of simple devices (Kieras 1984, in press-a), it became clear that without the strategic knowledge, the knowledge of structure and principles simply could not be applied to a task. For example, in the "phaser bank" study done by Kieras and Bovair (1984), the knowledge of system structure consists of information about the fictitious internal components of the device and how they are connected up to each other. The system principles consist of ideas such as how the components are energized in an all-or-none fashion, and an energized component can send energy downstream to another component that it is connected to. The principles also include the fact that energy can be transmitted through a switch only if the switch is on, and that indicator lights show whether energy is present.

The strategic knowledge involved is more complex. In these experiments, people were set the task of getting the phaser bank to "fire", as indicated by one of the indicator lights flashing, by manipulating the controls of the device, and using the indicator lights to detect and compensate for malfunctioning components. There are many strategies that subjects could use, some of which might not depend in any important way on the knowledge of system structures and principles. However, the strategy in the simulation model made explicit use of the how-it-works knowledge. The strategy was to first switch the device on, then reason about where energy is present in the device, and then work backwards from the goal component (where energy is desired to be) through the connections, until a source of energy has been found. Along the way, the required control settings are noted. The system structure is obviously being used here, and also the system principles such as that switches have to be on to allow energy to pass through, and that indicator lights show where energy is present.

The point is that without such strategies, the knowledge of system structure and principles simply does not have any consequences for behavior. On the other hand, such strategic knowledge is not necessarily general; at best it may be generic to classes of devices. For example the strategy just described may work quite well for any device consisting of a set of components with indicators and controls that has all-or-none energy states. The wiring in a house, or much of the wiring in an automobile, may meet this criterion, making such models perhaps generically useful. However, it is clear that the strategic knowledge is still specific to at least a class of devices. Perhaps the best real world example of the importance of this strategic knowledge is that of electronic trouble shooting. Skilled trouble-shooters apparently know more than just electronics concepts; they also have a repertory of strategies which can be used to isolate malfunctions as well.

Thus for a person to have a mental of a system requires knowing what components the system is made out of, how these components are connected, and how the components behave and interact with each other. In addition, for the mental model to be usable in a task, the person must have strategies for applying the structure and principle knowledge to meet task goals.

Designed Systems Versus Naturally-Occurring Systems

Systems such as telephones, airplanes, power plants, and computers, are *designed systems* in the sense that they were deliberately assembled in a technically informed manner from well understood components into a form that has specified functions and capabilities. This informed goal-directed construction sets engineered systems apart from naturally occurring systems, in which the capabilities and functions of the system and how it works may be unknown. That is, in naturally occurring systems, how the system "really" works is not understood, and so a technically correct understanding of the system is usually missing. However, in designed systems, the actual structure and behavior of the system was deliberately chosen, and so there is at least one technically correct description of how the system "really" works, and this description exists at the levels of analysis that are conventional in the technical domain responsible for the device. In practical terms, this means that designed systems are ones in which it is possible to distinguish mental models that are *technically correct* from those that are incorrect but possibly useful. For example two mental models that are equally accurate at predicting the behavior of a designed system might be distinguishable in how accurately they describe the actual structure of the system. However, in naturally occurring systems, technically correct descriptions are normally not available, meaning that two mental models that are equally accurate in describing the behavior of a system must be accepted as essentially equivalent.

Engineered Systems Versus Computer Systems

Computer systems are of course designed systems, just as automobiles, airplanes or power plants are designed systems. However, there seems to be some important differences between what could be termed *engineered systems* versus computer systems. By *engineered systems* is meant mechanical and electrical systems such as those found in automobiles, radios, or air conditioners. Clearly, what gives the computer system its unique power is its ability to represent essentially arbitrary combinations of symbols and processes, without limitations imposed by any concrete constraints. On the other hand, systems made up of gear wheels or electronic components can do only certain things. Such systems seem to differ from computer systems in that they perform concrete operations rather than abstract manipulations of symbols. They also seem to be made up of concrete components as opposed to abstract programs and processes. Finally, engineered systems within a domain tend to contain relatively few types of components, rather than essentially arbitrary structures, and likewise the possible behaviors of those components are limited rather than arbitrary. It would seem that mental models for engineered systems might be fundamentally easier to acquire than those for computer systems, simply because concrete content may be easier to learn; for example, there is more potential for perceptual-motor involvement. In addition, if there really are fewer components and principles for engineered systems, then mental models for them would be easier to learn than for computer systems. Perhaps such engineering knowledge may be more generic than knowledge of computer systems.

Task-Relevant Versus Task-Irrelevant Models

Previous work (e.g. Kieras & Bovair, 1984) suggests that a mental model is useful only if it is relevant to performing the task. To do so, it must support inferences that are useful to the task, such as inferring a procedure or the cause of a malfunction, and which are hard to learn either because of the learning situation or the nature of the procedures themselves such as arbitrary content or length (cf. Kieras, in press-b). For example, the earlier studies such as Kieras and Bovair (1984) compared operating the device using a mental model versus having to learn how to operate it by trial and error. In the trial and error situation, the procedures were fairly difficult to identify and to remember, but quite simple to infer using the mental model information. But results recently obtained in my laboratory show that it is possible for people to learn the same device most rapidly of all when a logically complete procedure for operating the device is presented in a reasonably clear and concise fashion. Thus, if simple procedures

can be presented quite directly, the mental model turns out to be the *hard* way to learn how to operate the device.

Thus, in the context of a task, mental models can be both relevant and irrelevant. In trouble-shooting automobile electrical systems, mental models are apparently quite important, but in the routine use of the ordinary telephone they are quite unimportant. In the context of computer systems, mental models are both relevant and irrelevant. Recovering from computer system crashes or malfunctions seems to be a case where understanding how things work is important. But learning arbitrary command names probably is not facilitated by mental model knowledge. In fact, it could be argued that if a mental model is required in order to successfully operate a computer-based system, it is probably a symptom of either poor design of the system or inadequate technology. For example, the ordinary telephone is well adapted to the task of communication and because of its intrinsic reliability and the power of the switching mechanisms, it is simple for even a child to operate just by rote. Thus the design and technology make knowing a mental model irrelevant to the task of operating the telephone. Computer systems have apparently not reached a similar level of technological maturity.

Computer Versus Application

Mental models for computer systems could take on one of two forms. One would be a model of the computer system itself; the other would be a mental model of a particular application that runs on a computer system. The mental model for the computer system would consist of knowledge of the internal operation of the hardware and operating system software. This information is quite similar between different computer systems, because the concepts of digital hardware and computer programs are essentially the same. There will be some stereotypy at high levels; for example, command language interpreters are highly similar to each other. In contrast, the mental model for an application on a computer system is likely to be quite specific to the type of application, and there may not be much similarity between different applications. For example, a spreadsheet has little similarity to a word processor, and so rather different mental models would be required. It seems clear that there are more kinds of applications than there are kinds of computers and operating systems. In a dedicated system, such as a dedicated word processor, a mental model of only the application may be adequate for supporting interactions with the equipment. An intriguing possibility is that engineered systems do not require this distinction between the mental model for the hardware supporting the function and the application, perhaps because engineered systems are all essentially dedicated systems.

Metaphorical Versus Veridical Models of Computer Systems

In modern computer systems we have seen the appearance of styles of user interface that implement metaphors based on familiar notions. Of course the Macintosh, with its "desk-top" metaphor, is a key example. Apparently, the user's mental model should be based on the metaphor for the interface to be effective. Such *metaphorical* mental models can be contrasted with *veridical* models, which are based on a technical understanding of how the system "actually" works. Of course, the philosophically-inclined would object, making the hackneyed point that nobody *really* knows how *anything* works. However, in the context of the practical issues, this objection is merely pedantic obfuscation. As discussed above, designed systems are associated with technically accurate descriptions of their structure and principles. There is a very serious problem: the people that construct equipment know this technical knowledge of how the equipment actually works, and left on their own, they burden documentation and training with this knowledge in all of its glorious detail. This knowledge cannot be simply dismissed as merely one "view" of the system among many, because in the technical domain this knowledge has the special status of being a technically complete and accurate description of how the system works. Such descriptions are routinely prepared and circulated within the technical domains and are often supplied to the user communities as well. The practical question is whether this *existing technical description* is useful for the mental models of

ordinary users, or whether this technical veridicality should be discarded in favor of the familiarity of metaphors.

