THE PROCESS OF SYSTEMS DESIGN: SOME PROBLEMS, PRINCIPLES AND PERSPECTIVES

Jon A. Turner

December 1985

Center for Research on Information Systems
Computer Applications and Information Systems Area
Graduate School of Business Administration
New York University

Working Paper Series

CRIS #110

GBA #85-101

Submitted to **Management Science.**

THE PROCESS OF SYSTEMS DESIGN:

SOME PROBLEMS, PRINCIPLES AND PERSPECTIVES

Abstract

This paper explores issues that are central to designing, and particularly to the design of information systems. It portrays the context of design, the considerations that go into designing - how these are in conflict, and how they are ultimately resolved - and the role of creativity in this process. A set of design principles is presented and discussed.

Acknowledgements: Conversations with Professors Ron A. Weber and Richard J. Welke have been helpful in clarifying a number of the notions presented. I am indebted to my wife, Margaret Helfand, a practicing architect, for many ideas and illustrations. The material on design elements was developed for an advanced course in Systems Design at NYU. Most of all, I have been motivated by my students who ask good questions.

1.0 INTRODUCTION

While there is general agreement on the kinds of information that should be gathered for the purpose of designing a new computer application system ([1], [2]) and how this material should be organized and presented ([3], [4]), there is relatively little written about the process of design itself. What are the component parts of design and what role do they play in the process? What is the vocabulary of design? Precisely what is meant by the term 'to design'? What thought processes are involved in designing? How does one recognize a good design from a poor one? And finally, how should design be taught?

At one level the answers to these questions are obvious. A good system design is one that meets the 'requirements' of the situation. But, on closer inspection this answer raises a number of unresolved issues. How are the requirements to be determined? Davis [5] has observed that there are four general approaches for determining information system requirements.

The first involves asking persons involved in the utilizing system for a statement of requirements. This approach presumes that users have a satisfactory way to structure their problem, that various biases can be removed, and the absence of what Ringle and Bruce [6] refer to as 'communication failure'. The second approach is to derive the requirements from an existing system, either a system that will be replaced by a new system, a system description, or a proprietary system. A third strategy is discovering the requirements from experiments with an evolving system, for example, by prototyping. The fourth strategy is to synthesize them from an analysis of the activities of the object system. Methods such as BIAIT [6], strategy set transformation [7], critical success factor analysis [8], process analysis [9], decision analysis, socio-technical analysis [10], participative system design [11], and input-process-output analysis [12] are typical of approaches being used.

Assume that several of these methods were applied to the same problem, will they, then, produce the same set of requirements? Are the requirements for a system dependent on the analysis method used as well as the system being designed? Given the systems principle of 'equifinality' [13], isn't the best strategy to use all of these methods? On what basis does one then decide conflicts among requirements produced by different methods, or even by the same method? Is it even reasonable to expect that a complete set of requirements will ever be established?

Then, how are candidate designs (to meet these requirements) generated? What is it about requirements that leads to the formulation of design solutions? Once defined, on what basis is it decided that one design best meets the stated requirements? Questions like these seem to be at the crux of an understanding of the process of systems design. Yet, there is little research and few papers that have as their goal an understanding of the process itself. Clearly then, these questions are neither being asked nor answered.

There is agreement on the importance of design in information system implementation. Errors are most likely to be introduced into an information system during the requirements definition and preliminary design stages [15], and errors that occur during these early stages are likely to be most costly [16]. Ginzberg [14] indicates they are the most important stages of the implementation process. With Artificial Intelligence (AI) moving into commercial applications it becomes even more important to understand the design process in greater detail. Knowledge engineering, the process of extracting knowledge from an expert and codifying it, is now considered to be the most difficult part of expert systems design [31].

The literature is laced with pleas to design better systems, for example, to make them more 'user friendly'. It is implied in this literature that design is synonymous with the application of a methodology, such as Structured Systems Analysis [3] or BIAIT [6]. The presumption is that by mechanically executing the proper methodology a good design will emerge. My experience suggests, however, that although methodology plays a role, it is not sufficient. Much of the process appears to be a 'black art' practiced by a small group of highly skilled individuals.

Something appears to be missing in the way information system design is characterized in the literature. Absent is a model of design itself, one that could serve as a basis for an understanding of the process. This is a necessary precursor to our saying how design should be done and for our ability to teach it. Yet, designing information systems is a just a special case of designing artificial objects, which is more developed in architecture and engineering. Consequently, much can be learned from an understanding of how design is perceived in these fields.

This paper reviews some of the writings on design theories from engineering and architecture along with recent research findings in information systems that provide insight into underlying cognitive processes. Major elements involved in design are identified with a goal of presenting a unified approach to the design of information systems. Finally, issues involved in the teaching of design are discussed.

2.0 THE PRACTICE OF INDUSTRIAL DESIGN

Much of what follows in this section has been adapted from a series of seven articles by L. Bruce Archer that appeared in Design during 1963 and 1964. Although the focus is mostly on industrial design, the notions appear to apply equally well to the design of any artifact, including information systems (IS).

The art of design has been defined as "selecting the right material and shaping it to meet the needs of function and aesthetics within the limitations of the available means of production" [14]. The problem is clear in this formulation. The finished product has some function to perform, which must be understood by the designer and eventually represented in the product. Along with these functional needs are aesthetic considerations, involving subjective judgments, that are shaped by the values of the designer and the client(s). These two categories of factors - functions and aesthetics - are fundamentally different in nature and are likely to be in conflict. The process is further constrained by the production method to be employed - not all designs that resolve functional and aesthetic considerations will result in a high quality product after production.

Thus the the design of a product implies also the design of the method of production.

These are the three dimensions upon which the art of design is practiced: functions, aesthetics, and methods of production. It is considered an art because the rules for moving from one state to another are not well understood for large portions of the process. In IS design the difficulty is in visualizing the product and making value preferences explicit.

When user needs were simple, materials few, and manufacturing methods relatively crude, the designer was able to adopt rules of thumb in a manner close to sculpture. Today, the job of the designer has become more difficult - there is a galaxy of materials to chose from, with many of them having no true shape, color, or texture of their own. At the same time, the cost and complexity of tooling means that the designer cannot afford to be wrong. Similarly, in IS, the number of options for implementation have multiplied greatly over the past 10 years.

Aesthetics, in practice, consists of doing things which are calculated to please the senses, and in appraising things according to their appeal to the senses. A measure of what is pleasing or displeasing to most people or to different classes of people can be determined using market (survey) research techniques. Aesthetics may be said to fall into two broad divisions - descriptive aesthetics, which deals with empirical facts about perceivable qualities and the statistics of preferences; and ethical aesthetics, which is concerned with good taste and bad taste, or appropriateness.

