Extract from Cognitive Factors in Design: Basic Phenomena in Human Memory and Problem Solving.

Tom Hewett
Departments of Psychology and of Computer Science
Drexel University,
Philadelphia, PA 19104, USA
Phone: +01-215-895-2461
FAX: +01-215-895-1333
hewett@drexel.edu

Introduction

Content.

Consisting of a series of "hands-on" demonstrations and exercises, supplemented with mini-lectures and thought questions, this tutorial introduces and illustrates a number of basic processes and phenomena of human memory and problem solving. Selected materials focus on helping the attendee develop both intuitive understanding and background knowledge. Emphasis is on high-level phenomena related to (a) problems of designing interactive systems or (b) to understanding human-computer interaction and the types of preconceptions users bring to that interaction.

Objectives.

The tutorial has five basic objectives. The first is to provide attendees with an intuitive understanding of a variety of phenomena through direct, "hands-on" (actually, "minds-on") exposure. Demonstrations and examples focus a participant's attention upon significant aspects of memory and problem solving processes which he or she may not otherwise ordinarily notice. The second objective is to help attendees develop a basis for making educated design choices when interpreting guidelines and when guidelines fail, conflict, or are non-existent. The demonstrations, examples, and mini-lectures create a general understanding of memory and problem solving. The third objective is to relate some of the phenomena being demonstrated to human-computer interaction. Occasional mini-lectures, examples, and thought questions in the notes will be used to bridge the gap between the demonstrations and general characteristics of human-computer interaction. The fourth objective is to assist attendees in undertaking self-directed study on these or related topics of their own choosing in cognitive psychology. The demonstrations and examples are chosen to supplement present or future textbook knowledge with insights based upon

direct experience. Suggestions for further reading are provided. The fifth objective is to provide those who may be asked to teach some of the psychological aspects of human-computer interaction with a useful set of teaching materials. All of the demonstrations have been classroom tested, work well, and can be done with minimal equipment.

Intended Audience.

The tutorial will be of benefit to those interested in human-computer interaction who have not had coursework in cognitive psychology, but who need access to knowledge of some basic psychological processes. It will also benefit who have done, or expect to be doing, self-guided reading in the area. The tutorial will benefit those who work with, or expect to be working with, cognitive psychologists or human factors specialists. Finally, since most of the demonstrations in the tutorial can be done with a minimal amount of equipment and work well for most students, the tutorial will also benefit those who expect to be teaching some aspects of the psychology of human-computer interaction. The tutorial is not intended for the human factors specialist, for the individual with extensive training in psychology, especially cognitive psychology, or for the individual seeking a state-of-the-art review of the latest research in cognitive psychology.

Presenter Background.

Tom Hewett is Professor of Psychology and Computer Science at Drexel University where he teaches courses on Cognitive Psychology, The Psychology of Human Computer Interaction, The Psychology of Human Computer Interaction Design, and Problem Solving and Creativity. He has been a visiting fellow, visiting professor or visiting researcher at the University of Vienna, Vienna, Austria, Tampere University, Tampere, Finland, Twente University, Hengelo, The Netherlands, Loughborough University, Loughborough, United Kingdom, The University of the Aegean, Syros, Greece, and the Battelle Pacific Northwest National Laboratory, Richland, WA. He is a member of the Association for Computing Machinery (ACM) and that organization's Special Interest Group on Computer Human Interaction (SIGCHI). He is also a member of the Society for Applied Research in Cognition, the Human Factors and Ergonomics Society and the IEEE Computer Society. Tom regularly offers a professional development tutorial on cognitive aspects of interactive computing system design to hundreds of interface designers at both conferences and in-house training sessions. He has taught a week long course on Human Problem solving for the User System Interaction program at the Technical University of Eindhoven, The Netherlands. In addition, Tom has made a variety of conference presentations, both invited and refereed.