Metaphorical models. With metaphorical mental models, the user is invited to think about the computer in familiar terms. Familiar examples of such models are: the computer is an idiot that blindly follows instructions; the computer display is like a desktop with paper documents and folders on it. The key feature of a metaphorical model is that this familiarity allows the user to make useful inferences without specific training. A few key inferences apparently justify the entire metaphor. For example, the idiot metaphor is used to emphasize the fact that a computer only does what it is told. In the case of the desktop metaphor, a key inference is that one can change the logical location of a piece of information by moving the corresponding icon through space, just as it is done on a desktop. In a well-designed metaphor the ability to use such easy inferences apparently result is very easy-to-use, "intuitive" interfaces. However, it is possible that apparently good metaphors are in fact confounded with other properties of the system. For example, the Macintosh also has a pervasive consistency in its methods, in the sense of the Kieras and Polson (1985) work, and a very thorough concealment of internal mechanisms. Both of these would reduce substantially the amount of learning required to interact successfully with the system. This may be responsible for the success of the Macintosh, rather than the desktop metaphor. Appropriate studies to answer this question have not been done.

The problem with metaphorical models is that there is a limited amount of detail possible in the inferences enabled by the model. These inferences are not correct past a fairly superficial level of the model, meaning that one can not expand the hierarchy of levels of analysis while staying within the metaphor. This means that most inferences below a superficial level are false. Some example trivial false inferences: Since an instruction-following idiot's feelings can be hurt, one must be careful to be polite to one's computer. With use, manila folders get worn out and have to be replaced, so a Macintosh user will periodically have to get fresh folders and move everything into them. But there are some more subtle false inferences: An instruction-following idiot can understand English sentences, meaning that computer commands can be expressed in natural language. The pages in a paper document can get out of order, meaning that the user must explicitly fasten document pages together. It is impractical to have folders inside of folders, meaning that you need a different kind of storage to hold folders than you do to hold documents. The question is whether the inferences that are supported by the metaphorical model are adequately correct for the user's needs, or whether the user has to learn additional models in order to effectively use the system.

Veridical models. A typical veridical model for a computer system would consist of a hierarchical description, to some level of analysis, of the system hardware and software. It does not have to be complete, because it could be expanded as needed to deeper levels of analysis, and there is a well-defined mapping from one level to another. The key feature of a veridical model is that the expansion remains technically correct down the level of conventional primitives, and that inferences can be generated by general rules for that level of analysis. Such models would be complex and hard to learn and use if the system itself is arbitrary and complicated. For example, a veridical model for a typical computer text editor may be very difficult to learn and use if the text editor is of the style that involves a large number of arbitrary single-keystroke commands, rather than a more structured organization.

The key advantage of veridical models is the inferential power resulting from their being based on technically correct descriptions. A simple example can be given for how files are stored on a disk. At the top level, the mental model is that the file is stored as magnetic patterns recorded on the disk. This can be expanded into information about how the file is organized and accessed. The disk is divided into sectors, and a file is stored in several sectors on the disk. The location of the sectors for a file is recorded on the disk in association with the file name in a directory which is in a standard location on the disk. Thus reading a file involves searching the directory for the file name, getting the location of the file sectors, and then reading the

contents of those sectors. Note that this description is at a relatively gross level of analysis, but it supports some very elaborate and subtle inferences. For example, it supports the inference which seems puzzling to novice computer users, that damage to a small part of the disk, the directory, can result in a file not being readable on the disk even though all of the data is still present. An additional, even more subtle inference is that it may be possible to reconstruct the file by examination of the contents of the disk sectors.

Another characterization, suggested by Clayton Lewis, of the difference between veridical and metaphorical models is based on four aspects of a mental model. The first is the *familiarity* of the model components. Metaphorical models normally do well in this regard. Second is the *size* of the model, which is how much information it contains. Typical veridical models for a computer system will be quite large, metaphorical models fairly small. The third aspect is the *scope* of the model, which is how much of the system behavior or procedures it will explain, or support inferences about. As noted above, metaphorical models normally will not generate inferences below a very superficial level, whereas veridical models, because they are technically accurate, potentially can explain everything about the system above the conventional primitive level of description. The fourth aspect is the *derivational length* of the model, which is the length of the inferential path from the model to the phenomenon to be explained, or procedure to be inferred. Within their scope, the derivational length of metaphorical models is very short compared to veridical ones. For example, dragging an icon from one folder to the other on the Macintosh causes the corresponding document to be moved to the other folder. Within the desk-top metaphor, the explanation is trivial; of course, moving something through space changes where it is. The explanation for the same phenomenon in terms of how the Macintosh actually works will be *much* more detailed and complex, even at a gross level of analysis. For example, it would be critical to describe how a folder is represented on the Macintosh, and how membership of a file in a folder is represented, and how the dragging operation brings about a change in this representation. Even an experienced Macintosh user and computer expert may completely lack this information. The design of the Macintosh hides these details extraordinarily well, limiting our ability to acquire this knowledge incidentally, but at the same time, strengthening the desk-top metaphor.

Thus, there are substantial advantages and disadvantages of both types of models. A good metaphorical model will contain mostly familiar concepts, and although its scope is limited, it covers the most important and frequent cases, and allows these useful inferences to be made very quickly and easily. A good veridical model will typically be quite large, with lots of arbitrary content; its scope is very great, but involves typically long derivational paths. Most seriously, the cases covered by a veridical model beyond those handled by a good metaphorical model will typically be only the low-frequency, unusual cases.

The Empirical Puzzle

Do mental models really help humans interact with a computer system? In the simple systems studied by Kieras and Bovair (1984), there is clear evidence that understanding the concepts or principles underlying the system will produce improvements in performance. If by a mental model we mean some kind of understanding of the system above and beyond the procedures for simply operating it, there have been many failures to demonstrate any beneficial effects of learning such information (See Polson, this volume, for further discussion). Methodological problems in such research are especially difficult. How does one get subjects to learn a complex set of concepts and apply them reliably? How does one get the subjects to use this mental model, instead of settling for rote learning, which may in fact be the easiest way to do the task? Since mental models are supposed to help especially with problem solving situations, how do you deal with the vagaries and high variability of problem solving data? How does one ensure that the experimental task is one in which mental models would be expected to play a positive role? This empirical situation cannot really be resolved without considerably more progress on clarifying the theoretical concepts of mental models and what they do. Certainly as

long as the concept of a mental model is vague, we will not be able to define specific and precise experimental manipulations for demonstrating mental model effects, or exploring how mental models work.

Practical Puzzle

The basic practical puzzle is how to choose a good mental model for the user to have. There are many specific questions here, involving both training and design. In terms of training, how do you explain an existing system to the users? Training and documentation could convey mental models better; a close look at most training documentation reveals that it in fact contains a hodgepodge of incomplete and poorly organized material.

In terms of design, how do you design the system so that it is easily understood? The major drawback of veridical models is their size and complexity. Can we devise systems with veridical models that are easy to learn? That is, instead of taking the risks of limited coverage with metaphorical models, it might be better to design systems in such a way that a simple veridical model that can be easily learned is adequate to perform with the system.

References

Bovair, S., Kieras, D. E., & Polson, P. G. (1988). The acquisition and performance of text editing skill: A production system analysis. (Technical Report No. 28), University of Michigan.

Kieras, D. E. (1982). What people know about electronic devices: A descriptive study. (Technical Report No. 12, UARZ/DP/TR-82/ONR-12). University of Arizona. (DTIC AD A122189)

Kieras, D. E. (1984). A simulation model for procedure inference from a mental model for a simple device. (Technical Report No. 15, UARZ/DP/TR-84/ONR-15). University of Arizona. (DTIC AD A143909)

Kieras, D. E. (in press-a). The role of cognitive simulation models in the development of advanced training and testing systems. To appear in Frederiksen, N., Glaser, R., Lesgold, A., and Shafto, M.(Eds.), *Diagnostic monitoring of skill and knowledge acquisition.* Hillsdale, N.J.: Erlbaum.

Kieras, D. E. (in press-b). What mental model should be taught: Choosing instructional content for complex engineered systems. To appear in Psotka, J., Massey, D., and Mutter, S. (Eds.), *Intelligent Tutoring Systems: Lessons Learned.* Hillsdale, N.J.: Erlbaum.

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

Kieras, D. E., & Polson, P. G. (1985). The formal analysis of user complexity. *International Journal of Man-Machine Studies*, **22**, 365–394.

Acknowledgements

Work on this paper was supported by the Office of Naval Research under Contract No. N00014-85-K-0138, NR 667-543.

