Cognitive Processes Involved in Solving Information Systems (IS) Design Problems

E. Sue Weber
The University of Arizona

Cognitive Processes Involved in Solving Information Systems (IS) Design Problems

By E. Sue Weber

Department of Management Information Systems
The University of Arizona
Tucson, Arizona

ABSTRACT

The author characterizes systems analysis and design as a cognitive problem-solving process and suggests that many implementations fail because Information Systems (IS) designers do not adequately understand the cognitive processes involved. The author explores problem understanding as well as the dynamic relationship between it and plan development and points to areas in which research will not only increase the probability of successful IS implementations but will also contribute to the theoretical foundations of IS.

Introduction

Systems analysis and design can be characterized as a problem-solving process. If we look at the history of failures of Information Systems (IS) projects, it seems fair to say that IS designers do not appear to be particularly able problem solvers. We have an extensive literature which details the failures of the past and attempts to identify problem areas where designers need to develop sensitivity and expertise (Bostrom and Heinen, 1977); Ginzberg, 1981; Keen and Gerson, 1977). We have a nascent literature which has begun to explore cognitive issues in the design process (Boland, 1978; Malhotra, Thomas, Carroll, and Miller, 1980). Here, the ways in which knowledge affects the problem-solving performance of software designers have been investigated (Bonar, Ehrlich, Soloway, and Rubin, 1981; Jeffries, Turner, Polson, and Atwood, 1980) as well as the relationship of the problem-solving behavior of IS designers to successful performance (Soloway, Ehrlich, Bonar, and Greenspan, 1982; Vitalari and Dickson, 1983).

This new line of inquiry is very promising. It is the thesis of this paper that we will significantly improve our ability to design successful IS when we better understand the cognitive processes involved in solving design problems. This understanding will contribute not only to improved practice, but also to IS theory. In this respect, initial research results already appear to necessitate a restructuring of our models of the IS design process.

Human Information-Processing

THE FRAMEWORK

Knowledge and its organization is vital to successful problem solving in complex, realistic situations (Chase and Simon, 1973; Chi, Glaser, and Rees, 1982; DiSessa, 1983; Larkin, 1983; McDermott and Larkin, 1978; Simon and Simon, 1978). As a result, it is important to understand how individuals perceive, learn, store, and retrieve information.

The human information-processing framework describes mental events in terms of transformations of information from input to output. Information in the form of some physical energy is received by sensory receptors sensitive to that particular form of energy, transformed into nerve impulses, and sent to a sensory register in the central nervous system. In order to avoid information overload, most of these signals are blocked and are processed no further. This filtering process is affected by the experience and prior knowledge of the individual and is, therefore, highly personal. Only a small subset of the original stimuli is kept for representation in short-term memory (STM) or active consciousness. Here the information available to the individual is limited and must be kept active or it will be lost. New information in STM can be integrated with known information, recalled for this purpose from long-term memory (LTM), and then stored in LTM as either declarative or procedural information.

Once information has been stored in LTM, it must be retrieved to be used agaoin. In conscious response, information flows from LTM to STM and then to a response generator which organizes the individual's responses and guides the effectors, those involved in design usually being the hands, arms, and voice of the designer.

KNOWLEDGE REPRESENTATION

The knowledge we possess is thought to be of two types. Declarative information is all the facts, generalizations, theories, and personal memories that we have ever stored in LTM. Procedural knowledge is knowledge of how to do something. procedural knowledge appears to be important in performing competently in familiar situations; declarative knowledge, on the other hand, appears to be important in deciding what procedures to use in novel situations. Both are important to solving design problems. Research has shown that successful problem solvers possess a great deal of domain-specific information. They often possess more declarative knowledge than unsuccessful solvers. More importantly, they appear to possess more procedural information. The critical difference, however, appears to derive from the organization of their domain-specific knowledge (Chi, Feltovich, and Glaser, 1981; Chi, Glaser, and Rees, 1982; Gagne, 1985).

In order to discuss differences between successful and unsuccessful problem solvers, I shall briefly present three forms of knowledge representation that fit with the architectural constraints of the human information-processing system. There are many views of how LTM is structured. I accept the view that all of an individual's declarative knowledge is represented in a propositional network. This network is a hypothetical construct; we do not actually know how information is represented in a physiological sense. According to the construct, a basic unit in the human information-processing system is the proposition. It corresponds to an idea but is more abstract than a sentence. We remember the gist of what was said rather than the exact wording. Propositional networks are sets of interrelated propositions, all ideas ultimately being associated with all other ideas. Propositions sharing ideas are more closely related than those that do not.