Perhaps several other definitions are in order. Pure science investigates the nature of phenomena, but passes no judgment about them. Practical science, on the other hand, seeks to pass judgments, to help people with a problem choose what to do. Ethics is the science of distinguishing between right and wrong, good and bad, appropriate and inappropriate. Technology, too, is concerned with helping people choose what to do. However, technology is concerned with problems about <u>means</u> while ethics is concerned with problems about <u>ends</u>. Thus ethics is an essential element not only in the practice of aesthetics, but also in the practice of any profession which is involved with the exercise of a value judgment.

It is quite possible to work out by scientific methods who likes what, where, and in what circumstances. Predictions can even be made about where the trends seem to be going. But there are no immutable truths in aesthetics. Its essence is <u>choice</u> with the aim of appropriateness, and the criteria are the center of gravity and the periphery of all the choices made so far. Individuals have their own standards and a consciousness of other peoples standards; each makes their own choice. A designer's special problem is that they must foresee the probable future choice of <u>other</u> people, as well as their own. In the majority of cases it is far quicker and more appropriate to handle the whole aesthetic side of design by intuition, provided there is an adequate body of prior experience to base it upon. But it is possible to reduce the area of unknown and to define by systematic examination those elements in the problem which should properly be judged intuitively and those that lend themselves to objective evaluation.

This view is a departure from the traditional picture of IS design in that it acknowledges the role of individual values and aesthetics in the process. It suggests, also, that it is permissible to resolve certain issues on the basis of intuition and how experience (and learning) colors intuition.

2.1 Design Principles

A key element in the act of designing is the formulation of a <u>prescription</u> or <u>model</u> for a finished work in advance of its embodiment. When a sculptor produces a sketch of a proposed work, only then can they be said to be designing it. The sketch represents a formulation of the idea before its embodiment in the final material.

Another aspect of design deals with the role of <u>creativity</u>. Arriving at a solution by strict calculation is not regarded as designing because the solution is seen as arising automatically and inevitably from the interaction of the method of solution and the data. In this regard the process of calculating is considered to be non-creative. However, the selection of a solution method or the representation of a problem in a form that permits it to be solved by calculation may be considered design, if it does not follow directly from the statement of the problem. It is characteristic of creative solutions that they are seen to be apt solutions <u>after</u> completion and <u>not</u> before. Consequently, some sense of originality is also an essential part of designing.

So it can be said that the process of designing involves a prescription or model, the intention of embodiment in some physical form, and the presence of a creative step. This description implies a purposeful seeking after solutions rather than idle exploration. It also implies that certain limitations exist which constrain the acceptable solutions and that recourse to random action will not be sufficient. It also suggests that some creativity is involved in the forming of solutions.

There can be no solution without a problem; no problem without constraints that limit the acceptable solutions; no constraints without a pressure or need. Thus, design begins with a need. Either the need is met automatically, and there is no problem, or the need is not met because of certain obstacles or gaps (which may be in the articulation or understanding of the need). The finding of means to overcome these obstacles or gaps constitutes the problem. If solving a problem involves the formulation of a prescription or model for subsequent embodiment as a material object and requires a creative step, then it is a design problem. The skills necessary for its solution depend upon the nature of the constraints. For example, in industrial design the major constraints involve the importance of visual elements in the design of the end product and the fact that the end product is made by industrial methods. In designing information systems, the major constraints are the functionality of the system, and the environment, both organizational and operational in which the system will reside.

The art of design is the art of reconciliation. In industrial design, manufacturing (building), use, and marketing (selling, including product support) produce a complex of competing factors that must be reconciled into an end product. The considerations involved in this resolution can be reduced to three categories involving three human factors (motivation, ergonomics, aesthetics), three technical factors (function, mechanism, structure), and three business factors (production, economics, presentation) giving nine design factors altogether. Some of these factors, such as economics relate to matters of fact susceptible to measurement and optimization. Others, such as aesthetics, relate to matters of value which can only be assessed subjectively. This variation in the quality of factors is characteristic of design problems.

It is the nature of design problems that they often begin with an analytical phase involving objective observation and inductive reasoning. In contrast, the creative phase at the heart of the process requires subjective judgment and deductive reasoning. Once the crucial decisions have been made, the design process proceeds with detailing of the design, for example, producing working drawings and specifications in architecture or program specifications in IS, both objective and descriptive. The design process is thus a creative sandwich. The bread of objective analysis may be thick or thin, but the creative act is always in the middle!

One of the frequently made mistakes in IS design is to presume that the objective portion, involving, for example, documenting an existing system, constitutes all of the design activity. This view is incomplete because it does not recognize the creative process that takes place in the middle. However, this creative process could not be accomplished without the preceding objective process of preliminary information gathering aimed at identifying (incompletely) the needs and constraints of the situation.

2.2 Requirements For Design

Industrial designers and architects speak of identifying _objectives_ and _constraints_ at the beginning of the design effort, usually as part of developing the 'program' [14]. This involves asking the right questions in order to recognize the needs and pressures for change. Constraints are identified paying particular attention to unknowns. The essential criteria by which a 'good' solution can be distinguished from a 'not good' one are also identified at this point. Note that what is produced is a list of the attributes which the final solution is required to have. This result, however, is a statement of the problem, _not_ the answer.

2.3 The Creative Leap

There still remains the crux of the design problem - the _creative leap_ from a pondering of the question to finding a solution. Industrial designers and architects appear to establish a first approximation to a solution to the problem based on _prior experience_ [14]. This essentially means finding the connections between the goals, in terms of the attributes of a good solution, and the facts of the situation as mediated by the designer's knowledge and experience. Constraints serve to bound the problem and may provide useful clues as to where solutions may be found.

Designers appear to search their minds for a solution to their problems by examining all kinds of analogies [14]. They look at other people's end results, checking whether something on those lines would answer their own problem. Only after all sorts of solutions have been reviewed, including phenomena and artifacts in the most unlikely fields, do designers return to the question and examine other questions of a comparable kind handled by themselves and others.

If this still yields no result the designer tries to reformulate the problem. Only as a last resort do they attempt deductive reasoning, proceeding from analysis of the data to the necessary conclusion, instead of the other way around.

One might say in IS (or Artificial Intelligence) terms that the designer first attempts to identify an acceptable solution by investigating analogies from other people's end results or solutions. Acceptability is determined by a backwards, depth-first search from potential solutions to parameters of the problem. Knowledge and experience provide the search paths and constraints serve to bound the search. If no solution is found the designer examines similar problems for a solution. If one is still not found, the designer attempts to reformulate the problem in a manner to produce a solution. If one is still not found, the designer then attempts a forwards, breath-first expansion of the problem to see if it leads to a solution.

2.4 Role Of Experience

The effects of prior experience, expectation and purpose on an individual's capacity to perceive evidence and judge hypotheses have been well established [14]. People have developed agile machinery for filtering out of the hail of signals with which their senses are bombarded from those few which are significant at any given time. The filtering apparatus of the nervous system suppresses what it takes to be accidental, spurious or irrelevant sensations before they reach consciousness. It is here that prior experience and learning play an important role. Research has shown that what people think they see is based upon a comparison with complex collections of previous expectations, fulfilled or disappointed.