A SIMPLE PROCEDURE FOR EVALUATING USER-INTERFACE DESIGN

Stephen T. Knox, Wayne A. Bailey, Eugene F. Lynch, and Gerald M. Murch
Cognitive Interface Project
User Interface Research
Tektronix Laboratories

A procedure for user-interface evaluation has been employed at Tektronix to assess learnability issues related to the 11000 series laboratory oscilloscope (Knox and Bailey, 1987). The procedure, called the "question-asking protocol," was used previously at the IBM Tokyo Cognitive Engineering Laboratory (Kato, 1986). The Japanese scientists used the method to study user behaviors when interacting with a prototype word-processing system. Researchers asked subjects naive to the editor command set to perform typical word-processing tasks. Each subject was provided a "tutor" who would dispense user-interface information on an "as needed" basis. That is, when the subject needed to know a specific command to carry out some editing operation, a query was made to the tutor. Consequently, the subjects' queries were the verbal data used by the researchers to address the following user-interface issues: 1) What instructional information do users of this system require? 2) What features of this system are harder to understand than others? and 3) How do users develop an understanding (or misunderstanding) of the overall user-interface?

This procedure was adapted at Tektronix to study the 11000 series laboratory oscilloscope. Six subjects naive to the user-interface of this series performed four typical measurements. Application of this procedure resulted in additional verbal data apart from the subject queries. Subjects performing the tasks tended to talk aloud to the tutor about what they were doing. This type of subject behavior has been previously reported (Connally and Tullis, 1986) and the verbalizations constituted added information (Bailey and Kay, 1986).

The verbal data allowed insight into the subjects' cognitive processes of building a model of the 11000 series user-interface in the context of solving applied problems. In conjunction with the verbal data, the video recordings of the experimental sessions resulted in non-verbal data that documented user-interface concepts which were easily understood. The video recordings of the experimental sessions and the subjects' verbal protocols were then used to address learnability issues of the user-interface. The purpose of this report is to detail the methods involved in applying this procedure. We will also include information on approaches to analyzing the data.

Equipment

A video camera, recorder and monitor are required to tape and later view the experimental sessions. Subjects should perform the experimental tasks in an area which gives them a sense of privacy.

Experimental Tasks

The subjects should be given a task or set of tasks that are typically performed with the system under study. Care should be taken to not make the tasks so difficult that they cannot be solved except by expert users of the system. A subject approaching the system for the first time may

have a model of the system's functionality, but will have little or no model of how to access the functions with the new user-interface. Consequently, the difficulty level of the initial tasks should allow the subject to focus primary attention on learning the system interface. Task difficulty may be increased as the subject becomes more confident.

For example, when evaluating learnability of a new editor, the initial experimental task should not be changing the font and margins of an indented paragraph. Ideally, the subject needs tasks that first acquaint him (her) with the edit and enter modes, cursor movement, and deletion of text. In evaluating the user-interface of the 11000 series, we asked subjects to acquire and measure the period of a signal as the initial task. Consequently, subjects learned the basics of signal acquisition, horizontal and vertical scaling and the menu-structure of the automated measurement system before moving on to the complex tasks involving more than one signal.

Instructions To The Subjects

In any experimental situation, subjects will feel as if "they" are being tested, as opposed to the system. Care must be taken to alleviate any subject apprehension during the instruction period. If subjects are handled in a careful manner from the start, complete and accurate data will be recorded. If subject apprehensions are not attended to, few questions and verbalizations will occur.

The subjects must be made aware that the purpose of the procedure is to evaluate the user-interface. It should be stressed that important information occurs when the subject encounters problems in understanding how to access system functionality. Therefore, subjects should never hesitate to ask questions of the tutor. The following instructions should be given:

1) You were selected for this study because you have had no experience with this system. You are not expected to "figure out" how to use this on your own.

2) Your primary goal is to verbalize any questions you might have about using this system in the context of completing the experimental tasks. Try to refrain from exploration that is unrelated to the tasks.

3) Please make your questions specific. Try to avoid general questions such as "what do I do now?" Specifically ask the tutor whether a function exists or how to access that function.

4) Some parts of this system are more complicated to use than others. You may not always remember information given you about these complicated parts. Please be prepared to ask the same questions over again.

5) The perfectly transparent user-interface has not yet been designed for any system. If you are having difficulty in understanding something, the user-interface is at fault, not you. Please be patient, and continue asking questions.

Tutor Training

The tutor should be relatively expert with the interface of the system under study and with its application to the experimental tasks. Improper actions by the tutor can seriously bias the data given by the subject. Therefore, the tutor must carefully monitor his (her) interactions with the subject. The following must be attended to:

1) Do not induce the subject to ask any particular question.

2) If you do not understand what the subject is asking, have them clarify. Under no circumstances should you infer what the subject is asking. To help facilitate applying this caution, take your time and think before you answer "do I understand what the subject is asking of me?"

3) Answer the specific question only and avoid giving more information than is requested. For example, if the subject were to ask about an editor, "how do I move the cursor up one line" the response should NOT be "cursor movement is accomplished by using the four directional arrow keys." The correct response would be "use the directional key with the arrow pointing up." Thus, if the subject does not ask how to move the cursor in other directions, an inference can be made that the interface design for cursor movement in the up direction has been generalized to other directions. The "key" to avoid giving unnecessary information is to TAKE YOUR TIME. Ask yourself "Do I understand the specific, local problem."

Some systems may have advanced functions of which the subject may not be aware. Ideally, the user-interface aids the subject to access these functions and use them. However, subjects may not find the advanced functions. After the subject has completed the task without using advanced functions, the tutor may wish to tell the subject that such a functions exist, and have the subject repeat the task using these functions.

Also, the tutor should always be sensitive to behaviors indicating that the subject is having problems with the experimental tasks. The subject may not understand the task or may employ a strategy that makes little sense to the tutor. Here, the tutor should intervene and ask the subject to clarify the operations they are performing. In this way, the tutor will be able to discern the nature of the problematic behavior. If an unfruitful strategy is being used, the tutor should briefly enumerate all alternative methods of solving the task. Or it might be the case that the subject did not fully understand the task itself, in which case the tutor should clarify the misunderstanding.

As a general rule, the job of the tutor is to answer the subject's questions and not teach the subject about the system. The tutor's responsibilities are only to the local question of operation and to keeping the subject on task.

Tutors are made, not born. Thus, the tutor should be trained during pilot experiments. That is, conduct a few protocols, and use the tapes to evaluate the performance of the tutor relative to attending to the above guidelines. These pilot sessions will enable the experimenters to critique the tutor's interactions, so that the best possible data is collected during the "real" experimental sessions.

Analysis

In well conducted protocols, the video tapes offer a wealth of information about the learnability of the user-interface. This information is given by both the verbal protocols and the non-verbal (visual) data. Viewing the video tapes and listening to the verbal data can provide insight into the development of a cognitive model of the user-interface. In particular, all the data which address the following issues should be studied:

1) What are the model building behaviors? For example, after learning the access path to a function, is this path generalized to other similar functions? We found that once subjects learned the arrows icons assigned scaling operations of the waveform to the control knobs, this hypothesis about icon behavior was generalized to other icons. Sometimes the subject will verbalize these hypothesis testing instances (e.g. "if I do this will that

happen?), and at other times the subject will test a hypothesis implied from previous information without verbalizing.

2) What are the major interface problems? What surprised the subject? Did the subject perform an action and obtain a result that differed from their expectation? Why did the subject have the expectation? Did the subject have a misunderstanding of system functionality or access to functions? Can that misunderstanding be traced to the model of the interface that the subject was developing?

3) What concepts of the interface are most difficult to learn? Were there behaviors or questions by the subjects that the system designers found surprising? Did subjects repeatedly ask questions about some particular feature? Why is this feature difficult to learn? Does the feature or access to the feature behave contrary to the subject's expectations.

4) Are there any interface "traps." That is, do subjects ever get the system into a state from which there is "no return" save for toggling the system on-off button?

5) In the context of writing a manual, what instructional information do new users need? Which system features promote the formation of a cognitive model of the user-interface? Can this model be consistently generalized to the entire system?

In general, the verbal data document how users develop an understanding of the system user-interface. They record some problematic aspects of the interface as well as identify some features which promote learnability of the system.

References

Bailey, W. A., & Kay, E. (1986). Towards the standard analysis of verbal data. *IEEE Systems, Man, and Cybernetics Proceedings*, 297–302.

Kato, T. (1986). What "question-asking protocols" can say about the user interface. *International Journal of Man-Machine Studies*, **25**, 659–673.

Knox, S. T. & Bailey, W. A. (1987). *Learnability of the 11000 series laboratory oscilloscope*. (Technical Report UIR-804-002). Tektronix Laboratories, OR: User-Interface Research.