Procedural knowledge is represented by productions or condition-action rules. A rule's *if* clause specifies the conditions that must be present; its *then* clause specifies the actions that occur when those conditions are met. Productions are thought to be interrelated in sets called production systems. The result of the application of one production in the set provides the conditions needed for another production in the system to apply. A sequence of related actions takes place automatically.

Images are analog knowledge representations. Research suggests that people use mental imagery when tasks require spatial manipulation of information. There are informal reports that people also use imagery in thinking about abstract relationships. Designers are often said to use mental imagery to solve problems. Reresentations serve different functions in STM and LTM. In the former, they are manipulated and transformed. In the latter, they are preserved. There is general agreement on whether it is used in LTM. It is, however, a compact way of manipulating information within the constraints of active consciousness.

One of the important points about this hypothetical construct is that knowledge is represented in forms that reduce the burden on STM. Propositional networks keep related knowledge accessible. When we think about an idea, related ideas come to mind. Production systems reduce the burden by letting control flow automatically from one step in a sequence to another. Because of its automaticity, the sequence of procedural operations require little space in STM. It is thought that mental imagery may require minimum space because of pre-processing by the sensory receptors (Y.R. Wang, personal communication, August 1, 1985).

INTERACTIONS BETWEEN DECLARATIVE AND PROCEDURAL INFORMATION

LTM can be thought of as a propositional network in which procedural units are embedded close to related propositions. This model of LTM reflects the close interaction between declarative and procedural information in learning and in problem solving. Declarative information interacts with procedures in everyday problem solving by providing the data they need. It also appears to mediate the insight required for creative problem solving. In acquiring a new procedure, learners often represent it to themselves in declarative form until the steps become automatic (Anderson, 1982). Other learners have developed procedures for learning declarative information (Weinstein and Mayer, 1985).

LEARNING DECLARATIVE INFORMATION

Learning encompasses four sub-processes: 1) selection, 2) construction, 3) integration, and 4) acquisition (Weinstein and Mayer, 1985). During selection, the learner attends to environmental stimuli and transfers this information to STM. In construction, connections are built in STM between ideas contained in this information.

During integration, the learner actively searches for related prior knowledge in LTM and transfers it to STM; in addition, external connections are built between incoming and prior information. In acquisition, the learner actively transfers integrated information from active consciousness into LTM.

Organizational strategies are important in selection and construction. Organized incoming ideas activate related propositions in LTM. Activiation spreads along the links of the propositional network to related propositions. By classifying an encounter with an unhappy user as an instance of user resistance to change, a designer would trigger facts related to resistance and change already organized and stored in LTM. Organization enhances the memorability of information a great deal. It may keep the spread of activation of the propositional network to the relevant area of LTM; and it may provide pointers in STM to the relevant areas of LTM. Although retrieval from memory is not perfect, it occurs systematically and reflects the organization in memory of the individual's knowledge (Greeno, 1973). There is growing evidence from a number of studies that successful and unsuccessful problem solvers differ primarily in the quality of the organization of their domain-specific knowledge (Chi, Glaser, and Rees, 1982).

During integration, these new propositions and related prior information may stimulate the generation of other new propositions. All the new propostions, whether perceived in the environment or elaborated by the learner, are stored close to the prior knowledge activated during learning. The resulting elaborations provide alternate pathways for retrieval so that if one pathway is blocked, another can be used. These alternative routes may be one reason why designers have found problem solving by analogy so effective and efficient.

LEARNING PROCEDURAL INFORMATION

An important difference between successful and unsuccessful problem solvers is that the former have much more domain-specific procedural knowledge. There are two principal types of procedural knowledge: pattern-recognition and action-sequence procedures. Pattern-recognition procedures are essential to the ability to recognize and classify patterns of stiumli. Many patterns are learned through generalization and discrimination. In generalizing, we respond in a similar way to stimuli that differ. Discrimination, on the other hand, restricts the range of situations to which a procedure applies and is stimulated when a known procedure does not work (Gagne, 1985).