The conclusion is that people cannot believe their own eyes, and that the most painstaking care has to be taken, both in data analysis and in hypothesis seeking, to counterbalance the effects of the 'perceptual' filter. Observers contributes from their own experience, by either addition or subtraction, to their perception of the phenomenon before them. Thus, for example, in IS requirements analysis, an analyst must be careful in filling-in requirements for users based on their own experience.

Perception is a two-way business. One is confronted with the need for rich, wide, and fruitful experiences among designers as well as the capacity for flexibility and fantasy in thought in order to recognize those aspects of a design problem that are important. Along with experience, however, comes the danger of biasing the way real data is interpreted.

2.5 Associative Nature Of Design Thought

It appears that mental processes work in loops, taking one jump ahead and then two steps back with great rapidity [14, 17]. One works from design ideas to anticipated consequences and then backwards to test the appropriateness of the design idea as an answer to the functional problem, and from a practical point of view.

2.6 Design Idea

There is a real distinction between a design idea and any one embodiment of it. The design idea is an invention, an abstract concept, while the finished design is one of many possible embodiments of the design idea. For example, in a patent application, the inventor is asked to describe separately the invention and a material embodiment of it. The description of the invention is interpreted literally and is deemed to cover all of the variations that the inventor wishes.

On the other hand, the description of the material embodiment of the invention is interpreted freely and is regarded merely as an exemplar. Hence, there is a logical difference, both in intention and method, between the kind of activities and thought processes that go on in the synthesis phase and that which goes on in the development phase. Whereas abstract analysis is needed to show that a given design idea is the best recipe for solving a problem, the proof is in evaluating the final product.

Detailed development is intended to fill gaps and solve problems in making selected design ideas work. This is another way in which experience comes into play during the act of designing, through the rejection of certain ideas as impractical. Because of the different thought processes involved, it is usually necessary to suppress any new thoughts on basic design ideas once the development stage has been entered.

2.7 Design Development

Engineers may be weak in searching for original design ideas, but they are strong in the technique of developing detail. Drawings, three dimensional models, stress and wind tunnel models, bread-boards and prototypes are analogies used in design development. The more abstract the form, the more basic and flexible can the appraisal be. The more realistic the form, the more directly can overall or ultimate effects be judged. Abstract models are used for development and test of basic ideas while more realistic models are used for development and test of the design embodiment.

To provide sufficient room for maneuver, the product designer strives to keep a minimum interdependence between design elements and to maximize the contribution of any one element to the solution of the problem. The best design is the crudest that will just do the job. It is the one which gives the highest quality whole with the lowest quality elements.

The information available to the designer is so patchy or unreliable on so many matters that it would be impossible to arrive at solutions without making some assumptions or judgments which go beyond the evidence. Never the less, almost any proposed design solution constitutes an hypothesis based upon imperfect evidence, and it must be subject either to the test of the marketplace or some indirect analysis.

2.8 The Art Of Communication

In all cases where a division of labor exists between design and production, information about exactly what is intended must be conveyed by the designer to the producer. The art of communication is thus an essential element in design practice. Since no designer can have the knowledge to specify every aspect of the design, there is much to be said for leaving as much room for interpretation as possible. The key to the practice of the art appears to be communicating the design idea in a <u>general</u> way along with a <u>specific</u> description of the particular embodiment. This is accomplished by preparing a complete set of design details showing the properties <u>required</u> of each component rather than specifying how these properties are to be provided. Thus, it would be better to define the range of surface roughness which would be acceptable rather than to call for an amount of grinding or polishing.

2.9 Summary

Design is basically an ethical aesthetic judgment involving the values of both designer and client(s). While some portions of the process are concerned with objective evaluation and information gathering, a major part is intuitive. Key is the identification of obstacles or gaps in knowledge, the recognition of constraints and an understanding of their implications, and the identification of the criteria by which a good design can be distinguished from a poor one. The central element is the design idea. Experience plays a critical role in separating the important from the unimportant, in guiding how the designer's effort is allocated, and in determining the fit between solutions and problems.

The whole key to the systematic analysis of design problems is in the correct evaluation of priorities and criteria, and these are human value judgments.

This suggests that we should be attempting to identify the aesthetics of IS design rather than trying to transform all aspects of the process into methodology. Fruitful areas for investigation may be the cognitive processes used by expert designers and how designers represent problems in their minds.

3.0 PSYCHOLOGICAL RESEARCH INTO IS DESIGN PROCESSES

There are relatively few reports of studies that investigate the cognitive processes involved in design. Malhotra et al. [17] observe that existing theories of design are characterized as being 1) application specific models or procedures, 2) verbal models of the psychological process involved in creativity, 3) general discussions followed by a formalism of some portion of the design process, or 4) descriptions of design stages. These theories fall short of providing an adequate theory of design in that they do not differentiate design from other kinds of problem solving activity, they do not provide a basis for verification or empirical testing, and they do not provide a unified framework or model for viewing the whole design process.

3.1 The Design Situation

Malhotra et al. represents the design situation as:

> A problem state is said to exist when a human or other goal oriented system has a goal but no immediate procedure that will guarantee attainment of the goal. ... Problem-solving occurs in moving from a problem state to a non-problem state. In problem-solving, then, a person begins in an initial state, uses transformations that move him from one state to another, and ends in a final state. Any of the states may be well-defined or ill-defined [17, p. 120].

In the case of design problems, the designer need not start from a specific initial state. Although constraints may restrict what is used or considered, the transformations are not usually limited.

Real world design situations are characterized as situations where:

> .. the goals are typically fuzzy and poorly articulated and cannot be mapped directly into properties of the design. Thus, the exact configuration of the final state is not prescribed. A part of the design process consists of formalizing and refining the design goals into functional requirements that can be matched by properties of the design. Even so, it is usually difficult to tell how well a design meets a particular functional requirement. In addition, the functional requirements often cover different dimensions and the trade-offs between them are rarely well specified [17, p. 120].

The properties of a design arise from a combination of design elements, indivisible units with certain properties, and the design organization, the way the design elements interact. It is interesting to observe how closely this formulation of the design problem parallels that of industrial designers represented in section 2.

3.2 Client-Designer Dialogues

In order to gain further insight into the design process, Malhotra et al. studied problem-oriented dialogues between people attempting to solve real world problems. Client-Designer (C-D) dialogues can be considered to consist of the translation of design goals into a set of functional requirements that the design must meet and the generation of a design to meet those requirements. In reality, C-D dialogues are more complex often involving implied requirements, examination of partially proposed designs to test violation of some unstated goal, substitution of a design solution with a better one, and the combination of design components into a solution. Note how much of this process appears to be implicit and unstated.

The researchers observed that the Client-Designer communication was composed of cycles, each one broken into a small number of of mutually exclusive states. The states were defined by the major activities pursued in them and consisted of: 1) goal statement, 2) goal elaboration, 3) solution outline, 4) solution elaboration, 5) solution explication, and 6) agreement on solution.