Connally, C. E., & Tullis, T. S. (1986). Evaluating the user-interface:Videotaping without a camera. *Proceedings of The Human Factors Society*, 30th Annual Meeting, 1029–1033.

ORGANIZATIONAL INFLUENCES ON INTERFACE DESIGN

Jonathan Grudin
MCC

Consider the expression "User-centered System Design." At least two minds are implied – the user and the designer – and mental models have a role for each. The organizers' description of this workshop focused on conflicting user mental models – models of task domain and models of the system interface. Other work, including mine, has focused on the designer. At the last CHI conference Tom Landauer illustrated the possibility for conflicts between the designer's model of the system, and the designer's model of the user. My work has attempted to show that the designer employs additional models that influence the design process to the detriment of the user interface. These are models originating in the design task domain. The designer has a complex, largely unconscious model of the process of interface design and development. This model is an occupational necessity, but it can work systematically against user-centered design.

"User-centered System Design" points to another symmetry. One might imagine a typical user of a product built by a typical designer. We're usually aware that it's a mistake to think of a "typical user" – users vary all sorts of ways. However, less attention has been paid to the interface designer. Who is this user-centered system designer? Is it someone like ourselves? In fact, interface designers vary significantly. Many of them only spend a fraction of their time on interface design. More importantly, most interfaces don't have one designer. In a company of any size, a single interface will be designed by several people, from marketing, technical writing, software engineering, training development, human factors, industrial design, and so on. Many of them probably won't even meet each other. As a result, organization, coordination, and communication among people is a key part of interface design.

Let me explain what started me worrying about this topic. A mystery arose over the years I participated almost daily as an engineer in interface design meetings. It began with my amazement at the number of hours that we could spend arguing about an interface design decision without reaching a conclusion. Meetings could end, follow-ups be held, but some issues were finally dropped only due to the exhaustion of the engineers involved, and as likely as not would be revived months later for another round of discussion.

At the same time, I gradually realized that compromise is the way of life for engineers, finding acceptable tradeoffs among alternatives. Why couldn't these folks, who make compromises every day, ever agree on some particular interface topics? That was the mystery. The take home message is, if professional compromisers can't agree on an issue, there's more going on than meets the eye.

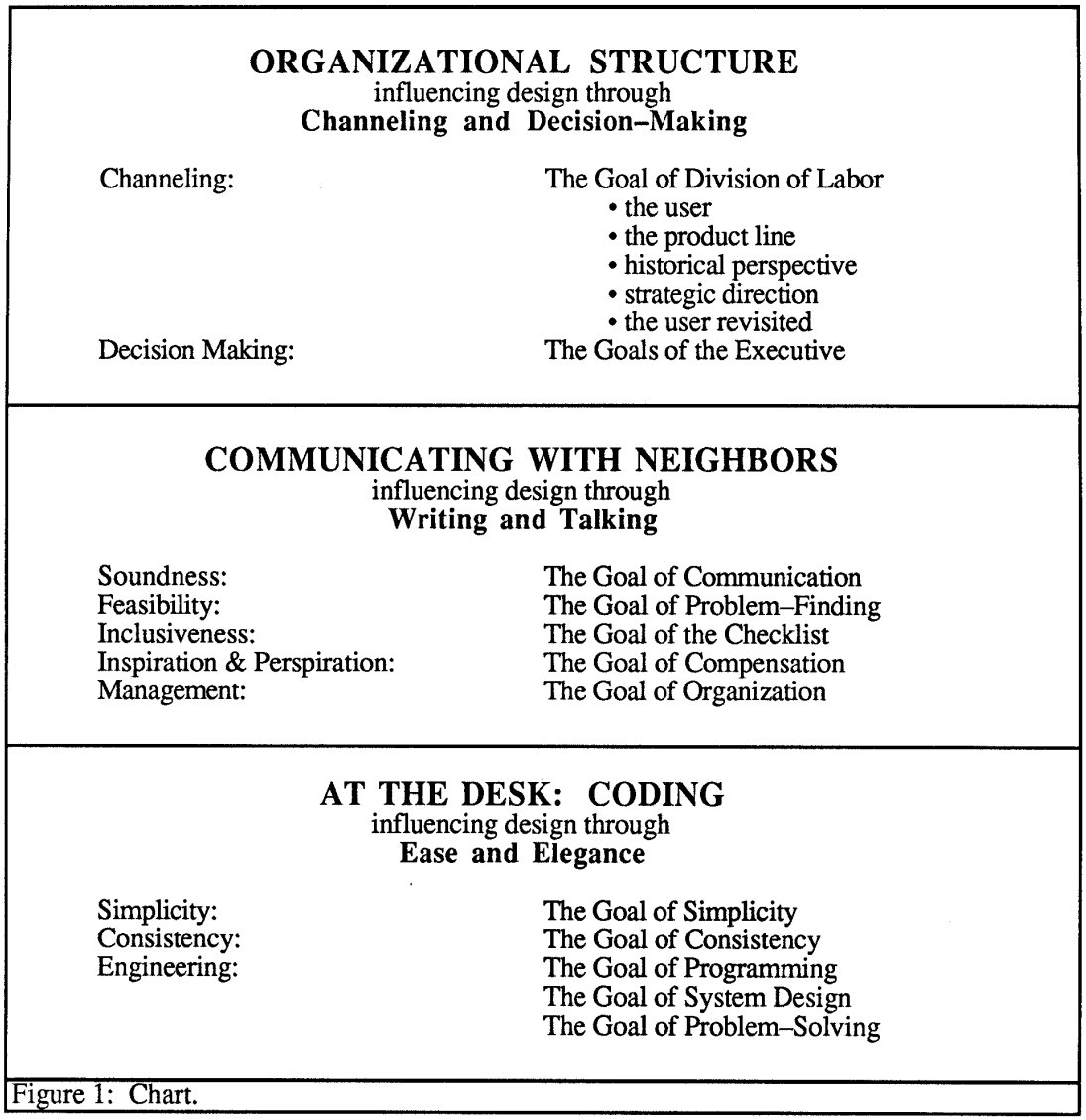


Figure 1: Chart.

Figure 1 is what emerged when I took a hard look at these intransigent problems.

First let me explain what it is about. In an organization of any size, user interface designers have many goals in addition to designing interfaces. They may be coding, or marketing. They're certainly communicating, trying to anticipate design or implementation problems, delegating, managing people, working out a rational division of labor, and so on. Designers become skilled at these activities. They develop models for how to reach these non-design goals efficiently and effectively. Each of these activities has a logic to it. The problem is, particular interface choices may make it easier or more difficult to reach these other, orthogonal goals. In the absence of adequate evidence about which design alternative will be better for the user, and sometimes in the face of such evidence, the desire to reach these other goals can consciously or unconsciously lead to conflict and bad interface decisions.

I'll give a quick overview of the figure, then get to some examples. Down the middle are about 12 goals that are tangential to interface design. All are worthwhile, but each can adversely

effect design. Each box corresponds roughly to a level in the organization. At the corporate level, Division of labor is worked out, and executive decision-making takes place. In the middle box are activities that work at the team or group level. Ideas must be communicated, problems anticipated, thoroughness insured, rewards and punishments administered, and the organization chart adjusted. At the bottom are goals that programmers focus on: simplicity, consistency, coding, system design, and problem-solving in general. I'll skip the plight of the Human Factors Engineer on the right. I'll just cover Communication and Division of Labor here.

Communication. Interface designs have to be communicated in an organization, among the people responsible for different parts of the design, their managers, the implementers, marketing representatives, and so on. This communication is usually done on paper. Communicating the interface design is a good goal, as are all the rest of these. The problem is, the interface that's easiest to communicate on paper is not necessarily the easiest to use once it's built. You might think that a design that's more easily described will be easier to use, and in the absence of evidence to the contrary, why not go with a design that also promotes your goal of communication?

The conflict arises because paper is a static, spatial laid out medium, whereas human-computer interaction is a temporal, focused-in dialogue. Try the following exercise. Figure 2 shows a set of pull-down menus. The menu category is at the top of each column, with the selections appearing below it. Your job is to search for the menu selection called "Print." Ready?

EDIT	**FILE**	**INTEGRATE**	**DISPLAY**	**CREATE**
Copy	Close	Execute	List/Icon	Folder
Copy Link	Print	Edit	Search	Document
Move	Send	Read	Save Results	Graph
Link	Object Profile	Link Profile	Reorder	Database
Place	Map	Make Version	Reselect	Spreadsheet
	Word Index			
	Update			
	Delete			
	Revision History			
	Compact			

Figure 2. Pull-down menus.

