COGNITIVE MODELS OF SYSTEM DESIGN

Jon A. Turner

May 1986

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

Working Paper Series

CRIS #130

GBA #86-80

To appear in **Critical Issues in Information Systems Research**, R. Boland and R. Hirschheim, eds., John Wiley, New York.

Introduction

Quite simply, the fact is that we do not understand very much about designing complex, computer based information systems. I mean that we don't know what system design consists of, we don't know how it is done, and we don't know how to teach it. Furthermore, our lack of knowledge about the process of system design is the greatest single barrier to improving our ability to apply Information Technology (IT) and to increasing system development productivity, a major goal of most information system departments, executive management, and the industry as a whole.

This is not to imply we can't design information systems, for that is obviously not the situation. There are many examples of successful systems. But we do not understand well the *process* of design. And without that understanding we can never systematically apply it, or improve it.

This is not a problem unique to information systems. There there is little agreement or understanding as to what the process of design involves in other fields, for example, architecture or engineering [Alexander 64]. The situation is just more pronounced in information systems. Engineering and architecture are fields that consistently produce objects through a process of design. And the computer itself is one of the best examples of an artifact that was the result of conscious conceptual and practical design activity.

But information systems are different than most other

artifacts in three ways. First, they are abstract and not materialized in a form that is easy for people to comprehend as a whole. After a building is constructed it is quite straight forward to understand it and to respond. It is not easy to visualize an information system. Relatively few people have the skill, or prospective needed to comprehend it, even after it is constructed. People come in contact with only a portion of the system forming, at best, a partial view.

Second, an information system needs to correspond to a complex, non-specific set of human behaviors as well as a set of explicit data transformations. It must reflect accurately the tasks that people perform and the interactions among them. Rather than being an arbitrary form, such as a building, an information system has a structure that is dictated by a group of poorly understood, inconsistent human activities. Third, because an information system is eventually represented by a computer program, it's correctness is subject to verification. A building, by comparison, can not be judged correct, only appropriate.

These distinctions in the form of the artifact suggest differences in the process of design. Design of an information system must not only accommodate the normal design activities involved in engineering and architecture, it must provide means for comprehending human behaviors and representing them in a consistent fashion. It must reconcile the imprecision of human activity with that of precisely specified operations required by a computer. These transformations must be explicitly stated rather than left to accepted convention. Consequently, the

design of an information system is more demanding and more mysterious than that of many other artifacts.

There are two basic strategies used for the design of information systems: the life cycle approach; and evolutionary design, best typified by prototyping. The life cycle approach consists of three overlapped and interlocking activities: analysis, design, and implementation. While these activities are all highly related and frequently inseparable, it is usual practice for a description of the system to be produced in each phase as a means of conveying the information gained and decisions made to following stages. This is particularly true when more than one person is working on a project and they must communicate. Thus, the analysis phase produces a requirements statement or needs analysis, while the design stage produces program specifications or data flow diagrams (with a data dictionary, pseudo-code, and structure diagrams). The implementation stage, of course, produces running code. These stages could also be considered different levels of abstraction, or detail of the system. Most of the methodology that comprises software engineering applies to the implementation stage, or the later portion of the design stage that concerns program design; or are conventions for describing a system at one phase or another[1].

---

[1]While these conventions are important for the purpose of consistency and in communicating detail design they do not *directly* contribute to an understanding of requirements. The detailed design is seen to follow from a statement of requirements. Some authors contend that this documentation is never read and is impossible to keep consistent [McCracken 81].

In contrast, prototyping combines all of these activities in one step. A preliminary understanding of the requirements are gained and a working system is built immediately. Adjustments are accomplished by feedback obtained from actual use by the client. Complexity is introduced through refinement over time. In a prototype, the requirements statement, or data flow description *may never* exist separate from the materialization of the system.

In both of these approaches the quality of the resulting system is determined largely by the degree to which the designer understands the requirements, or needs of a system. Both approaches suggest that requirements analysis is performed top down, from general to detail. Requirements analysis tends to be accomplished as part of a closely spaced sequence of activities at the beginning of a project in the life cycle approach. In prototyping, requirements analysis is performed continuously over the duration of the project. Both rely on a dialogue between designer and users to elicit an expression of needs[2].

In both approaches, there is relatively little methodology to guide the designer, or the user for that matter, in obtaining an expression of needs. The presumption is made that 1) users know what information they need, and 2) they will freely disclose it if asked. As Ackoff [Ackoff 67] points out, this is unlikely

---

[2]It is symptomatic of our lack of understanding of the process of design that the most useful skill in accomplishing this activity, interviewing, is not included in most courses or textbooks.

to be the case since most users do not know what information they need (and, incidentally, wouldn't know what to do with it if they received what they requested). As Davis notes, "simply asking prospective users of the information systems to specify the requirements will not suffice in a large percentage of cases" [Davis 82], due to constraints on individuals as information processors, the variety and complexity of the information requirements, and the patterns of interaction among users and designers in defining requirements. If this were not enough, free disclosure assumes an absence of organizational politics, which in most settings is unrealistic [Keen 81].

If simply asking users to state these requirements won't suffice, then different and expanded approaches are needed. Yet, the prescriptive literature is silent on what these approaches might be, or how the process really works[3]. If a true understanding of this process is to emerge, it must be based on the cognitive activities individuals invoke when designing.