In examining the dialogues, Malhotra et al. noted that they consisted of a series of design cycles each composed of a <u>regular succession</u> of states 1 through 5. Closer examination revealed that a diversity of content underlay this apparent regularity of structure. For example, a cycle may start with a fresh set of requirements, or, although solution suggestions and discussions always follow discussion of requirements, the solution that is outlined need not apply to the requirements that precede it. New requirements are often uncovered in the process of examining solutions and these may start their own design cycles. This behavior conforms to the associative nature of design thought noted earlier.

Generation of design solutions seems to consist of attempting to find design elements to meet functional requirements and then tying these elements together into a coherent design. As Malhotra observes, this corresponds roughly to bottom-up design. Although this was not the only <u>design strategy</u> exhibited, it was the predominant one and it seemed to be encouraged by the fragmentary presentation and elaboration of requirements. This pattern suggests the backward, depth-first search proposed in section 2. When the design problem was complex, designers often left, returning at a later time with potential design solutions. This gave the designer sufficient time to internalize the requirements and to use different strategies in generating solutions.

In pondering why the designer works with fragmentary information rather than waiting for a complete set of requirements to emerge, Malhotra et al. suggest that a dialogue requires the contribution of both parties - somewhat like a dance. Since the designer's domain is solutions, he presents them, while the client describes requirements. The premature introduction of solutions appears tied to helping the client articulate and elaborate his goals.

3.3 Design Studies

Malhotra et al. measured the designs produced by subjects on their <u>originality</u> and <u>practicality</u>, two factors that are considered to constitute creativity. Subjects were given a design problem involving the location of a restaurant in an old church. A procedure for measuring the originality (O) and practicality (P) of each design was developed permitting the relationship between them to be investigated.

The researchers found a significant negative correlation between O and P, suggesting that a trade-off existed. The distributions of the two parameters were also interesting; P was fairly normally distributed while O was skewed with little variation. Evidently, only a few subjects produced highly original designs. There was no correlation between subjects that claimed to have designed top down, to have planned their approach, or to have tackled the more difficult problems first, and either O or P. What was predictive of O and P were the subject's expressed goals. Those subjects that claimed to be more interested in having a design that was novel, imaginative, and original scored lower on P. Those that strived for a design that was workable scored lower on O.

One group of subjects were given a list of words that had been found useful in similar design problems. These subjects scored significantly higher on P but not on O. The researchers reasoned that, in design, subjects often possess relevant knowledge which is not spontaneously accessed. Much of this information will be recognized as being relevant after it is cued for recall.

In another study, Malhotra et al. asked four subjects to write a set of functional requirements for a query system including defining functions and specifying the syntax of queries. An analysis of the designs showed wide variation in approaches taken and in products. At one extreme, a minimum set of functions was provided along with a simple syntax; the other extreme was a complex set of functions and syntax. The two other subjects took completely different approaches. One said that a standard query language could be used; the other provided a menu of queries to select from.

The researchers concluded that the sub-goals or solution strategies generated from the higher level goals seemed to vary widely and there did not seem to be an orderly procedure for generating sub-goals. The selection of sub-goals appeared idiosyncratic and to depend strongly on past experience.

In a follow-up study, subjects were provided with specific functional requirements for a query system and asked to design the data structures and algorithms [17]. Comparative analysis of the designs revealed that their content were all different - in module content, data structures, and algorithms. For example, two subjects did not specify a data structure at all. Each of the other six used a different structure: a vector, a PL/I data structure, an attribute-value structure, two kinds of tables, and a list with switches. Furthermore, when interviewed, the subjects indicated they did not have strong reasons for selecting the structures they did.

In addition, along with a great diversity in algorithms, the solutions contained errors, inconsistencies, and unwarranted assumptions. They also varied greatly in their level of detail. The researchers concluded that, unlike engineering designs, it was difficult to tell whether a software design was complete, consistent, or even met functional requirements.

Another difference between engineering and software is that the components of engineering design are sub-assemblies and other structural elements, while those in software are constructed anew from the basic facilities of the programming language. Consequently, the software designer can create a bewildering array of intermediate pieces for constructing a program. This diversity of components means that choices are made on an arbitrary bases, often on the basis of prior experience or familiarity.

These findings are supported by a study of the design process performed by the author. Students in a Systems Analysis and Design course (n=21) were given a written description of a commodity brokerage operation (matching of buyers and sellers, issuing a contract, processing letters of credit, shipping notification, issuing insurance, duty clearance, and commission calculation) and asked to produce data flow diagrams [3] representing the system as they understood it existed. (Students had prior experience using this technique in class.) The resulting diagrams were then analyzed to determine their similarities and differences.

Specifically, the following were investigated:

1. The boundary of the system as portrayed by the context diagram. That is, what activities are included in the system, and what are external. For example, does the system maintain its own accounts receivable file or does it only prepare transactions that are posted by an external system.

2. The data flow names and data element contents of the flows.

3. The process functions as represented by lower level diagrams.

The diagrams showed many more differences than similarities. Students varied widely in what they included in the system. Like the results of Malhotra et al.'s experiments [17], these students made a number of assumptions, many in direct conflict with the written description of the problem. Students varied widely in the names and contents of their data flows. Although there appeared to be four somewhat different approaches taken, that is, the diagrams could be grouped into one of four categories, there was no underlying single category that all of the diagrams could be reduced to. Furthermore, based on objective criteria, it was difficult to tell which of the four categories was superior; they all had strengths and weaknesses. However, in class discussions, it was often possible to combine aspects of different diagrams to arrive at one that most people agreed was better than the others.

One of the most striking findings was that students used different strategies in decomposing the system into parts. That is, the data flows (and corresponding processes) as represented in the first level data flow diagrams were considerably different. Four strategies were evident. The first, and most common, was a functional strategy, grouping activities around major functions being performed, for example, in generating a contract. There was, however, considerable variation in the functions that were selected as the basis of decomposition and how they were interconnected.

The second strategy was process oriented. Students visualized the system as having certain processing in common, and used these as organizing themes. For example, having all file update performed by one process. The third strategy was similar to the first in that it was functionally based, except that the functions had a strong time orientation and were highly operational. The fourth strategy was some combination of the prior three.

When questioned, students could explain the logic of their approach to decomposition quite clearly. They were, however, unsuccessful in persuading their colleagues that their own approach was preferable. It was hard to escape the conclusion that how students thought about the problem influenced their choice of a decomposition strategy. How they thought about the problem was largely a function of their background and experience. This finding is consistent with Malhotra et al. [17] and the representation of industrial design in section 2.

In summary, students differed widely in the entities they represented in their data flow diagrams, in what they selected to include in their systems, in the contents of their data flows, in the names chosen for data flows and processes, and in their approaches to problem decomposition. Rather than one solution to the problem, there were many, and it was difficult to select objectively among them. Any model of the design process must permit strongly individualistic approaches to design.