The question is: How much time did you spend studying the menu categories, the labels on top? Probably not much at all – you scanned the entire set for a few seconds and found the word. This was from a design specification. The problem is, a user will never see all the pull-downs unfurled simultaneously the way you did. The user has to find "Print" by looking at the column headings and guessing which one "Print" falls under. If "Print" appears in an inappropriate category, the reader of the spec may not notice – the word is still quickly found. But the user could get lost many times before learning where to find it. Also, the logic underlying the overall categorization of menu items is easier for the spec reader to grasp, looking at the full set. The user has to piece the picture together from pieces seen briefly over a period of time, and may never grasp the high-level categorization, and may encounter problems that don't occur to the spec reader.

There's a corresponding problem in the temporal domain. Multi-step temporal sequences occur very naturally in computer use, but can be difficult or messy to represent on paper. It might make sense in specific places to have 3 or 4 levels of pull-downs, or a second-level pull-down that appears with a couple sentences of explanation, but these would be messy to represent in a

spec diagram such as this. As a result, pulldown menu sets can end up flatter and less clear than they would if the design exploited the full range of opportunities that dialogue affords.

The conclusion isn't to avoid written design specs, which obviously meet a valid communication need. The problem is that the designer works with knowledge or a model of what constitutes a clearly communicated design, which may conflict with the model or reality of an easy-to-use design.

I'll finish with the effects of Division of Labor. This goal is different from the others in that it doesn't distort the interface design directly. Division of Labor distorts the interface design by giving the designer a distorted user model. I'm choosing to talk about it because in my experience, it had devastating effects on the implementation of Help systems, which are of interest to a number of people at this workshop.

Division of Labor is obviously necessary. It leads to communication patterns that are designed to channel specific information to some people without bothering others with it – Division of Labor both routes and conceals information. As things stand, information about users may end up being concealed from user interface designers in several ways.

First, in a product development environment, **direct access to the real user population** may be blocked, particularly at the user site. For example, a sales rep may not want developers to meet important customers. A developer may offend or mystify the customers and make them anxious. Or, the developer may talk about products in development and leave the customer reluctant to buy more of what's now available. And it can get worse – what if the customer is the CIA? The fact is, access to external users is often so tough that interface designers give up trying, at which point it's hard to center the interface on the user.

Next, **the product line**. Middle managers may be concerned with an entire product line. But in general, most developers are focused on their particular application or even piece of one, and don't know about the entire product line. When interface developers are only aware of their own application, it's natural to optimize the interface to it without realizing that some features may be inconsistent with interfaces to other applications that the customer will use simultaneously. Optimizing the interface for your own application at the expense of a broader consistency is a major temptation anyway, because you tend to imagine the user with only your application. A lack of inter-product consistency is almost inevitable if the designers don't know which set of applications will be used together in particular end-user environments.

Installed base issues. Similarly, only a certain level of management is typically knowledgeable about the current customer base. Until recently, user experience and expectations weren't a major consideration. But as the user base grows, as people develop strong preferences through years of use, as retraining costs rise, minor interface enhancements won't succeed any more than the Dvorak keyboard has succeeded in replacing the QWERTY. But this historical perspective and user base familiarity is usually only known to experienced, senior managers, and not to designers whose tendency is more likely to be toward innovation anyway.

Similarly, a handful of executives may carefully guard the company's strategic direction. This includes information about which applications, systems, and configurations will be aimed at which user populations. Again, the designer of an application interface who lacks this information can't produce an interface that will maximize consistency across the set of applications that the end-user will confront. The designer will optimize for the one application, and will likely introduce unnecessary inconsistencies in the end-user environment.

I mentioned the problem of user access at the design stage. The last case represents another time when **user access** is denied to the designer and developer: **after the product is shipped**. This may be one of the most insidious barriers to good interface design.

As I said, engineers are constantly engaged in compromise, working with tradeoffs. "This implementation will save 10K bytes but be a little less modular," or "This implementation will run a little faster but take a month longer to complete," or "This chip will permit two extra slots but add $500 to the sales price." The thing about these decisions is that they're made once, the price is paid – an extra month of development time, a more powerful chip – and then they're history. The problem is, it's all too easy for these professional arbitrators, compromisers, to treat the user interface as just one more bargaining chip. "This interface would be a little nicer, but take an extra two weeks to code." The interface can get traded down the river like these other things partly because to the developer, once the decision is made, it's history. This wouldn't be true if the developer got feedback from users for the life of the product. THE INTERFACE REALLY IS SPECIAL. The two weeks development time, or 10K of memory, can be committed and it's over, but that user interface may be out there bedeviling thousands of users daily for years. However, this isn't apparent to many developers – once it's built and shipped, it's out of their lives, they're on to the next job.

Division of Labor is devastating to Help system implementation because user difficulties and complaints aren't heard by the development organization. Software bugs may make it back, but user interface complaints go to the customer support organization. Developers aren't notoriously sympathetic to less experienced users in the best of circumstances, and will be less so if they never hear back from users. A good help system or tutorial system could well save the company a ton of money in customer service calls, in customer training, and so forth. But they usually won't get built because for a large company, the savings would be in the customer service budget, but the expense of developing the Help system will come out of the development budget.

In conclusion, Division of Labor conspires to prevent designers from getting a good understanding of users and user environments, and thus prevents the formation of accurate user models and task models, while the other goals in Figure 1 bring specialized knowledge or models to bear that can conflict with good interface design.

MENTAL MODELS AND ANALOGICAL REASONING

Robert Thomas King and Erich J. Neuhold
Institute for Integrated Publication
and Information Systems
Gesellschaft fuer Mathematik und Datenverarbeitung
Darmstadt, Federal Republic of Germany

Analogical reasoning has traditionally been treated in logic and even in cognitive psychology as a formal, syntactic process involving the structural comparison of arguments or situations. It is argued in this paper that analogical reasoning should be based on the comparison of mental models, even of different domains, with consideration of pragmatic and contextual information. Applications of this approach include the construction of user interfaces to complex information systems.

Introduction

Analogical reasoning can be seen as a type of induction (Holland et al., 1986), in which knowledge of familiar situations is transferred to new situations which bear some similarity to the familiar situations. The notion of "similarity", of course, is slippery and somewhat difficult to define. If I have in the past bought Brand X shoes and they have worn well, I can infer that when I need to buy shoes again, I can assume that new Brand X shoes will also wear well, so I will decide to buy Brand X again. This is a type of intradomain dependency, in which experience of shoe buying and wearing influences my future decisions in shoe buying. Interdomain dependency arises when knowledge of one domain is transferred to another. For example, experience of the behavior of fluids is used to explain phenomena of electricity: voltage is described as "pressure", and amperage as "rate of flow". Although fluid dynamics and electricity are separate domains, they exhibit structural similarities which permit analogies. In studies of logic (e.g., Copi 1978), analogy is used to demonstrate the validity or invalidity of certain arguments which are clearly valid or invalid and which bear structural similarity to the original arguments. Thus the following argument:

> Congressman Jones favors price controls.
> Communists favor price controls.
> Therefore, Congressman X is a Communist.

is refuted by comparing it to the structurally identical argument:

> Congressman Jones breathes air.
> Communists breathe air.
> Therefore, Congressman Jones is a Communist.

which is patently invalid. However, as legions of logic instructors can testify, beginning students of logic often fail to accept the analogy, since breathing air is clearly irrelevant to one's political persuasion, but one's attitude toward price controls is not. Thus considerations of context and background knowledge play an essential role in everyday analogical reasoning.

An example is provided by the well-known experiments of Wason (see Wason and Johnson-Laird, 1972; Anderson, 1980), in which subjects were presented with four cards which had letters on one side and digits on the other. The cards were laid out so that the subjects could

see the letters E and K and the numbers 4 and 7. There were asked to name the minimal set of cards which must be turned over to prove the hypothesis that "if a card has a vowel on one side, then it has an even number on the other side". The correct set is the cards showing the E and the 7, but only 4 percent of the subjects answered in this fashion. Almost everyone agreed that the card showing the E had to turned over (corresponding to the inference rule of modus ponens), but very few saw that the card showing the 7 had also to turned over (modus tollens), although a large proportion insisted that the card showing the 4 had to be turned over, in spite of the fact that that was irrelevant to the hypothesis. From this experiment it could be concluded that subjects normally interpret simple implication as bidirectional implication. However, when the experiment was repeated in a structurally identical form but with another content (Johnson-Laird et al., 1972), the results were strikingly different. Subjects were shown four envelopes, two face down and two face up, where one of the face-up envelopes bore a 40-lire stamp and the other a 50-lire stamp, and one of the face-down envelopes was sealed and the other was not. The subjects were then asked to name the minimal set of envelopes to be turned over to test the hypothesis that "if a letter is sealed, then it has a 50-lire stamp on it". Here 21 out of 24 subjects made the correct choice, turning over the sealed letter and the letter with the 40-lire stamp. It is apparent, then, that content and background knowledge play a role in the application of reasoning.