Action-sequence procedures are sequences of actions coupled with pattern-recognition procedures, recognition of patterns being a necessary condition for the correct application of rules. Learning action sequences is usually slow and awkward. First, the learner represents a series of actions in propositional form. Then a procedural representation of the action sequence develops with practice in trying to produce the action sequence. Computer programming in an action sequence that is typically learned in this way. Procedural knowledge is developed only through practice and feedback. Consequently, expertise in solving design problems takes years of experience.

Problem Solving

A DEFINITION

There are many definitions of problem solving. I use a general definition which equates normal directed thinking and problem solving. In this view, problem solving involves the active manipulation of perceived, learned, and retrieved informaion. It can be conceived of as a search of a problem space.

SUCCESSFUL PROBLEM SOLVING

Many (Jeffries, Polson, Razan, and Atwood, 1977; Larkin, McDermott, Simon, and Simon, 1980; Simon and Simon, 1978) have explored the differences in knowledge between those who solve problems skillfully and often correctly and have found differences ih their search strategies—processes used to select the operators to apply to a problem. Others (Chase and Simon, 1973; DeGroot, 1966) have found that successful and unsuccessful problem solvers search in similar ways but that successful problem solvers are able to represent problems in ways that make search easier. In solving complex dewsign problems, much of the search occurs in LTM. Memory functions as a second environment, parallel to the sensorial environment, through which the problem solver directs the search. In any particular problem-solving episode, much of the search is guided by the content and organization of the information acquired by the individual to that point. Problem solvers use task-specific strategies to the extent that they have been able to ascertain the underlying structure of the task. Otherwise, they use general search strategies.

Problem difficulty is a function of the amount of structure of the problem, the power of the chosen solution methods, and the knowledge available to the solver for constraining the problem space. Unfortunately, the IS designer must solve problems where the goal state is poorly defined, the operators are not given, and the problem space is enormous. In addition, the IS knowledge base is incomplete and solution methods are weak (Vitalari, 1981). With research into the knowledge required for successful IS design, we would eventually be able to enhance the problem-solving skills of IS designers through the development of knowledge-based design tools.

PROBLEM UNDERSTANDING

While we have not agreed on the best methods for solving IS design problems (Zave, 1984), we do agree that we must begin by understanding them (Boehm, 1981; DeMarco, 1982; Ginzberg, 1981; Keen, 1981; Weber, 1984). Yet, when we agree that problem understanding is an essential first step, our consensus is somewhat meaningless. We speak of problem understanding, implying that there is a single global problem that must be understood before it can be solved. This is not true. There is a myriad of problems and attendant subproblems that must be addressed in designing a system. We speak of understanding, but we have not defined what we mean when we say that an individual understands a design problem. Nor have we explored empirically the dynamic relationship between problem understanding and solution.

PROBLEM REPRESENTATION

We can, in fact, be said to understand a problem when we are able to construct an internal representation in memory of the problem space and a set of operators for moving from one state to another in that space (Simon, 1981). Therefore, the first step in solving a design problem is to represent the problem. Problem representation can be a very difficult task requiring a great deal of experience or expertise. Yet, because it determines which knowledge will be activated, it effects the ease with which a problem can be solved and even whether it can be solved at all (Chase and Simon, 1973; DeGroot, 1966). Larkin (1983) argued that in scientific problem solving an important part of expertise is the ability to represent the problem and, in so doing, udnerstand it. She found that successful solvers usually construct a sophisticated problem representation while unsuccessful solvers use primitive representations. Since a problem representation is constructed on the basis of the solver's domain-related klnowledge and its organization, the relationship between expertise and success is not surprising.

PATTERN RECOGNITION

Designers learn problem patterns through experience. With time these patterns become more and more refined. Gradually they are also linked to action-sequence procedures. To understand a new problem situation, an experienced designer scans the environment for familiar patterns (Vitalari, 1981). Much of our IS literature is aimed at providing designers with declarative knowledge and pattern-recognition procedures that we hope will contribute to the efficient management of the designer's mental resources and, as a result, contribute to the successful management of the project. Keen and Gerson (1977) and Markus (1983), for example, have helped build a rich body of patterns to aid in the interpretation of political events which often occur during IS implementation. Much of designers' practical experience results in the development of action-sequence procedures which are linked with pattern-recognition procedures.