Tom is a published courseware author and has worked the development and evaluation of several interactive computing projects, including a semi-intelligent, on-line assistance program for users of bibliographic database search services, and an interactive hypertext guidebook to the Macintosh and micro computing facilities at Drexel University. Some papers have described the

structure and implications of a taxonomy for thinking about instructional computing and have explored some of the pedagogical and institutional implications of universal student access to personal computers. . Other research papers have focused on issues related to the evaluation of interactive computing systems and the impact of evaluation on the design process. Tom chaired the ACM SIGCHI Curriculum Development Group that developed recommendations for undergraduate curricula and courses for Human Computer Interaction and has participated in several other curriculum development projects. He served for four years as Vice Chair for Operations of SIGCHI and was one of the general co-chairs for the CHI '94 conference held in Boston, MA, USA. He was one of the program co-chairs for the 4th Creativity and Cognition conference held in Loughborough, UK. Recent research activities have included working with a group of computer scientists interested in developing a scientific Problem Solving Environment (PSE) which integrates symbolic and numeric computing. Another recent project involved working with a multi-disciplinary team of engineers and computer scientists working on a project in networked engineering design. Tom is also a regular collaborator with researchers at the Creativity and Cognition Research Studios (formerly located in the Department of Computer Science and the College of Art and Design at Loughborough University in the UK, but recently relocated to the University of Technology, Sydney).

Acknowledgements.

The instructor greatly appreciates the feedback from several people who have taken the time to offer comments and suggestions about the content and usability of this tutorial and these notes, but special thanks are extended to Cynthia Rainis, George Casaday, Jared Spool and Verena Giller. A major revision of these notes was funded in part by NSF Grant CCR-9527130.

Preliminary Observations and Caveats

"Learners do not do what designers want them to do; instead they tend to get actively involved and to think and plan and solve problems (Carroll & Mack, 1985)."

Whether explicit or implicit, the designer of an interactive computing system has a model of the user. That model is the set of ideas, specifications, and assumptions about the tasks the system is being designed to accomplish, and about the intended users, their strengths and limitations, and how they perform their tasks. Problems with usability often arise simply because designers did not recognize discrepancies between their model and reality. Some computing systems have been designed as if humans were similar to computers (or to the designer's model of the system), seeing only what they are instructed to see, doing exactly what they are told, and waiting passively for instructions. In addition, users of these systems are treated as if they have only the memories created by experience with the system, and always understand exactly what the designer intends. Furthermore, after a user has been told something once, that is deemed sufficient. Memory is perfect. Redundancy is avoided. Similarly, system feedback to the user is either nonexistent or is cryptic and couched in terms appropriate to describing the internal status of the machine.

Users, however, often take an active role in interacting with computers, bringing previous experiences to bear in learning or understanding how to use the system. Some users can and will explore and experiment with the system. Also, **users can and will assign a meaning or interpretation to a wide variety of things, regardless of whether that meaning was intended by the designers**. In the context of human-computer interaction (HCI), consider the situation where a relatively new user sees a blinking message on the screen which simply says "**Working.**" One normal reaction is to interpret the message as meaning that the system is busily working on some task and that what is needed is to wait patiently until the the computer is done. The user might never consider the possibility that the system has just sent a one-time message signal to the terminal telling it to keep blinking. In general, humans have a great capacity for coming up with sensible interpretations of many things which may initially seem far fetched. Consider the sentence, **"I accidentally broke your wolf."** This sentence sounds quite strange at first reading, but providing a sensible interpretation is actually quite easy. In one context the sentence might refer to a porcelain wolf which once rested on your mantle before I bumped into it. In another context the sentence might refer to a pet wolf which had been accidentally dropped while being fished out of a tank of liquid oxygen. Bringing a context to bear in order to be able to interpret events is a very natural, routine human activity.

Similarly, when users find themselves in a strange situation without a clear sense of what is going on, they are, as suggested by Lewis (1983), in a fog of relevance. In that fog, **they can and do detect ambiguities which might remain unnoticed by designers.** As an example of our ability to detect ambiguity, notice that the sentence, **"Time flies like an arrow,"** can be interpreted in no fewer than five ways. The sentence could be a response to any of the following questions.