3.4 Design Process Model

Malhotra et al. propose a three part model of the design process: goal elaboration consisting of discovery, elaboration and statement of the goals of design, design generation involving a selection of design elements and organization such that their combined properties meet the functional requirements, and design evaluation concerned with determining how well the properties of a design meet stated or unstated goals. While all three parts involve subjective thought and judgment, the latter two appear to be mostly subjective. The researchers speculate that the quality of the design depends on strategies employed for problem solving, and that a good representation of the design reduces complexity by highlighting only the important features on which the design is to be based. This suggests that one of the differences between expert and novice designers may be the way they represent problems in their minds.

Jefferies et al. [18] observe that design tasks are too complex to be solved directly. Consequently, an important aspect of designing is decomposing a problem into more manageable parts. There are two prevailing views as to the basis for this decomposition. They differ in the nature of the problem reduction operators that apply and in the evaluation functions used for determining the adequacy of alternate solutions. With data structure oriented approaches [19], a designer specifies input and output data structures and then performs the decomposition by deriving the mapping between them. Because these methods involve the derivation of a single 'correct' decomposition there is no need for evaluation criteria. Data flow approaches [3] are guidelines for identifying trial decompositions of a problem. These methods are more subjective allowing a designer to exercise some judgment in what is included. Various heuristics are used for evaluating potential solutions.

Most software design methodologies require that the design proceed through several iterations, each being a representation of the problem at a more detailed level. This mode of decomposition, in general, leads to a top-down, breadth-first expansion of the design. Critics have pointed out that it is sometimes necessary to understand certain lower-level functions in order to identify some high level constraints [20], a process similar to what Malhotra found in the C-D dialogues.

Goal oriented specifications can be thought of as defining the properties that a solution must have. A review of automatic programming research indicates that there are several components to the task of software design [18]. The first involves the translation of the initial goal oriented specifications into a high-level functional decomposition of the original problem. Key here is the basis upon which the decomposition is performed. Second, this incomplete, abstract description of the problem must be refined into a set of formal specifications that precisely define functions performed, data and control structures. Then, specific data structures and algorithms that satisfy functional and efficiency criteria are selected.

Observations of expert software designers suggest that they possess a knowledge of the overall structure of a good design and the process of generating one. In a study of novice and expert designers, Jefferies et al. [18] found that novices' solutions were neither as correct, or complete, as experts, although they applied the same general problem solving techniques. Novices were also unable to apply the more efficient problem solving techniques used by the experts. The researchers concluded that skilled designers have knowledge of the global structure of the design task and its guiding control processes independent of a specific situation. This abstract knowledge about design and design processes develops through experience permitting efficient management of the designers resources in task performance.

3.5 Discussion

The characterization of design presented above differs from the life cycle model described in texts and used in many practical situations [24], which portrays the system implementation process as consisting of activities, grouped into phases, which are executed in time sequence. While the groupings are convenient from the standpoint of discussing the general activities in each phase, the distinction between phases is not nearly as clear in practice as it is implied in the model. Also, the life cycle model is static, it does not well represent the dynamics of design - the variables of interest and their interaction.

The life cycle model suggests that design closure can be reached by linear iteration of the process and that closure eventually results (refer to Table 1). If the process is followed long enough and enough iterations are performed, a good design will emerge. The implication is that one correct design solution exists and that it follows directly from the application of the methodology. This suggests that the process is rational, even subject to optimization, and that process and outcome are independent. Aesthetics plays no role in this model and all trade-offs can be reduced to values on a single scale. Experience does not come into play except in familiarity with executing the methodology.

Factor	Life Cycle Model	Creative Sandwich Model
Thought Process	linear, iterative	associative
Closure	reached	not reached
Search Strategy	forward, breath-first	backward, depth-first
Evaluation Criteria	not represented	developed with problem expansion
Aesthetics	plays no role	values guide selection of potential solutions
Conflict	not represented	resolved by application of values
Experience	plays no role	guides evaluation of potential solutions
No. Solutions	one implied	many
Completeness	solution implied complete	incomplete
Independence	product implied independent of process	product and process interdependent

Table 1

Comparison of Different Design Models

The creative sandwich model (and the C/D dialogue sub-model) suggests the importance of associative thought and heuristics in identifying and evaluating both solutions and requirements. It is a process based on discovery and learning, intimately related to a person's prior experience, involving a creative leap in order to understand a problem at a fundamental level and to construct imaginative solutions. Rather than one solution, there are many possible. Considerable skill and judgment is needed to resolve conflicts among potential solutions - they are not simple uni-dimensional trade-offs. The values of the key actors play an important role in how these trade-offs are made. It is the interaction of the properties of the solution that determines its quality.

Support for the creative sandwich model of design is provided in Boland's study of systems analysts [21]. Analysts using an approach based on mutual learning (analysts and users teaching each other about their specialties) performed better than analysts using the traditional approach to design (analyst in control playing the lead role). Boland found that designs produced by the learning analysts tended to be of higher quality and to use different control strategies.

As Boland suggests, different processes of interaction may help define different problems. Learning combined with shared control may produce an environment that better matches the way design actually takes place for complex systems where the solution is not obvious. It is an environment that builds mutual confidence and supports discovery. It encourages the associative thought process that characterize complex design situations. This may partially explain the success of prototype strategies for system development that create a similar situation for promoting discovery and discourse. In this case both developer and client learn about the system as it evolves.

The C/D Dialogue model of design proposed by Malhotra et al., with the designer presenting hypothetical solutions to problems and tracing through their implications as a way of encouraging a client to reveal additional requirements, appears to follow the experience of many designers. Central to this process is the importance of a common language between client and designer and the role of experience in providing a context in which to interpret what the other party means, and in generating plausible solutions and implications.

Many designers seem to start a line of thought, developing it as they go along, including the tracing through of implications. In other words, designers may not know in advance where their thought process will lead them. This is one reason why a checklist of factors to be considered in design improves performance. The list may act as a reminder to consider certain factors while not constraining the designer's thought process. Without the cue the designer might not consider these factors because they were not explicitly on his design path.

Prior research [17, 18] suggests that while designers follow logical segments, they jump around in selecting what segment to focus on next. This characterization of thought involved in design as an associative process underlines the importance of experience in creating a rich context and in subconsciously guiding the evaluation process. Aesthetics forms the basis upon which many of the conflicts among alternate solutions are resolved. This process is subjective, resting heavily on values and preferences formed at earlier times. It does not appear to be particularly critical where the process is started. As one designer expressed the process, 'it is only important to get it in the right quadrant. That, in itself, removes three quarters of the options.'

Designers seem often to back into their solutions, removing as many options as possible using elimination as a strategy for simplification. This is consistent with the backwards searching nature of the process. Solutions that do not connect to the particulars of the situation can then be rejected.

Designers appear to develop criteria for evaluation at the same time that they expand the needs for a system. These criteria are then used in the evaluation of alternate solutions along with their aesthetic rules.