Not only do the patterns triggered by environmental features tell us what to see and where to see it; they also direct retrieval from memory. We actively seek information relevant to the current situation not only from the problem situation but also from our memories. Rumelhart and Ortony (1977) suggest that the process of remembering is similar to that of perceiving. In remembering, however, memory is the data source rather than sensory experience. In this connection, it is important to realize that the memories on which this process is based are not fragments of the original sensory input, but are instead fragmentary representations of our initial interpretation of that input.

A designer may have successfully managed a project fraught with political issues in the past. When environmental clues trigger the retrieval of a problem pattern, what will be recalled will be those features of the original problem that are consistent with the pattern. What we think we see is thus based upon comparison with complex collections of previous experiences and expectations. We cannot, as designers, believe our own eyes and must take care in data analysis and in hypothesis seeking to counterbalance the effects of the perceptual filter. Designers contribute from their own experience, either by addition or by subtraction, to their perception of the phenomenon before them. As a result, they need rich, wide experience as well as the capacity for flexibility in thought (Archer, 1964).

QUALITATIVE ANALYSIS

McDermott and Larkin (1978) have developed a multistage model of the process of problem representation. They speculate that a solver's first step is merely to translate the problem as originally presented into a form which clearly describes the situation specified in the problem. A second stage involves the drawing of a sketch or diagram of the situation to represent the literal objects in the problem and their relationships. The third stage is a qualitative analysis of the problem. Qualitative analysis, which novices appear to skip (Chi, Feltovich, and Glaser, 1981; Larkin, 1980), is the process by which experts are thought to construct a rich and rather abstract internal representation in which objects and relations are linked to solution procedures. It exploits fully the problem solver's specialized knowledge of the relevant domain to generate a theoretical problem description. In physics problem solving there is also a fourth stage in which equations are generated (McDermott and Larkin, 1978).

Chi, Feltovich, and Glaser (1981) have developed a model in which there is more interacting among the stages than was proposed by McDermott and Larkin (1978). They hypothesize that experts, as opposed to novices, do not construct a literal problem description as their first step. They believe that for experts, qualitative analysis occurs immediately, even if briefly, and that it reoccurs often during the process. For the expert, solving a problem seems to begin with a provisional identification of the problem type after a gross preliminary analysis in which problem features are matched with known patterns. The knoowledge useful for the problem is indexed when a given problem is categorized as a specific type. Once a potential category is activated, declarative and procedural data for solving the problem are available within seconds. One becomes an expert in a field by acquiring and organizing the declarative and procedural knowledge necessary for success in that domain. The knowledge bases of experts are organized in ways that are well suited to the demands of domain-specific problems. Their hierarchical organization helps manage the constraints of STM by allowing experts to keep all appropriate hypotheses about the nature of the problem in mind while considering additional data. Experts become able not only to recognize situations and to provide information about them but also to use powerful skills to deal with problem situations as they arise (Simon, 1981). Consequently, solving a problem becomes a matter of categorizing the problem into one or more problem types and using the available knowledge.

Information Systems Design

DESIGN KNOWLEDGE

Jeffries, Turner, Polson, and Atwood (1980) found that experienced designers acquire a gret deal of well-organized abstract knowledge about design and design processes. They found it guides the generation and evaluation of design alternatives, prescribes functions that design components must satisfy, and helps determine what design components must satisfy, and helps determine what design elements to consider next. In contrast to novices, experienced designers generate and evaluate more alternatives, are more methodical in their expansion of design components, and consider more of the important functions that need to be accomplished.

DECOMPOSITION

IS designers are expected to handle ill-structured problems as a matter of course. Polya (1957) has recommended that a solver in such a situation first attempt to understand the problem. The problem as a whole should be made so clear and should be so well impressed upon the solver's mind that the solver can attend to component details with no fear of forgetting it. A grasp of the entire problem helps drive the process of decomposition; it also helps the solver simplify and concentrate on those details which are apt to contribute to problem resolution. One of the ways in which expert software designers differ from novices is in their ability to deal with details at many levels of decomposition and yet never lose track of their ultimate goal (Jeffries, Turner, Polson, and Atwood, 1980).

According to Polya, whose recommendations derived from years of experience in solving complex problems, the next step is to break the problem down into its components. A single design problem is a complex of hundreds of subproblems, each of which can be resolved so as to produce a cluster of acceptable solutions. Jeffries, Turner, Polson and Atwood (1980) found that when expert designers perceive a particular problem to be complex, they decompose it into a group of more manageable subproblems, eventually reaching a point where all subproblems have known solutions. The process of decomposing complex problems into manageable units seems to be central to the task of design in any field and mastery of decomposition appears to distinguish expert from novice designers (Archer, 1963–1; Jeffries, Turner, Polson and Atwood, 1980).