Mental Models

Mental models are internalized representations of external situations (see Johnson-Laird 1983; Holland et al. 1986). They enable us to understand complex situations or processes by constructing mental representations which correspond structurally to the external situations or processes. Thus, to repeat an earlier example, an electrical system may be mentally modeled as a hydraulic system, or the rotation and revolution of planets around a sun may be modeled as clockwork. Models may indeed employ models themselves: Eddington modeled the structure of an atom (electrons revolving around a nucleus) on the model of a planetary system (Copi 1978). One way of instantiating models involves production systems (Newell and Simon 1972; Anderson 1983; Holland et al. 1986), which consist of condition-action rules and a working memory. Here one must distinguish between diachronic and synchronic rules: diachronic rules represent changes in state, and are typically related to causal connections between events ("If I expose silver salts to light, they will darken"); synchronic rules refer to static relations such as logical or physical entailment ("If X is a mammal, then X has a four-chambered heart"). In the sense that rules can be understood to represent properties, sets of rules with similar triggering conditions can be seen to define categories, including default assumptions ("If X is a dog, X barks").

Analogical Reasoning

We will discuss analogical reasoning here in application to problem solving. We describe a situation as being defined by a state function (a complex rule set S). A change of state involves the application of a transition function T to the initial rule set, resulting in a new rule set G describing the resultant state. The term 'state' here is being used rather loosely; it may also refer to continuous processes. Note that different transition functions may effect the same change in state: to cause water to boil, I may either heat it or put it in a (partial) vacuum. Analogical reasoning in this context may be described as the application of known procedures for causing state transitions to new situations which bear similarity to the known situations. Since alcohol is a liquid like water, I may reasonably infer that alcohol can be caused to boil by either heating it or putting it in a vacuum. The known situation and transition functions can be called the source of the analogy, and the new situation and transition the target (cf. Holland et al. 1986).

Analogical reasoning has four steps (Holland et al. 1986):

1) constructing mental models of source and target;

2) selecting the source as a relevant analog to the target;

3) mapping the target onto the source; i.e., determining which elements of the target correspond to elements of the source;

4) deriving a new transition function T' for the target based on the transition function T of the source.

These steps may occur concurrently and be interdependent; also incomplete knowledge may to applied to select a potential source which is reinforced by ease of mapping between the components of the source and the target. Continued application of analogical reasoning results in schema induction (cf. Schank 1982). Success in boiling water and alcohol leads me to infer that I can boil all fluids by applying heat to them or putting them in a vacuum. The process of schema induction is essential in establishing interdomain analogies. For example, if I know that ingestion of a small amount of alcohol results in my being in a relaxed state, but that the ingestion of a large amount results in a stupor, I may infer that one aspirin tablet might help alleviate a headache but that several could cause an adverse reaction; this is an intradomain analogy. If then I observe that a little physical exercise is pleasant and improves my physical condition, but that too much exercise may cause sore and even injured muscles, I may well combine this knowledge with my knowledge about alcohol and aspirin and induce the schema that a little of something may be good and pleasurable, but too much of it may be deleterious. (Rule: live moderately.) Particularly striking cases of schema induction are commonly termed "insight". Newton's discovery of universal gravitation resulted from the insight that if the sun attracted the earth, then the earth must also attract the sun (Cohen 1981). Newton was purportedly puzzled by the fact that an apple fell to the earth but that the moon didn't; what he realized was that the moon indeed was continually falling toward the earth, but missing it each revolution.

Example: Electronic Communication as Mail

Networked computer systems normally offer their users some form of communication facilities with which messages can be exchanged between users. These facilities may be modeled on conversation (e.g., the Symbolics "Converse" facility or the Berkeley UNIX "talk" utility) or on the postal system (various forms of "electronic mail"). We will consider here the latter form, which involves messages of more permanence than the "conversation" utilities. The analogy with the postal system is strong: messages ("mail") is delivered and picked up, after which it may be read and filed away or discarded. The mail system consists then of four phases: messages are delivered to a mailbox, also called an "inbox", where they reside until they are picked up; they may then be read and afterwards disposed of, either by discarding them, filing them, or even (in some cases) returning them to the mailbox for future reference. In the case of the UNIX mail system, mail to a particular user is written to a file (normally "/usrpub/mail/username"). The existence of this file causes the message "You have mail" to be issued after a login. With the "mail" command, the user can fetch his/her mail to a buffer, where it can be read; the messages can then be written to another file (called a "folder") or left in the mailbox file. They can also be deleted (discarded). The Symbolics Zmail facility is similar: mail is delivered by writing it to a file (called "mailhost:>username>mail.text"); after invoking the Zmail facility, the user can issue (or click on) the command "Get Inbox" which fetches the messages to a buffer where they may be read. The inbox itself is renamed and deleted, and messages are automatically written to a file called "mail.babyl" when the facility is exited. There is thus no option for returning mail to the inbox. The postal analogy is perfect in both cases but for the minor point that the inbox file is deleted after it is emptied, whereas a physical mailbox is of course not destroyed after each use. It therefore should in principle present no problem to combine UNIX mail and Symbolics Zmail in installations where

computers running UNIX and Symbolics Lisp Machines are networked together. To send mail from Zmail to UNIX mail requires only that the corresponding UNIX file be written to by the Zmail facility, and vice versa. In practice, however, this approach is impracticable. The Symbolics Genera network software contains information on the structure of the UNIX file system and its command codes, so that Zmail can write to the appropriate UNIX mail file, but the UNIX operating system has no knowledge of the Symbolics file system. It is therefore possible to send mail from Zmail to UNIX, but not vice versa, and the problem is intractable as long as one is stuck in the analogy that the sending system has to deposit the mail in the other system's mailbox. The system is analogous to the actual postal system in West Germany. Employees of companies, universities, or research institutes have both private mailboxes (at their home addresses) and official mailboxes at work. An employee on vacation can have his firm forward his official mail to his home or vacation address, but West German postal regulations prohibit the forwarding of personal mail to business addresses. A person at home can thus receive in his home mailbox both personal and business mail, but in his office mailbox only business mail; the person at home is analogous to the UNIX mail recipient, who gets both UNIX mail and Zmail; the person at the office is analogous to the Zmail recipient, who can only get Zmail. What is the solution? One possibility is that the office person send someone (e.g., the office mail clerk) to his home mailbox to fetch his personal mail and bring it to him at the office. This way of viewing the situation solves the UNIX mail/Zmail dilemma. Since the Symbolics Genera software "knows" the UNIX file system, the Zmail facility can be instructed to look for incoming mail not only in its own inbox file, but in the UNIX inbox file as well. The UNIX mail system thus does not need to write to the Symbolics mail file; normal UNIX mail can be fetched in this fashion by the Zmail facility, which, however, does write to the UNIX inbox file. The solution to integrating Zmail and UNIX mail results therefore from the insight that a mail system not only delivers mail, but can be used to fetch it as well, and a consistent application of the postal analogy can lead to this insight.

Example: Personalized Views of Complex Databases

The user of a complex information system is often confronted with the existence of large fact and document databases which have been independently created and are independently managed, and which may be based on entirely different structural principles, e.g., hierarchical, relational, or object-oriented. Usage of these databases has typically required that the user be cognizant of the underlying conceptual schema, in order to place a query against the database. The user may, however, have a specific goal in mind, such as building a new industrial plant or planning a vacation, and he or she is faced with the problem of integrating information presented in several different forms. To accommodate end users or application programmers who do not wish to have to learn the conceptual schema of a database, many systems offer predefined "views" of the data based on assumptions concerning the interests or knowledge of the user. However, the user may wish to place a query which has not been anticipated by the designer of the user interface, or a new user may not know which view to choose in advance in order to get the desired information. What is needed is a system by which views can derived dynamically through interaction with the information system (Neuhold and Schrefl 1988). In this conception the user places queries concerning objects of interest, which are in the user's attention toward the database through several queries or transactions, thus establishing a context in which the objects occur. A knowledge-based system which records the contexts of interaction and the objects which are queried can then, through processes of analogical abstraction and schema induction, infer the view that the user is taking. This approach requires, furthermore, that the information recorded in a database be represented abstractly; an approach in this direction is offered by the VODAK data model described in Klas, Neuhold and Schrefl (1988). For example, consider the case of an information system containing data on a village near the seacoast of an industrial company. A query concerning hotel facilities in the village is compatible with several views. If a subsequent query involves the proximity of the hotels to the beach (or if a suitable beach is present) it will cause the system to induce the "touristic" view and subsequent queries will be responded to with this view in mind. A query

concerning deep harbor facilities and perhaps good rail and road connections will induce the "industrial" view. Note that these views overlap somewhat (hotel facilities, restaurants) and may even involve contradictory elements (a harbor could be seen in one view as a yacht harbor, in the other as a freight harbor). The same facts may be seen in the views in an entirely different light. Proximity of the village to industrial centers could be seen in the industrial view as promoting easy expansion, but in the touristic view as offering convenient facilities for weekend relaxation for nearby workers.