"How does time fly?" "Which flies should be timed?" "What do time flies like?" "How fast should you time flies?" "How do you time flies?"

In the context of HCI, suppose we have someone who became uncertain about how to proceed with a particular problem and is currently working through a task specific mini-help system which is coaching them through the task and which constrains choices to those appropriate for task completion. Among the things visible on the screen is a button labeled "**Finish.**" Does this button mean, "I'm done," "OK, you have the information you need, now complete the task for me," "Take me to the last piece of this task that I need to work on," "Now, that I've shown you a rough idea of what I want, make it a polished product," or, "Translate this whole thing into Finish (an arcane programming language known only to three software developers who work out of a garage in Intercourse, PA)"? How to respond with out being wrong?

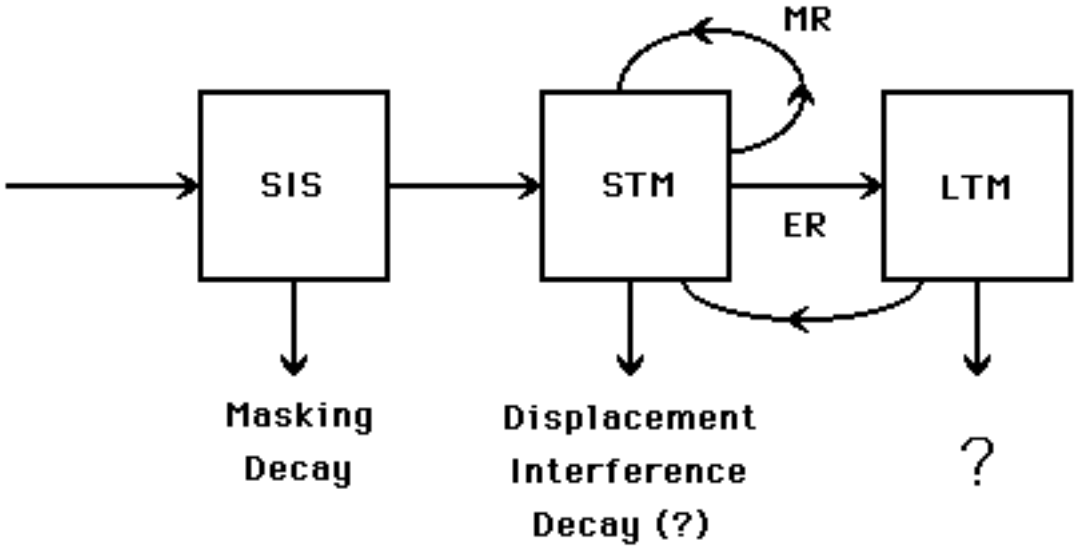
The emphasis in this tutorial is on demonstrations and exercises which highlight some of the quite remarkable things humans do in interacting with, learning about, and making sense out of the world around them. These things may seem quite ordinary because we do do them regularly, often without deliberation. Nonetheless, some of them are quite complex and not well understood. Thus, in one sense our interest here is in an engineering rather than a scientific approach. Before spending a great deal of time worrying about detailed principles for use in designing computing systems for usability, we want to scope the problem to within an order of magnitude.

Since this tutorial emphasizes development of an intuitive feel for the material, it is important to recognize that **an intuitive understanding must be informed and validated by research results**. Ever since its founding, Experimental Psychology has taught us not to trust personal hunches and intuitions about the reasons for human behavior. As you will see in the exercises and demonstrations in this tutorial there are a variety of ways in which our ordinary day-to-day intuitions about the reasons for our own behavior, or for the behavior of others, can lead us astray. Consequently, most demonstrations used in this tutorial are designed to replicate phenomena which have been studied under controlled laboratory conditions. In fact, several are drawn directly from research studies. However, the controlled conditions necessary for research are not necessarily appropriate for demonstration purposes (or for developing design ideas), so that occasional procedural license has been taken. Chosen to illustrate general phenomena, these demonstration may not work for everyone but all work reliably with many people, most of the time.