The difficult task is to reconcile the solutions of the subproblems with one another. When the optimum solution of one subproblem entails the acceptance of a poor solution for another, the designer must rank order the complex of solutions. Eventually subproblems and their solutions must be coordinated and linked together in a coherent design which must then be implemented (Archer, 1963-1). Malhotra, Thomas, Carroll, and Miller (1980) found that generation of design solutions seems to consist of attempting to find design elements to meet functional requirements and then tying them together into a coherent design.

PROBLEM SOLVING BY ANALOGY

The second task that Polya (1957) set the problem solver was that of breaking the problem down into its components. When the principal parts were distinctly arranged and clearly conceived and when one's memory seem responsive, the solver's first question was to be: "Have I seen this problem before?" IS designers do seem to ask this question first (Jeffries, Turner, Polson, and Atwood, 1980; Malhotra, Thomas, Carroll, and Miller, 1980). It is an extremely efficient way to solve problems and it enormously reduces the amount of problem solving effort.

Problem solving by analogy is a clear case of routine, rather than creative, problem solving. When a problem representation generated for a given subproblem is recognized as being analogous to an already familiar algorithm, that algorithm is evaluated for applicability in the current situation. If it is found to be appropriate, it is debugged and incorporated into the developing solution. Turner (1983) believes that designers first try to solve IS design problems by using this strategy and that they abandon it and attempt to restructure the problem only as a last resort. Problem solving by analogy is an extremely efficient way to solve problems and it enormously reduces the amount of problem-solving effort. But it has its dangers. If we habitually solve a given type of problem in one way, the solution used becomes automatic. Once it is automatic, learning alternative solutions is difficult because the conditions that trigger those alternatives never enter STM. When this question does not elicit a ready response, the problem solver should try to see the problem in a different way by looking for traces of other patterns among the stimuli in the environment. By varying the surface features of the problem a new pattern may emerge that will trigger an appropriate problem category.

A preference for this analogical process may be related not only to its efficiency and its capacity for conserving the designer's mental resources, but also to the way in which many designers have learned their craft. A great many experienced designers have had little formal training. When one learns on the job in a bottom-up, data-driven fashion, one learns techniques first. Concepts are slower to develop. When techniques or algorithms work, we tend to fix them in our memory. Because it can stifle creativity, problem solving by analogy needs to be used with care. Designers need to realize that the routine application of past habits can inhibit creative problem solving in situations that demand novel solutions.

THE ROLE OF SKETCHES

When a solver is not able to retrieve an algorithm for solving the problem from memory, one must be constructed on the basis of the information stored in memory. To help them, problem solvers sometimes use what Greeno (1973) called imaginal processes to represent the important relationships in a problem. A key element in the design process is the creation of a model of a finished work in advance of its embodiment (Archer, 1963-1). Designers often use mental imagery in developing solutions, the analog representations being manipulated for that purpose in STM. Sketches, diagrams, and drawings are ways of generating external representations that relate a problem directly to an individual's knowledge of the world. They help reveal inconsistencies in this information and also serve as a set of external memory structures.

SEEING THE LIGHT

Sometimes the problem can be solved only by a creative restructuring of the individual's problem representation. Trying to fit problem elements together in a new way is extremely difficult. Insight, seeing connections between seemingly different areas, appears to be mediated by declarative knowledge and is often triggered by a clue that the solver often never consciously notices.

Checklists of factors that designers have found by experience to be relevant to particular types of problems are effective clues (Archer, 1963-2; Malhotra, Thomas, Carroll, and Miller, 1980). There are, of course, times when an designer does not possess relevant declarative knowledge and providing direction does not trigger recognition of appropriate information. In these situations designers with different areas of expertise can profitably work together. Brainstorming represents a technique for extracting in as short a time as possible a great deal of the declarative and procedural knowledge embodied in the collective experience of such groups.

ANALYSIS VERSUS SYNTHESIS

We usually speak and write of analysis and design as if they were two separate processes widely separated in time and function. Our models of the IS design process certainly reflect a prevalent understanding that these are distinct processes. Studies of problem solving in complex, real-world situations, however, have shown that they are but two facets of the same process and that, for experts, they occur within seconds of one another. When a problem is understood, that is, when a problem is categorized as a particular type on the basis of the knowledge available to the solver, a solution model is instantaneously available. If this is true, and it appears to be, then analysis and design must be seen as one inextricably linked process rather than a set of divergent and convergent processes.