The design is the sum of all of the prior design decisions, the locus of these decisions. Some unifying concept is needed to permit accepting or rejecting new decisions so that the whole remains consistent. The role of this unifying concept has been overlooked in the writings about design, yet it remains an important notion (see section 4.1).

Heuristics play a role in simplifying the search and in identifying what design segment to pursue next. One area for further research is the identification of heuristics used by expert designers. Candidates include, analogy, assumption in the absence of data, hypothesis testing, verification by comparing data from multiple sources, plausibility, symmetry, and consistency testing. Another line of research is to understand what separates good designers from poor ones. Finally, additional protocol studies are needed to provide insight into how designers represent problems and the knowledge structures (domains) they bring to bear on a problem.

Weber [35] has suggested a useful way to characterize this situation. The 'task environment' is a problem as it appears in reality. Persons represent problems in their mind (in order to solve them) as 'problem space'. It is not clear whether problem space contains only a representation of the problem or also all of the possible solutions to the problem. 'Operators' are functions that when applied to a representation of a problem transform it from state s to state s'. 'Search strategies' are procedures used to search problem space for operators that apply to a problem or for desired transformations of a problem. 'Memory' which may be short or long term, contains sensory data and knowledge, the later consisting of facts, relationships among facts, operators, and control structures or 'schemas'. Schemas are a special kind of meta-knowledge that deals with how to apply the knowledge contained in memory, for example the configurations for applying particular operators. Schemas serve to link various features of the task environment to specific operators and to provide default values for missing variables. Thus, they are central to the interpretation of events and in problem solution. It is not clear what is static and what is dynamic in this formulation. One notion is that experts develop (learn) better schemata for dealing with problems in their domain of expertise, or in industrial design terms, their aesthetics are more highly developed.

4.0 DESIGN ELEMENTS

Prior research has shown the usefulness of a checklist of design elements as a cue to recalling general knowledge about the process [17]. It is my belief that experienced IS designers consider implicitly (that is, have developed refined schemas for) the following nine elements of design. They are presented here to make them explicit and in hopes that, as such, they will serve as a new, somewhat different, vocabulary for design.

No time sequencing is implied. That is, these issues are not necessarily resolved in the order in which they are listed. Nor are they likely to be the way people think about design. The cognitive processes involved in design seems to be associative, rather than sequential, with each practitioner following their own pathway. The set of topics presented below is a checklist of issues that experience has shown to be important factors that must be resolved when designing IS.

4.1 System Concept

In order to resolve conflicts during design and to serve as a guide in making consistent decisions, a system concept is needed. The concept is the rationale or unifying theme of the system. Typical concepts might be 'minimal' or 'simple', 'privacy', 'elaborate', etc. depending on the setting and goal of the system. Concepts should be expressed as a word, or at most, a sentence. An elaboration of what the system should do is not the system concept. The concept is a distillation of the system, its essence. It is analogous to the design idea (section 2.6) in industrial design.

Several examples may clarify what is meant by a system concept. Suppose, as an architect, you have been retained to design an ocean front house on a relatively narrow lot (200' wide by 450' deep) with other houses to be built on adjourning lots. There are several concepts that could be used to guide the design. One approach might be to consider the 'panoramic view' of the ocean and dunes to be the concept. In this case the dining and living rooms are placed on the top floors to take advantage of the view, and the bedrooms are located on the bottom floor - an upside down house.

There are consequences to this arrangement. While the view is excellent, the house is also visible from the beach and is not private. There is a sharp division between the internal and external environment of the house. Because the living room is elevated, it is detached from the outside environment. To reach the ground and the beach it is necessary to go down a long flight of stairs.

If, on the other hand, 'privacy' is chosen as the concept, then the living and dining rooms are placed on the lower level, nestled in the dunes, blocked from view. In this configuration, the internal living spaces and external decks have an intimate relationship with the surrounding landscape, but the view is a different one. It is of the dunes rather than the ocean. The same design situation results in completely different solutions, because the design concepts are different.

Take another example - OS/360. Here is concept was 'complete' (one OS to serve all needs). While JCL permits almost infinite adjustment and configuration of the operating system, it is time consuming to learn and difficult to use. It certainly is not user friendly. Another design concept would have produced a different solution.

4.2 Boundary

The boundary defines what is inside the system, what is external to it, and what crosses between the two. Choosing the location of the system boundary is an important design decision [25]. Not only does this establish the scope of the system, but it is likely to be a major factor in determining the amount of resistance to be encountered during implementation (based on an analysis of the redistribution of power [22]) and consequently the probability of success.

Boundaries can be conceived of as being physical barriers that separate the system from its external environment. Messages or transactions that cross the barrier represent inputs to the system or outputs from it, although some care has to be taken when establishing the location of human operators in this model (whether they are considered part of the system and consequently internal, or whether they are considered external).

System boundaries can be shown in Data Flow diagrams [3] using the context diagram, although no methods are provided to investigate trade-offs in boundary location.

4.3 Division Of Labor

Another key design issue is the allocation of tasks between between a computer and the human operator, as well as the basis upon which this decision is made [23]. A large number of combinations are possible. For example, a system may be fully automatic with the operator only playing a role when a malfunction occurs, or completely manual with the operator performing all tasks (see section 4.3.1). It is much more likely, however, that the operator will perform certain tasks while the system performs others. The question is then, which tasks shall the operator perform and which shall be performed by the computer. Tasks may be statically assigned to a processor (human or computer) based upon a variety of rationales, such as, selecting the processor that is best suited to perform the task. Another approach is to assign tasks dynamically based on a variety of rationales, for example, selecting the processor that has the least load. See Turner and Karasek [23] and Rouse [26] for a more complete discussion of these points.

As part of task allocation design, care must be taken to insure that a system control strategy is explicitly established. This is necessary for internal consistency and integrity, and to be sure that there is no confusion as to which processor (computer or human operator) is to perform a task.

4.3.1 Operator Functions -

Another important design decision that is frequently overlooked is defining the role of the operator. Too often, the operator's job follows implicitly from the design of the computer portion of the system. It is a result of the design rather than the impetus for it. Thus, it is critical to identify the tasks the operator is to perform, and to make these the starting point for computer system design rather than the reverse. It is important to consider the mix of tasks to be performed by the operator, the interdependence with other jobs, and to provide adequate autonomy for the operator [23]. It is also necessary to consider the skill level required of the operator and the training that will be needed.

4.3.2 System Functions -

Most effort expended in design is directed to identifying the functions that an application system is to perform. The trade-off is usually between increased functionality vs. complexity (and consequently cost). Functions can be described as processes (bubbles) in data flow diagrams and detailed in pseudo-code or decision tables. Although substantial effort is expended in working out the details of functions, they largely follow from other design decisions (such as, boundaries or division of labor) that are often implicitly made. One of the purposes of this check list is to force these decisions out into the open so that they can receive the same amount of attention usually afforded system functions.