In summary then, one aspect of the overall goal for this tutorial is to help develop an approximate, intuitive feel for human memory and problem solving processes which will serve as a context for use in interpreting guidelines, in making design choices in the absence of guidelines, and in communicating with others on the development team. Another aspect of this goal is to develop that understanding in a context which illustrates some applications to HCI design and which facilitates further learning or teaching.

Human Memory

While there has been controversy about the accuracy of the representation of human memory used below (in fact, some researchers think we should forget it), we will use the following model of memory as an initial basis for organizing the discussion and examples. This model contains several components, a Sensory Information Storage (SIS), a Short-Term Memory (STM), and a Long-Term Memory (LTM). In addition there are several processes representing the transfer or manipulation of information. These include the processes of transfer of information into a memory, of loss of information from a memory, and of the types of rehearsal--Maintenance Rehearsal (MR) and Elaborative Rehearsal (ER)--involved in maintaining or transferring information. In addition, there are several factors which are thought to be responsible for the loss of information from memory.



In this model of memory, information enters a sensory information storage which is much like an input buffer in a computer and which holds a relatively direct neural representation of the sensory information. Once in the SIS, the information is either selected out for further processing, or it is lost, being "written over"or masked by successive information or by simply "decaying" (in approximately 200-250 milliseconds for the visual sensory memory, or iconic storage, and approximately 4-7 seconds for the auditory sensory memory, or echoic storage) if it is not refreshed.

The information passed to short-term memory is usually thought of as having been selected for further processing by an attentional mechanism (not shown here) which is guided either by changes in sensory stimulation, by expectations resulting from immediately prior processing or by expectations resulting from information already stored in memory, or by some combination of the three. (Neither the SIS nor the attentional mechanism will be discussed in this tutorial. The interested reader should refer to one or more of the recommended

texts on Cognitive Psychology for a discussion of attention and the various kinds of Sensory Information Stores--e.g., iconic storage in the visual system and echoic storage in the auditory system.)

Usually, STM is described as having a limited storage capacity (seven plus or minus two chunks) for a relatively brief duration (estimates range from 12 to 30 seconds without rehearsal) before information is lost through simple decay. (It got put it, nothing happened to refresh or preserve it, it wasn't there later when memory was queried). Another way in which information gets lost from STM is when new information displaces older information. (Think of watching a moving train from your office window, you can see only a limited number of boxcars moving along the track at any given time, those that have already passed out of view have been displaced by cars currently visible through the window.) Although interference between items does not strictly represent a loss of information from memory, if you query memory and retrieve an item similar to but not identical with one which was put in, there is a recall failure.

Information can be maintained in STM for periods of time longer than 20 seconds with maintenance rehearsal (MR). However, this simple repetition of material does not appear to be very efficient at transferring information into long-term memory (LTM). Rather, elaborative rehearsal (ER), which will be illustrated in the tutorial, appears to be the most effective set of processes for the transfer of information into long-term storage. Although the information in STM is usually thought to be represented by one of a relatively limited number of codes, there are several reasons to believe that any organizational structure in LTM is accessible as a basis for forming the chunks that are held in STM. This also means the capacity of a chunk is variable, depending upon the complexity of the knowledge structure upon which it is based.

LTM, which in many ways is an enigma, has a large capacity for storage of information for long periods of time. There is, however, no easy or obvious way to determine the limits of how much information can be stored, or for how long it can be stored. Several types of information are represented in LTM, including such things as facts and events, motor and perceptual skills, knowledge of physical laws and systems of mathematics, a spatial model of the world around us, attitudes and beliefs about ourself and others, etc. This information is more or less well organized, in a variety of ways, and varies in its accessibility as a function of several factors. Although there has been an ongoing controversy over whether all of these different types of knowledge can be reduced to a single underlying code or representation system, recent brain scan data suggest there are at least two different systems or forms of knowledge representation in the human brain.