Conclusion

The study of how human beings reason by analogy can be fruitful in constructing user interfaces to complex information systems. A knowledge-based system with sophisticated facilities for recording and structuring contexts of interaction can be constructed to recognize repeated patterns of activity and to induce schemas under which subsequent interactions can be judged. The process involves nonrigid pattern recognition, so that similarities in interaction patterns can be discerned without requiring exact matches. Rather, the system must be capable of constructing an abstract model of the interaction. Massively parallel approaches to computation (e.g., Rumelhart, McClelland et al. 1986) or pattern-based approaches to the study of cognition (Margolis 1987) offer promising prospects in this regard.

References

Anderson, J. R. (1980). *Cognitive psychology and its implications.* San Francisco: W.H. Freeman and Company.

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

Cohen, I. B. (1981). Newton's discovery of gravity. *Scientific American*, March, 1981.

Copi, I. M. (1978). *Introduction to logic.* Fifth Edition. New York: Macmillan Publishing Co., Inc.

Fischer, G. (1986). Cognitive Science: Information processing in humans and computers. In Winter, E. (Ed.), *Artificial Intelligence and man-machine systems.* New York: Springer Verlag Lecture Notes.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). *Induction: Processes of inference, learning,and discovery.* Cambridge, MA: The MIT Press.

Johnson-Laird, P. N. (1983). *Mental Models.* Cambridge, MA: The Harvard University Press.

Johnson-Laird, P. N., Legrenzi, P., & Legrenzi, M. S. (1972). Reasoning and a sense of reality. *British Journal of Psychology*, **63**, 305–400.

Klas, W., Neuhold, E. J., & Schrefl, M. (1988). On an object-oriented data model for a knowledge base. In *Proceedings of the European Teleinformatics Conference.* North Holland.

Margolis, H. (1987). *Patterns, thinking, and cognition: A theory of judgment.* Chicago: University of Chicago Press.

Neuhold, E. J., & Schrefl, M. (1988). Dynamic derivation of personalized views. Institute for Integrated Publication and Information Systems, Gesellschaft fuer Mathmatik und Datenverarbeitung, Darmstadt, Federal Republic of Germany.

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

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. (1986). *Parallel distributed processing*. Cambridge, MA: The MIT Press.

Schank, R. C. (1982). *Dynamic memory*. Cambridge: Cambridge University Press.

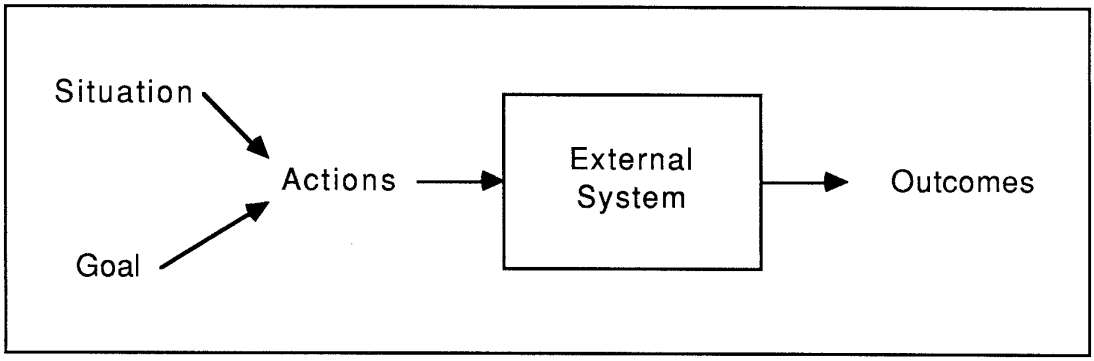
Wason, P. C., & Johnson-Laird, P. N. (1972). *The psychology of reasoning: Structure and content*. Cambridge, MA: Harvard University Press.

SOME THOUGHTS ABOUT MENTAL MODELS IN HUMAN INTERACTIONS WITH COMPUTING SYSTEMS: REFLECTIONS ON THE BRECKENRIDGE WORKSHOP

Gary M. Olson
Department of Pschology
University of Michigan

The concept of a mental model has emerged from recent research in cognitive science (Gentner & Stevens, 1983). A mental model is a user's† internal representation of the causal mechanisms that underlie the behavior of an external system. "Classic" work on mental models has examined the physical world (e.g., space, motion, substance) or "engineered" physical devices such as electrical circuits, calculators, watches, and computers. What exactly is a mental model? What other kinds of representations could user's have? Of what utility are mental models? What are the kinds of external systems for which people might have mental models? If the discussions at the Breckenridge workshop are any indication, there is lots of confusion about these issues. Surprisingly, a majority of the time at the workshop was spent on definitional issues. In these notes I present a few reflections on mental models triggered by these discussions.

Mental models are about a person's beliefs about what is going on inside an external system. The figure presents a schematic view of the kind of situation for which a mental model is relevant. In some situation, with some particular goal in mind, I (or someone I am watching) acts on an external system, things happen in the external system, and outcomes result. A mental model is my causal "theory" about what is happening inside the external system. My "theory" may or may not be closely connected to what is actually going on. It may in fact be wrong. My "theory" may have global coherence or it may not. But to be a mental model my internal representation must be a causal representation of what is going on in the external system.



There are other kinds of knowledge I could have about the behavior of an external system that would not be a mental model. The most elementary representation is a set of actions associated

† Person or program -- I will focus on persons' mental models in this paper, but the idea has been used in recent work in AI as well.

with specific outcomes, what Carroll and Olson (in press) call "simple sequences." If I do action A, outcome B will result. Many users of computer systems learn to do things on the basis of such pure "how to do it" knowledge. Indeed, for many systems, this is almost all one is ever taught. Back when I used the University's time-sharing system I learned by rote (well, not learned, since it was on a note taped to my terminal) how to direct printing to the laser printer at the computing center. I followed a specific set of actions that led to a specific result, but nothing about the actions made sense to me, even though in reality they were part of a complex set of instructions with many options and parameters.

If I had learned the set of rules and parameters associated with these commands, I could have constructed a richer representation for my "how to do it" knowledge. Here my knowledge of how to make things happen would have an organization based on goals, methods, selection rules, in short, all the things that make up the GOMS model (Card, Moran, & Newell, 1983). Such knowledge is richer in that it might allow me to construct alternative methods for achieving the same goal. But this is still "how to do it" knowledge, not "how it works" knowledge.

Only when my mental representation includes "how it works" knowledge has it attained the status of a mental model. To repeat my earlier point, such knowledge may or may not be veridical, and it may be complete or partial. I may know how something works at a high level of abstraction but not at the lowest level required if I were going to build it. As long as my "how it works" knowledge allows me to reason from what I do through various supposed internal stages inside the device to the eventual behavior of the device it is mental model knowledge I am using. Much of the discussion at the Breckenridge workshop focused on exactly where the boundary is between the various degrees of organized "how to do it" knowledge and true "how it works" knowledge. There well may be no sharp boundary. But clear cases ought to be distinguishable.

Mental models support several very important cognitive activities: the *interpretation* of a system's behavior, and *counterfactual reasoning* about what might result from certain actions on a system ("what if" reasoning). Interpretation is about understanding why an external system behaves the way it does. This can be important in learning about a system, and in handling novel behaviors. Both interpretation and counterfactual reasoning are important when impasses or unexpected behavior are encountered. Mental models support the kind of problem solving needed to cope with these.

Mental models have their primary effects on behaviors that occur over seconds, minutes, and hours, such as learning, reasoning, and problem-solving. The execution of skilled, routine behavior is dominated by other forms of mental representations (Anderson, 1982). Mental models support the deliberative, non-automatic behaviors most likely seen in novices, or in experts facing novelty or error recovery.