4.4 Decomposition

In order to deal with the complexity of most application systems, some method must be found to decompose (or expand) the system. The approach most frequently followed in design methodologies is a top-down, breadth-first expansion. Yet, the real issue is what rationale shall be used as the basis of the decomposition, because this determines how the system is perceived and understood.

One approach is to decompose a system functionally. Thus, in the case of the commodity trading system, it would be divided into parts that arrange a deal, arrange a contract, arrange for shipment, arrange for insurance, etc. Another way this might be accomplished is to use generic processing functions, such as data editing, file maintenance, report writing, etc. as the basis for decomposition. The method of decomposition selected is likely to influence which aspects of a system receive attention. It also influences the heuristics used to determine completeness and consistency.

4.5 System Structure

The structure of a system is composed of two parts. One, processing organization, represents the work organization or flow of the system. The second, data structure, is a representation of the way data elements are related in the system.

4.5.1 Processing Organization -

The flow of work (of a system) may be organized in a number of different ways. At one extreme, all of the work of a particular type may be handled for all products at the same time. This 'flow shop' model has advantages including low overhead, simple control structures, and specialization of function. At the other extreme, all of the work required to complete one product can be performed at the same time. This 'job shop' model has the advantage immediate and customized attention to the job, and it permits the jobs of operators to be designed with considerable task variety. Processing organization interacts with operator functions (section 4.3.1) and the processing structure of the system (i.e., whether the system is batch or interactive) and consequently with performance (section 4.7).

4.5.2 Data Structure -

Data structure is the organization of the data used by an application system. This may range from simple organizations, such as, sequential files with records and fields, to more complex structures, such as networks and relational schemes. The key issues are how data are to be used, the selection of indices, navigating (traversing) data structures, and the updating of files.

4.6 Operating Sequence

The operating sequence is the time ordered actions that take place in normal operation of an application system. Identification of these actions is useful in determining whether all the necessary functions of the system have been identified. It is a perspective that can aid in thinking through a system because, while tracing operational steps, certain omissions may become apparent.

4.7 Performance Measures

Every application system requires a control portion to monitor proper operation. Often this represents a significant part of the system, for example, controls in a file maintenance system or back-up and recovery features. Identifying performance measures that can be applied to a system at an early point in the design process is helpful in establishing the control structure of the system. Performance measures also provide a criteria for rejecting inappropriate solutions as well as keeping the designer focused on important aspects of the system. Consequently, they should be developed in conjunction with the expansion of system needs.

4.8 Extent Of Change

One dimension of design that is seldom explicitly identified is the extent of change of the new system over the system that it replaces. If no system exists, then all aspects of the new system represent change. The more normal situation is that a system of some sort is present, and the new system only represents change in certain aspects.

Recognizing those areas with significant change is important, from a practical standpoint, because people influenced by them are likely to produce the greatest resistance to the new system. If too much change is attempted the implementation will likely fail. Areas of change should be evaluated in terms of shifts in power and in job content, and defensive strategies adopted [36]. From the standpoint of implementation strategy, an analysis of changes will usually reveal stake holders to placate. Under certain circumstances accommodating them may necessitate redesign.

4.9 Discussion

These nine elements represent perspectives from which to consider design. Design is a search for conflicts and constraints. These perspectives represent categories in which conflicts and constraints are likely to be found. The system concept is necessary in all situations to maintain consistency among design decisions. The boundary also must be considered in all situations, although it tends to be stated implicitly as part of the need. This statement is frequently incomplete or changes as the design problem unfolds. The division of labor, operator and system functions, system structure, processing organization, and data structure are the dimensions of reconciliation. Although they follow, to some extent, from previous decisions, they are the primary arena in which design of IS is conducted. Decomposition, operating sequence, performance measures, and extent of change are important considerations that may modify primary trade-offs.

Design at this top-level should not be confused with detailed design at the system or program level. Detailed design is concerned with expanding the design in a particular instance. As such it addresses different issues, for example, the location of data elements on a display screen. While there are many specific details to be worked out, much follows from the top-level design, or from accepted practice. In this sense, detailed design is analogous to design development in industrial design or engineering (section 2.4) while top-level design is similar to the design idea (section 2.3). In top-level design there are much fewer guidelines and the designer has great latitude.

One might say that there are two categories of design factors: subjective and objective ones. Subjective decisions concern the system concept, system boundaries, and the other issues discussed above. Objective decisions, such as a particular file structure, follow from the subjective decisions. The difficulty has been that we have not acknowledged, explicitly, the presence of subjective factors. The consequence of this has been that, in many cases, objective decisions appear to be arbitrary.

There are a number of design frameworks for IS, the most common being the 'life cycle' description of the system implementation process discussed in Section 3.5. Another is the top-down, breadth-first design decomposition of structured analysis [3]. While useful as a means of portraying a system description, the research evidence suggests that people (designers) think in a much more fragmented, associative manner, more akin to bottom-up design. If this is the case, then a design approach that more closely parallels their design process will probably be of more use in actual practice.

A third model is the data flow model, where an enumeration of the data elements and their relationships is used to drive the remainder of the design. A fourth approach is the system theory methods of Churchman [28]. A fifth is the socio-technical method which first details the job to be performed and then uses this to drive the computer (application) system design [10, 27]. Finally, a sixth approach involves prototyping [30], where the system is evolved over time through actual use. These models stress particular perspectives to design that may not apply in all circumstances.

The approach to design provided here, based strongly on a system concept and a check list of factors to be considered in design, is intended to be quite general. It is complementary to, rather than competing with, these other design strategies. In this approach, the concept is used as the basis of deciding design conflicts and in generating candidate solutions. It explicitly recognizes conflict and attempts to provide a context for its resolution. The other models suggest that conflict does not exist, and consequently, they provide no means for its reconciliation.

The most difficult problem in design is deciding which issues, of the many possible, to focus attention on. The creative sandwich approach recognizes the role of experience in identifying what is important in the design situation, and identifies activities where energy should be expended.

4.10 Teaching System Design

So far this paper has not considered specific issues in teaching design. Current practice either takes detailed design problems and investigates the implications of alternative solutions, or the student is given a design situation and is expected to produce a detailed design (file layouts, input and output formats, pseudo-code, etc.).

There are a number of difficulties with these approaches in the teaching of top-level design. In the first case, a set of principles is provided that purport to represent common practice. As Shneiderman [29] observes, these principles frequently are in conflict. In addition, there tends to be no rationale for selecting among different alternative solutions. In other words, many of the detailed design decisions appear somewhat arbitrary (although it may be possible to say why something should not be done a particular way).

In the second case, where the student is given a design situation, it tends to be presented in such a complete and detailed manner that the design solution follows automatically. There just is no design problem. In situations where conflicts and gaps do exist in the description, students cope ingeniously by making assumptions that translate the problem into one where the solution is trivial. For example, transposing it to a hardware evaluation problem or a choice between different file organizations. This is not to say that hardware evaluation or file organization are not important detail design decisions that students must learn to handle, they are just not top-level design decisions and they involve different thought processes.