The factors determining accessibility of the information in LTM include such things as the conditions which existed at the time the information was stored, the recency of its last use, its degree of inter-relationship with other knowledge, its degree of uniqueness relative to other information, etc. While most authors stop short of claiming that information once stored in LTM is not forgotten, most discussions of failure to recall information from LTM focus on explanations such

as interference, the absence or inappropriateness of retrieval cues, or some type of organic dysfunction such as brain damage. While there are many researchers who argue strongly for the existence of a process of decay of information in long term memory there is still controversy of the the degree to which failure to recall information from LTM should be attributed to decay rather than interference or lack of appropriate retrieval cues.

There is an alternative model of human memory which holds that there is no longer a need to postulate a STM. Rather, there is a working memory system which is part of the larger memory system and not a separate memory. This working memory functions much like a workbench onto which only a limited number of chunks of information can be placed. These chunks are either activated from LTM or are part of the information being extracted from the world. Different degrees of persistence of information in memory are thought to be a function of the depth to which information has been processed. The greater the depth (or elaborateness) of the processing, the longer the retention period. In this model of memory, chunks are basically organizational units in LTM and the working memory capacity limitation is a limitation on how much information can be actively scanned or maintained in working memory at any given time by the attentional mechanism. While some evidence exists to support each view of memory, the bulk of evidence seems to favor working memory over STM (Anderson, 1995).

It is, however, not clear that differences between STM and working memory are critical for making interface or interaction design decisions in most circumstances. Consequently, the focus in this tutorial is on demonstrating phenomena with which any theory of memory must deal. For example, regardless of whether one thinks of the seven plus or minus two limitation as the result of limited capacity memory or as a result of a limit on the amount of information which can be activated and maintained in an active state at any given time, it is clear that there is a working memory capacity limitation. Similarly, regardless of whether one thinks that rehearsal processes have different degrees of depth and elaboration or that there are two different kinds of rehearsal processes, it is clear that elaborative rehearsal is more effective than maintenance rehearsal in insuring that the information will be accessible for a long period of time. Several of the recommended readings should enable the interested reader to pursue further the theoretical issues in the study of human memory (and to find references to the technical literature on memory). Other recommended readings should enable the interested reader to explore situations in which these distinctions may be critical in making to design decisions (and to find references to the technical literature on such design considerations).

The Demonstrations

Since some of the following demonstrations and exercises might be affected by knowing what is coming, this section of the notes is organized with the materials for all demonstrations grouped together, including some spaces for notes, etc. Integrated in with these demonstrations are some thought questions about HCI. Next there is a section of the notes containing an outline and a set of observations. These observations summarize one or more major points of the demonstration or exercise and provide a reference to the original material from which it was drawn.

For most of the memory demonstrations you will be asked to remember several things. Try your best. The effectiveness of these demonstrations is enhanced by the degree to which you work at remembering the material.

Thinking about HCI: User knowledge structures are important.

How might you use your knowledge of memory and the importance of knowledge structures to help you understand memory failures, errors or other types of user behaviors? Several years ago there was a series of classes conducted to introduce some University faculty to microcomputers and their potential. The machines for this class were 64K Apple IIs with a single external floppy and an Apple III monitor. Both the floppy and the monitor came as separate units and each had a red LED which lit up when the peripheral was in use. There was no hard drive. The different sections of the class were being team taught.