Conclusion

Information System analysis and design can be characterized as a problem-solving process. To solve a problem, one must understand the problem, devise and implement a plan, and finally evaluate and monitor the solution. Understanding means solution, but it requires knowledge. The understanding of IS design problems requires a great deal of information that is well organized and well suited to design tasks. Qualitative as well as quantitative differences between expert and novice designers derive primarily from differences in their knowledge bases. In the light of the extensive research in other areas as well of those few studies in IS, it is clear that many of the failures that IS designers have experienced have derived from problems in the development and management of their knowledge bases.

There are a great many problems that researchers in IS would like to solve, problems to which many people have devoted a great deal of time and effort. Because of the background laid by researchers in many disciplines, we can begin to explore these problems with a more realistic view of what it means to solve problems. It means that we have to be clear about what we do know. We must restructure our models of the design process to reflect reality more closely. In order to find a problem, solvers must have a model of the process; yet our basic model of systems design process would seem to be flawed. At the very least, this would reduce the gap perceived by many successful and experienced designers between practice and theory (Archer, 1963-1, 1963-2, 1964; Turner, 1983). It would improve our teaching; our students would be better prepared for the reality of systems analysis and design. More importantly such restructuring would help clarify our thinking and point to promising lines of fundamental research.

We need to discover what the experts in our field know. We need to learn more about the declarative and procedural knowledge important for successful IS design. A better understanding of the design process could lead to the development of knowledge-based design tools. A clearer understanding of problem solving could lead to the development of basic tools that could facilitate creative problem solving whenever such creative thinking was called for by either our students, our clients, or ourselves. Simon wrote that the proper study of mankind is the study of design, our common core of knowledge (Simon, 1981). Exploring the cognitive processes involved in solving IS design problems holds great promise for the future of our field and will extend to discoveries that we cannot now imagine.

REFERENCES

Anderson, J.R. "Acquisition of Cognitive Skill", Psychological Review, Volume 89, 1982, pp. 369-406.

Archer, L.B. "Systematic Method for Designers: Part 2, The Nature of Designing", Design, volume 174, 1963, pp. 70-73.

Archer, L.B. "Systematic Method for Designers: Part 4, Examining the Evidence", Design, Volume 179, 1963, pp. 68-72.

Archer, L.B. "Systematic Method for Designers: Part 5, the Creative Leap", Design, Volume 181, 1964, pp. 50-52.

Boehm, B. W. Software Engineering Economics. Englewood Cliffs, NJ: Prentice-Hall, 1981.

Boland, R.J. "The Process and Product of System Design", Management Science, Volume 24, 1978, pp. 887-898.

Bonar, J., Ehrlich, K., Soloway, E., and Rubin, E. "Collecting and Analyzing On-line Protocols from Novice Programmers", Behavior Research Methods and Instrumentation, Volume 14, 1981, pp. 203-209.

Bostrom, R.P., and Heinen, J.S. "MIS Problems and Failures: A Socio-technical Perspective: Part I, The Causes", MIS Quarterly, Volume 1, 1977, pp. 17-32.

Chase, W.G., and Simon, H.A. "Perception in Chess", Cognitive Psychology, Volume 4, 1973, pp. 55-81.

Chi, M.T.H., Feltovich, P.J., and Glaser, R. "Categorization and Representation of Physics Problems by Experts and Novices", Cognitive Science, Volume 5, 1981, pp. 121-152.

Chi, M.T.H., Glaser, R., and Rees, E. "Expertise in Problem Solving", In R.J. Sternberg (Ed.), Advances in the Psychology of Human Intelligence (Vol. 1, pp. 7-75). Hillsdale, NJ: Erlbaum, 1982.

DeGroot, A.D. "Perception and Memory Versus Thought: Some Old Ideas and Recent Findings", In B. Kleinmuntz (Ed.), Problem-solving: Research, Method, and Theory. New York: Wiley, 1966.

DeMarco, T. Controlling Software Products: Management, Measurement, and Estimation. New York: Yourdon Press, 1982.