Most mental models arise via induction from interactions with systems. Occasionally, we are taught an explicit mental model for a system (e.g., Kieras & Bovair, 1984). Because most mental models arise through induction, they are often inaccurate, incomplete, and changing. But they help us understand the systems we interact with, help us interpret the behavior of the systems we use, and help us recover from the inevitable errors and impasses we encounter.

What is the utility of mental models for the user? There are two classes of utilities. The first is external. Mental models are potentially important in teaching people about external systems. This is the role investigated in the important Kieras and Bovair (1984) studies. Mental models are also potentially important in designing systems. To the extent that mental models facilitate reasoning about how to interact with a system, its characteristics and interfaces could be designed to support induction of an appropriate mental model. Similarly, the characteristics of displays could be such as to facilitate mental model reasoning, supporting whatever mental model users had either been taught or had induced. For instance, direct manipulation interfaces

are a kind of mental-model supporting interface style (Shneiderman, 1983; Hutchins, Hollan, & Norman, 1986). Knowing about the kinds of mental models people use in interacting with a system could also aid in the design of help systems, or aid in the design of intelligent agents that would facilitate a person's interactions with a system. A final class of external reasons for the possible importance of mental models is communication. Mental models provide a language for talking about complex systems that allows one to infer characteristics of a system from a knowledge base. Mental models provide a basis for establishing mutual intelligibility about a system, a shared understanding that can be the foundation for discourse.

There are also important internal utilities for mental models. A mental model provides an effective mnemonic system for remembering what to do with a system. Ultimately, skilled performance will depend on automatic activation of highly specific procedural rules (e.g., Anderson, 1983). But during learning having a causal model of a system will make it much easier to remember what to do. Further, even after extensive learning, in a system with any complexity only a small part of one's interactions with a system are routine, and many novel behavioral sequences will need to be constructed. I have already mentioned the importance of mental models in handling novel cases and recovering from error or impasses. Finally, mental models could play an important affective role in dealing with complex systems, in that one will feel much better having systematic knowledge of a system than in having a set of rotely learned, isolated procedural rules.

Clayton Lewis listed some important criteria for the usefulness of mental models at the Breckenridge workshop. Briefly, these were:

Familiarity — Mental models are useful to the extent they draw upon familiar domains. For instance, one could think about the unfamiliar domain of electrical circuits in terms of the more familiar one of water flowing through pipes (Gentner and Gentner, 1983). There is a close tie between mental models and analogies, with mental models often being a kind of dynamic analogy.

Extent — Mental models help us make the potentially complicated behavior of an external device simple. To this end, a mental model is useful to the degree that it itself is fairly limited in extent. A really complicated mental model in it self is difficult to learn and use.

Derivational scope — A mental model is more useful when it covers more of the potential behavior of an external system rather than less. Can I derive predictions and explanations for all possible inputs to the system? Can I understand all possible outputs?

Derivational length — This refers to the length or complexity of the account I need to construct in order to understand any particular behavior of the system. Generally, I will prefer a mental model that has a modest derivational length associated with any particular explanation. Otherwise, I will get lost. Thus, if the system itself is quite complex, keeping derivational length modest will lead me to form mental models that are approximate, abstract, incomplete. I choose to form my causal accounts of system behavior at a level of chunking of components that leads to comprehensible derivations.

Mental models should be viewed for their **utility** for the user, for their **efficiency** in permitting the user to understand and adapt to external devices, and for their **robustness** in

APPENDIX. LIST OF PARTICIPANTS.

Guests:

Stuart Card
XEROX PARC
3330 Coyote Hill Rd.
Palo Alto, CA 94304

Jonathan Grudin
MCC
P. O. Box 200195
Austin, Texas 78720

Julia Hough
Lawrence Erlbaum Assoc.
110 W. Harvey Street
Philadelphia, PA 19144

Dave Kieras
Technical Communication Program
TIDAL Building
2360 Banisteel Blvd.
University of Michigan
Ann Arbor, MI 48109-2108

R. Thomas King
Institute for Integrated Publication and
Information Systems
Gesellschaft fuer Mathematik und
Datenverarbeitung
Postfach 104326
Braunshardter Weg 11
D-6100 Darmstadt
West Germany

Catherine Marshall
US West Advanced Technologies
Advanced User Interfaces
6200 S. Quebec St.
Suite 170
Englewood, CO 80111

Jim Miller
MCC
Human Interface Program
Microelectronics & Computer Tech.
3500 W. Balcones Center Drive
P.O. Box 200195
Austin, TX 78720

Jerry Murch
Tektronix
Beaverton, OR

Erich Neuhold
Institute for Integrated Publication and
Information Systems
Gesellschaft fuer Mathematik und
Datenverarbeitung
Postfach 104326
Braunshardter Weg 11
D-6100 Darmstadt
West Germany

Donald A. Norman
Institute for Cognitive Science C-015
University of California at San Diego
La Jolla, CA 92093

Gary Olson
Department of Psychology
University of Michigan
330 Packard Rd.
Ann Arbor, MI 48104

Guests from:

Army Basic Research Office
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

Michael Drilling

Steve Goldberg

Michael Kaplan

Judith Orasanu

providing a useful and efficient representation of a device across a wide range of circumstances.

What are the kinds of external systems I might have mental models about? Keep in mind that we are talking about "how it works" representations that are not necessarily accurate or complete. Here's what seems plausible to me:

Physical World	The work on naive physics is about the kinds of mental models we have about how the physical world works.
"Engineered" Systems	These are the mechanical things that we have built, including computer systems. Most mental model research has focused on these.
Biological Systems	I certainly have "how it works" knowledge about various biological systems, such as the plants I try to sustain in my yard, the insects that attack them, and the animals that eat them.
Psychological Systems	This is knowledge of why people do what they do. Young children project psychological accounts of behavior onto far too broad a class of external systems (animism).
Self	A special case of a psychological system is one's self. I have many theories about how I work, probably many of them wrong.
Social Systems	I have "how it works" knowledge of social systems, such as collaborations, organizations, and societies. Understanding how these systems work is an important part of coping with the social settings in which I operate.

It's important to examine the entire range of entities that mental models could be about in order to better understand exactly what a mental model is. Much of the work on mental models has focused on "engineered" systems, where the question of how the system really works has an answer. We also have mental models of systems for which we cannot answer the question of how it really works.

Computing systems represent an interesting case. While they are obviously a classic case of an "engineered" system, they also have a number of properties that lead us to form mental models similar to those we hold for biological, psychological, and social systems. As Lucy Suchman (1987) has pointed out, computers in many ways resemble social objects. Interactive computing systems are reactive systems in much the same way that living systems are. Many computer-based artifacts are social in character. Computing is increasingly used for communication and coordination, inherently social activities. Indeed, as Suchman points out, many of the forms of interaction with computing systems are increasingly linguistic in character. As we move into the networked world where computing becomes a central part of collaboration, computers will acquire even more of the qualities of a social agent, moving our mental models of computing systems even further into the realm of the psychological and social.

How do we study mental models? One way is to teach explicit models to people and examine the effect on their behavior (Halasz & Moran, 1983; Kieras & Bovair, 1984). But we also need to study "natural" mental models, those induced by users from their interactions with systems. This is a much greater challenge, though we possess many tools that can be used to uncover knowledge representations (Olson & Rueter, 1987; Suchman, 1987).

References

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

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

Carroll, J. M., & Olson, J. R. (in press). Mental models in human-computer interaction: Research issues about what the user of software knows. In M. Helander (Ed.), *The handbook of human-computer interaction.* North Holland.

Gentner, D., & Gentner, D. R. (1983). Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner & A. L. Stevens (Eds), *Mental models* (pp. 99–129). Hillsdale, NJ: Lawrence Erlbaum Associates.

Gentner, D., & Stevens, A. L. (1983). *Mental models.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Halasz, F. G., & Moran, T. P. (1983). Mental models and problem solving in using a calculator. In *Proceedings of CHI '83 Human Factors in Computer Systems*, ACM, New York, 212–216.

Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1986). Direct manipulation interfaces. In D. A. Norman & S. W. Draper (Eds.), *User Centered System Design: New Perspectives on Human-Computer Interaction* (pp. 87–124). Hillsdale, NJ: Lawrence Erlbaum Associates.

Kieras, D. E., & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive Science*, **8**, 255–274.

Olson, J. R., & Rueter, H. (1987). Extracting expertise from experts: Methods for knowledge acquisition. *Expert Systems*, **4**, 152–168.

Shneiderman, B. (1983). Direct manipulation: A step beyond programming. *IEEE Computer*, **16**, 57–69.

Suchman, L. A. (1987). *Plans and situated actions: The problem of human-machine communication.* New York: Cambridge University Press.