In one section there was a Ph. D. Mechanical Engineer of several years experience who adopted a rather interesting behavior. With only one floppy drive the word processing program had to be loaded in memory, the program disk removed, and the data disk inserted. In the early part of the word processing class period it was necessary for the students to swap floppy disks in and out of the drive several times. The visiting instructor noticed that every time this Engineer had to swap disks he would first reach out and turn off the monitor. Once the disk transfer was completed he would turn the monitor back on again. Eventually the class participants were at work on an exercise. In a private chat, the instructor asked the Engineer why he turned off the monitor before swapping a disk in and out of the drive. The Engineer provided a perfectly rational explanation. He explained that on the first day of class they had been told never to try and take a floppy disk out of the disk drive when the red light was lit. He then pointed to the red LED beside the monitor switch and explained that the only way to get it to go out was to turn off the monitor. **While it might not be possible to predict this particular case in advance it is reasonable to assume that if anything can be misunderstood by a user in a fog of relevance, it probably will be misunderstood by some user somewhere, sometime.**

Memory and HCI: A summary.

In considering the nature of human memory and the processes involved in storage and retrieval of information, it is clear that humans interacting with computers can and do use existing memories and memory structures to assign a meaning or interpretation to a wide variety of things, regardless of whether that meaning was the one intended by the designers. Consequently, it is important to understand and take account of existing knowledge structures and how people think about domain problems and the problems of interaction. In addition, designers need to take account of the fact that there are a limited number of "chunks" of information with which a user can actively cope at any given time. While the size of those chunks can be affected by the development of additional new knowledge structures, that growth of knowledge requires that the user be actively engaged in elaborating and assimilating that knowledge. Finally, the demands upon human learning and memory can be reduced by providing appropriate mnemonic cues in the interface and the flow of interaction.

Problem Solving

Thinking about HCI: Breaking the set or escaping from functional fixedness.

Norman (1988) pointed out that one can learn a lot about design failures from watching users cope with poor design (poor meaning non-functional in some way important to the user). Sometimes these coping solutions can be quite creative, demonstrating an ability to break a problem solving set or escape from functional fixedness. Consider the normal function of the plastic beverage glass used by airlines. A few years ago I flew to England on British Air. Being seated near the front of my section of the aircraft on both flights I had the opportunity to notice that the nearly full water pitcher on the beverage cart had an empty plastic water cup in it, floating on the water. I also noticed but did not appreciate the significance of the water pitcher's shape. Viewed from above it was basically diamond shaped. The pouring lip and the handle were at opposite ends of the long axis of the diamond, with the lip at the top of one vertical corner and the handle attached along the other.

In response to my query, the stewardess explained that the floating plastic cup helped keep water from splashing out over the pouring lip in reaction to stop-start motions of the cart that get the water sloshing back and forth along the long axis of the diamond shape. She also observed that the water seemed to run up the sides of the cup rather than out over the lip of the pitcher. A further query revealed that splashing water was perceived by British Air stewardesses in general to be much more of a problem for this particular pitcher design than for the one it had replaced. Floating the cup also made it possible to start with a relatively full pitcher and not have to make as many trips to refill the pitcher. In response to my final question the stewardess indicated that she had not come up with this solution, rather she had gotten it through the stewardesses "grapevine" and was pleased that it worked.

Turning to the realm of HCI, one can also learn a lot from the ways in which users create new uses or extend the software in ways not envisioned by the designer. Effectively these people are also breaking a problem solving set or escaping from functional fixedness to develop a new schema. Their novel uses

suggest ideas for new functionality. These uses can often suggest ideas for effective low-cost solutions to problems which might otherwise require an expensive fix or go unsolved. As an example, consider the electronic spreadsheet. Originally designed to replace the large paper worksheet used in accounting and financial planning, the electronic spreadsheet (e.g., Excel, Lotus 1-2-3, etc.) can be used interactively to explore functional relationships among a number of parameters. In addition, it can perform a variety of mathematical operations (e.g., Arganbright, 1984; Williams & Williams, 1985).

Consisting of a matrix of cells arranged in rows and columns, the typical spreadsheet offers a number of programming capabilities, including the ability to manipulate strings and do iterative calculations. In each cell the user can enter text, data, and formulas or instructions which tell the program to look up a value in another location and use it in some way. Furthermore, even in the less powerful spreadsheet programs, built-in functions can be combined to produce plots, providing both a graphical and a numerical method for illustrating relational concepts.