While detailed design is an important aspect of the design process and a necessary starting point, it should not be confused with top-level design. Detailed design is concerned with reasonably well structured trade-offs between a relatively few, easily defined and similar alternatives. Top-level design deals with more global issues, such as setting the boundary of a system, working out the division of labor between human and machine, or organizing the flow of work in a system. These decisions are more ill-structured that those encountered in detailed design and there are many alternatives and implications that are not apparent.

The question arises, how is top-level design taught? The approach used at NYU is to focus attention on these issues directly. Several exercises illustrate this. In one, students are asked to find an advertisement of an object they would like to own. They are then asked to deduce from the advertisement the nine design elements (sections 4.1 to 4.8) of the object (e.g., what is the system concept?). After working backwards from the product to the likely design decisions, the student is asked to use the nine element framework to describe their own design for a specific problem (provided in a paragraph description). Relationships between the descriptions of the various design elements are used as the basis for evaluating the designs.

5.0 CONCLUSION

This paper has observed that the life cycle model of system design suggests a top-down, breadth-first expansion of the design problem using a linear-iterative process. This approach does not acknowledge the role of conflict, experience, or aesthetics in design and thus differs considerably from actual experience.

A contrasting and complementary description, the creative sandwich model, stresses the role of experience and individual values in design. Conflict is encouraged because it indicates factors that must be reconciled. Expansion of the design problem is bottom-up, depth-first based on an associative process. Requirements are never fully articulated and they can not be separated from potential design solutions. Thus, the problem and the acceptable set of solutions converge over time.

Nine elements or design perspectives are identified as a useful checklist of issues to be considered at this top-level of design. Of these, the system concept, which forms the basis upon which consistent design decisions are made, represents the greatest departure from current thought. It is our contention that good designers intuitively develop a system concept as the problem and its possible solutions unfold, and then use this as the basis of future design decisions. But, this conjecture remains to be shown.

As with most hesitant first steps in a new direction, this paper raises many issues that have been inadequately answered. Hopefully, others will find the topic challenging and, as a result, our knowledge of the process of design will improve.

Bibliography

[1] - Semprevivo, Philip C., 1982, Systems Analysis, SRA, Chicago.
[2] - Burch, John G. and F. R. Strater, 1974, Information Systems, Hamilton, Santa Barbara, CA.
[3] - DeMarco, Tom, 1978, Structured Analysis and Structured Specification, Yourdon, New York.
[4] - Teichroew, D. and Hershey, 1977, "PSL/PSA," IEEE Trans. Software Engr., 3, 1, pp. 41-48.
[5] - Davis, Gordon, 1982, "Strategies for Information Requirements Determination," IBM Sys. J.
[6] - Carlson, Walter M., 1979, "Business Information Analysis and Information Technique," Data Base, 10, 4 pp. 3-9.
[7] - King, William R., 1978, "Strategic Planning for Management Information Systems," MISQ, 2, 1 pp. 27-37.
[8] - Rockart, John F., 1979, "Critical Success Factors," HBR, 57, 2 pp.81-91.
[9] - IBM Corp., 1981, Business Systems Planning - Information Systems Planning Guide, GE20-0527-3.
[10] - Bostrom, R. P. and J. Stephen Heinen, 1977, "The Application of Socio-Technical Theory," MISQ, 1, 4 pp. 11-28.
[11] - Mumford, Enid, 1981, "Participative Systems Design: Structure and Method," SOS, 1, pp. 5-19.
[12] - Lundeberg, M., G. Goldkuhl, and A. Nilsson, 1981, Information Systems Development: A Systematic Approach, Prentice-Hall, Englewood Cliffs, NJ.
[13] - Beer, Stafford, 1966, Decision and Control, John Wiley and Sons, NY.
[14] - Archer, L. Bruce, 1964, "Systematic Methods for Designers," Design, 173-183 (1963-64).
[15] - Harvey, A., 1970, "Factors Making for Implementation Success or Failure," Mgt. Sci., 16, 6 pp. B312-B321.
[16] - Brooks, Frederick P., 1975, The Mythical Man-month, Addison-Wesley, Reading, MA.
[17] - Malhotra, A., J. C. Thomas, J. M. Carroll, and Lance Miller, 1980, "Cognitive Processes in Design," J. Man-Machine Studies, 12, pp. 119-140.
[18] - Jefferies, R., A. Turner, P. Polson, and M. Atwood, 1981, "The Process Involved in Designing Software," in J. Anderson (ed.), Cognitive Skills and their Acquisition, Erlbaum, Hillsdale, NJ.
[19] - Jackson, M. A., 1975, Principles of Program Design, Academic Press, NY.
[20] - Boehm, B., 1975, "Software Design and Structuring," in E. Horowitz (ed.), Practical Strategies for Developing Large Software Systems, Addison-Wesley, Reading, MA.
[21] - Boland, Jr., Richard J, 1978, "The Process and Product of System Design," Mgt. Sci., 24, 9 pp. 887-898.
[22] - Markus, Lynne, 1982, "Power, Politics, and MIS Implementation," CACM, 26, 6 pp. 430-444.
[23] - Turner, Jon A., and Robert Karasek, Jr., 1984, "Software Ergonomics," Ergonomics, 27, 6, pp. 663-690.

[24] - Ginzberg, M. J., 1981, "Early Diagnosis of MIS Implementation Failure," Mgt. Sci., 27, 4 pp. 459-478.

[25] - Turner, Jon A., 1982, "The Role of Behavioral Science Models in Information Systems Design and Research," Information and Management, pp. 207-213.

[26] - Rouse, William B., 1981, "Human-Computer Interaction in the Control of Dynamic Systems," ACM Comp. Surv., 13, 1 pp. 71-99.

[27] - Mumford, Enid, and Mary Weir, 1979, Computer Systems in Work Design - The ETHICS Method, John Wiley and Sons, NY.

[28] - Churchman, C. West, 1968, The Systems Approach, Delta, NY.

[29] - Shneiderman, B., 1980, Software Psychology, Winthrop, Cambridge, MA.

[30] - Naumann, J. D., and A. Milton Jenkins, 1982, "Prototyping: the New Paradigm for Systems Development," MISQ, 6, 3 pp. 29-44.

[31] - Feigenbaum, Edward A., 1983, "Artificial Intelligence," IEEE Spectrum, 20, 11, pp.77-78.

[32] - Alexander, Christopher, 1964, Notes on the Synthesis of Form, Harvard University Press, Cambridge, MA.

[33] - Koestler, Arthur, 1964, The Act of Creation, Macmillan Co., NY.

[35] - Weber, E. Sue, 1985, "Cognitive Processes Involved in Solving IS Design Problems," to be presented at ICIS in December, and to appear in the Proceedings.

[36] - Keen, Peter G. W., 1981, "Information Systems and Organizational Change," CACM, 24, 1 (January), pp. 24-33.