DiSessa, A. "Phenomenology and the Evolution of Intuition", In D. Gentner and A.L. Stevens (Eds.), Mental Models (pp. 15-34). Hillsdale, NJ: Erlbaum, 1983.

Gagne, E.D. The Cognitive Psychology of School Learning. Boston: Little, Brown: 1985.

Ginzberg, M.J. (1981). "Early Diagnosis of MIS Implementation Failure: Promising Results and Unanswered Questions", Management Science, Volume 27, pp. 459-478.

Greeno, J.G. "The Structure of Memory and the Process of Solving Problems", In R.L. Solso (Ed.), Contemporary Issues in Cognitive Psychology: The Loyola Symposium (pp. 105-133). New York: Wiley, 1973.

Jeffries, R., Polson, P.G., Razran, L., and Atwood, M.E. "A Process Model for Missionaries, Cannibals and Other River-crossing Problems", Cognitive Psychology, Volume 9, 1977, pp. 412-440.

Jeffries, R., Turner, A.T., Polson, P.G., and Atwood, M.E. The Processes Involved in Designing Software. Englewood, CO: Science Applications, 1980.

Keen, P.G.W. "Information Systems and Organizational Change", Communications of the ACM, Volume 24, 1981, pp. 24-33.

Keen, P.G.W., and Gerson, E.M. "The Politics of Software Systems Design", Datamation, Volume 23, 1977, pp. 80-84.

Larkin, J.H. Understanding Problem Representations and Skill in Physics (Tech. Rep. Applied Cognitive Psychology No. 2). Pittsburgh: Carnegie-Mellon University, Department of Psychology, 1980.

Larkin, J.H. Mechanisms of Effective Problem Representation in Physics (CIP No. 434). Pittwsburgh: Carnegie-Mellon University, Department of Psychology, 1983.

Malhotra, A., Thomas, J.C., Carroll, J.M., and Miller, L.A. "Cognitive Processes in Design", International Journal of Man-Machine Studies, Volume 12, 1980, pp. 119-140.

Markus, M.L. "Power, Politics, and MIS Implementation", Communications of the ACM, Volume 26, 1983, pp.430-444.

McDermott, J., and Larkin, J..H. "Representing Textbook Physics Problems", Proceedings of the 2nd National Conference of the Canadian Society for Computational Studies of Intelligence. Toronto: University of Toronto Press, 1978.

Polya, G. How to Solve It: A New Aspect of Mathematical Method (2nd ed.). Garden City, NY: Doubleday, 1957.

Rumelhart, D.E., and Ortony, A. "The Representation of Knowledge in Memory", In R.C. Anderson, R.J. Spiro, and W.E. Montague (Eds.), Schooling and the Acquisition of Knowledge. Hillsdale, NJ: Erlbaum, 1977.

Simon, H.A. The Sciences of the Artificial (2nd ed.). Cambridge, MA: MIT Press, 1981.

Simon, D.P., and Simon, H.A. "Individual Differences in Solving Physics Problems", In R. Siegler (Ed.), Children's Thinking: What Develops? (pp. 325-348). Hillsdale, NJ: Erlbaum, 1978.

Soloway, E., Ehrlich, K., Bonar, J., and Greenspan, J. "What Do Novices Know About Programming?", In A. Badre and B. Shneiderman (Eds.), Directions in Human/Computer Interaction (pp. 27-53). Norwood, NJ: Ablex, 1982.

Turner, J.A. "Thoughts on the Process of System Design", Unpublished manuscript, New York University, 1983.

Vitalari, N.P. An Investigation of the Problem Solving Behavior of Systems Analysts. (Doctoral dissertation, University of Minnesota, 1981). Dissertation Abstracts International, Volume 42, 1981, pp. 2762-A.

Vitalari, N.P., and Dickson, G.W. "Problem Solving for Effective Systems Analysis: An Experimental Exploration", Communications of the ACM, Volume 26, 1983, pp. 948-956.

Weber, E.S. User and Analyst Mental Models as Predictors of Success in the Implementation of Management Information Systems. Unpublished doctoral dissertation, University of Texas at Austin, Austin, 1984.

Weinstein, C.E., and Mayer, R.E., "The Teaching of Learning Strategies", In M.C. Wittrock (Ed.), Handbook of Research on Teaching. New York: MacMillan, 1985.

Zave, P. "The Operational Versus the Conventional Approach to Software Development", Communications of the ACM, Volume 27, 1984, pp. 104-118.