A spreadsheet can be used to create an end-user interface to the program. Templates created by programming the spreadsheet with certain useful characteristics enable the end-user to accomplish a particular task without having to worry about the details of programming the spreadsheet. Quite often these templates, are custom tailored for the use of end-users who neither need nor want the full functionality of the underlying programming environment. As one indication of the utility of spreadsheet programs in solving particular classes of problems, templates for popular spreadsheets such as Microsoft's Excel and Lotus 1-2-3 generate vertical markets worth thousands of dollars annually to their developers. Most of these uses extend far beyond the original problem the electronic spreadsheet was intended to solve (elimination of tedious re-calculation of columns of numbers).

While most spreadsheet templates are used for financial modeling it is possible to create templates to model a much wider variety of things. For example, in a series of papers, Hewett (1985, 1986a, 1986b) illustrated some ways in which the spreadsheet could be used to create templates for students which would enable them to explore simulations of some of the neural circuits and

networks involved in phenomena found in the early stages of visual information processing. One of these papers (Hewett, 1985) describes a spreadsheet-based model of the type of neural network responsible for the visual effects of Mach bands. (Mach bands are a series of adjacent, increasingly brighter bands of gray. Each band appears to differ in brightness from one edge of the band to the other. However, each band is, in fact, uniform in brightness across its width. An example appears in Appendix B.)

The computational capabilities of most spreadsheet programs will easily simulate reasonably complex neural models involving networks with multiple layers of neurons, multiple circuits and multiple types of circuits. In addition, the more powerful spreadsheet programs can handle relatively powerful and complex mathematical models of a neural network (e.g., Halff, 1987). It would require a considerable investment of time and money to build a special purpose neural modeling program which could provide this kind of flexibility in both the creation of an interface and in the representation of a neural model.

There are other novel areas for spreadsheet use that do not involve modeling or simulation at all. One strategy many busy people use for keeping track of things that need to be done involves keeping a “to do” list. However, making and re-writing lists on paper can be time consuming and frustrating. The use of notes or cards may not be much better since the notes may be unorganized or easily misplaced. Failing to get a task completed on time because one did not find the reminder until too late can be very frustrating. It can also be frustrating to write down a good idea and then not be able to remember where the card or piece of paper has been filed.

Using a spreadsheet based list eliminates the need for re-copying lists and can prevent the loss of notes. It also offers additional advantages. For example, a number of time management consultants (e.g., Lakein, 1973) propose that tasks vary in their importance and in the urgency of their completion. Further, one of the biggest problems with personal and professional time management comes from letting others determine the importance and urgency of one's tasks without consideration of the goals one has set for one's self. Consequently, many time management consultants recommend both a "to do" list and frequent re-evaluation of the importance and the urgency of the tasks on the list. Combining

the capabilities of an electronic spreadsheet with some basic principles about decision making and about the optimal procedures for making rating scale judgments, it is possible to develop a tickle file which avoids the disadvantages of written lists or notes, and which produces an effective program for managing one's tasks and priorities (Hewett, 1988). As Landauer (1987) suggested, the challenge for designers, "... is not to just think up a new gizmo..., but to analyze systematically exactly what people need now that they don't always get, why they can't get it, and what would help them (p. 334)."

Some rules of thumb for effective problem solving.

1) **A problem is something that doesn't solve easily and so you may not know what you need to know.** One implication of the definition of a problem used here is that most of the things we do are basically problem solving, even though we may not think of those things as problems. For example, you often go to work in the morning. This activity fits the definition of a problem. When you wake up there is a gap between your initial state and your goal state--being at work. There are a set of operators--e.g., getting dressed, etc.--required to transform your initial state into the goal state, and there are restrictions on those operators--e.g., while stockings may go on either before or after slacks, stockings go on before rather than after shoes. Typically this whole procedure is done routinely and without a great deal of difficulty. When solving is difficult you may need more knowledge.

2) **When a problem you think you can solve doesn't solve, either you don't have the knowledge you need or you have misrepresented some piece of the problem.** In part, the difficulty with solving some problems is that we have a large number of problem solving schemas which are used quite automatically to deal with routine problems. The problem is posed, a schema is invoked, we decide on the relevant information, and bring the solution procedures to bear. Thus, when a problem doesn't solve as easily as it should, either you do not have the information needed to solve it, or you have misrepresented the problem in some way. The nine dots problem illustrates a situation in which people typically impose a condition upon the operators which is not required by the problem and which inhibits solution.

3) **If at first you don't succeed, try something different.** As illustrated in several examples in this tutorial, repeated, persistent attempts to solve a problem using the same strategy or procedures may consistently fail. If so, either new knowledge is needed or a new look at the problem representation and the assumptions in that representation. In other words, step back, take a deep breath, and try something different. If the difficulty does not seem to lie in the assumptions of the problem representation, look for a way to re-represent the problem. Look for a problem isomorph.

4) **Tackle any piece of the problem you do understand and the remaining pieces may fall into place.** Often, complex problems have a number of interrelated pieces. Sometimes the relationships among the pieces become clearer as one part of the problem is being solved and the solution or the solving can be used as a basis for learning something more about the knowledge required to solve the problem or about the structure and parts of the more complex problem.

Thinking about HCI: Making the reasonable assumption that computer users are problem solvers, can you think of ways to facilitate their problem solving activities by using these rules of thumb for effective problem solving in structuring such things as documentation, tutorials, on-line help libraries, cue cards, error messages, and error trapping routines? (i.e., How might we use these rules of thumb as design guidelines for interface and user interaction design?)

Thinking about HCI: There are two commonplaces which one often hears repeated in different contexts. "Users typically do not know or understand their own needs." "Users are always right." Notice that one of the implications of the material in this section on problem solving is that we are all experts at many things, including those tasks at which we have done for years that are now being computerized. Similarly, just because we can't easily say what we know about such tasks doesn't mean we don't know it. What do these observations suggest about how we can make sense out of the idea that users are right but don't know or understand their own needs? What are some of the consequences of these observations?

Thinking about HCI: As designers and developers we have schemas about interface, task, and interaction design, what is the most likely effect on our own thinking of having these schemas? (i.e., Are interface designers and programmers likely to be unaffected by problem solving set and functional fixedness?) Can you think of a better way to surface your own implicit assumptions than to encourage a novice user to use your software while you just watch (and bite your tongue if necessary to remain silent)?

Thinking about HCI: If we consider an interface to be an assistive representation which helps designers to think and talk about a set of design ideas and which can be used as a tool in helping developers to learn about the ways in which the users think about their tasks, activities, jobs and the development of the interface, can you think of some of the reasons why prototypes (especially paper mock-ups) might be useful things to develop before committing major resources to implementation of a particular design?

Thinking about HCI: What are the advantages and disadvantages of considering an interface and interaction style to be an assistive representation which helps users to think and talk about a set of problems and which can be a tool in helping them learn to think about new ways of doing their tasks, activities, and jobs?

Problem Solving and HCI: A summary.

In considering the nature of human problem solving it is clear that designers need to take account of the fact that users will bring to bear established problem solving strategies which are developed from having solved problems that have been frequently encountered in the past. While these established strategies can be used to facilitate usage they can also create process blockages which may only be solved by changing the way in which the problems of interaction are presented to the user or by creating a working environment in which the user has the flexibility to redefine, reconceptualize or re-represent the tasks or problems which the software is being used to solve. In particular, the software should make it possible, without penalty, for the user to suspend current activities and be able: a) to seek more information about the software or the problem domain; b) to re-examine or re-formulate some aspect of the interaction problem or domain problem with an eye towards modifying that problem in some way; c) to completely re-represent or re-formulate the entire interaction or domain problem; and d) to break up the interaction or domain problem into manageable sub-problems in ways that allow for productive recombination of elements or components.