In this chapter I will review research findings on the cognitive process of design, describe how design is characterized in engineering and architecture, propose a way to conceptualize design that is useful for IT, and suggest further avenues of research. My goal is to draw together what is known about

---

[3]Davis does identify the broad strategies for determining information requirements as: asking; deriving from an existing system; synthesis from characteristics of the current system being used; and, discovering from experimentation with an evolving system. The difficulty is that, in practice, all of these strategies are used.

information system design so the process may be better understood.

Research

One way to characterize design problems is that they consist of a set of initial conditions and a *goal* but no immediate procedure that will guarantee attainment of the goal. Beginning at the initial state, operators or transformations are used to move from one state to another until a final state is attained (hopefully the goal). In real world design problems, however:

> ... 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 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 [Malhotra 80, p.120].

This characterization of real world design problems contrasts sharply with the idealized formulation presented above. It suggests that the goal is evolved along with adjustments in initial conditions rather being known a prior. It focuses on *properties* of the design *solution* and how well they match the functional requirements derived from the design *goal*. Properties of a design solution arise from a combination of design *elements*, indivisible units with certain characteristics, and the design *organization*, the way the design elements interact and fit

together[4]. More importantly, it shows the central role of dialogue in clarifying some of the ambiguities. In practice, however, only some of them will be resolved and the issue becomes identifying what guides the discrimination between significant and insignificant.

Malhotra [Malhotra 80] in studying dialogue between people attempting to solve real world problems found that it consisted of the translation of design goals into functional requirements that candidate designs must meet and the generation of designs to meet the requirements. He concluded that the dialogues were more complex, in reality, 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

---

[4]For example, part of the solution for an interactive system may be a set of data elements arranged in screen formats which are then invoked in different sequences under particular conditions.

solution. Much of this process was *implicit* and *unstated*[5].

From this study it appears that generation of solutions seems to consist of attempting to find design elements that meet functional requirements of the problem and then tying them together into a *coherent* design. This corresponds roughly to bottom-up design. Although this was not the only design strategy exhibited in Malhotra's study of dialogues, it was the predominant one and it seemed to be encouraged by the fragmentary presentation and elaboration of requirements.

The results of this study suggest that problem definition and solution generation are not independent activities; they are inter-related. Consideration of potential solutions raises questions about potential requirements which then give rise to new requirements. This is sort of a hole finding-hole filling activity. Requirements and solutions migrate together toward convergence. The fragmentary nature of the dialogues suggest that they play an important role in stimulating cognitive

---

[5]Malhotra noted that the dialogues were composed of cycles, each one broken into a number of mutually exclusive states he defined as 1) goal statement, 2) goal elaboration, 3) solution outline, 4) solution elaboration, 5) solution explication, and 6) agreement on solution. A diversity of content underlay this apparent regularity of structure. For example, although discussions and solution suggestions 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 suggests that design involves a strong associative component and that deeper structure, to the extent one exists, has yet to be revealed.

processes, rather than solely conveying predetermined information.

The prototype development strategy seems to match this problem definition/solution generation process more naturally than does the sequential and compartmentalized life cycle approach, which may partially account for the popularity and success of prototyping and evolutionary design as implementation strategies in end-user computing (EUC). This is not an either/or situation; but rather an observation that in the life cycle approach it may be unrealistic to expect that requirements will ever be completely articulated at the beginning of the project, and unless provisions are made to capture design solutions that are generated as part of the requirements definition activity, important information may be lost.

A related question is whether, for any design situation there exists a solution that is clearly superior. If no superior solution exists, and there are many acceptable ones with little to choose among them, then the solution generation and evaluation problem is quite different. Instead of searching for *the* correct solution, an acceptable solution only need be recognized.

One way to investigate this issue is to see whether people working separately on the same problem arrive at similar solutions. Turner [Turner 85] studied the similarities and differences in solutions provided by experienced students who

were all given the same design problem[6]. The analysis reveled many more differences than similarities. There was wide variance in what was included in solutions; arcs, names and contents of data flows were different, as were processes. Subjects made a number of different assumptions, many in direct conflict with the written description of the problem.

Further analysis showed that there appeared to be four different strategies used by subjects to decompose the problem. The first and most common was a functional decomposition strategy, the grouping of activities around major business functions being performed. There was, however, considerable variation in the functions selected as the basis of decomposition and the ways they were interconnected. The second strategy followed was process orientated. Subjects recognized certain common information processing functions, such as updating a file, and grouped these together. The third strategy, similar to the first, was functional decomposition with the function selected because they occurred at the same time. The fourth was a combination of the first three.

When questioned, students could explain the logic of their approach to decomposition quite clearly, but they were unable to convince their colleagues (the other subjects) of the superiority of their approach. It was hard to escape the conclusion that how

---

[6]Data flow diagrams, used to represent solutions, were compared on the basis of 1) boundaries; 2) data flows, including arcs, names, and element contents; and 3) process functions as represented by lower level diagrams.

subjects thought about the problem influenced their decomposition strategy, and how they thought about a problem was largely a function of their *background* and *experience*.

One possibility is that these results are due largely to the use of students as subjects rather than experienced professional information system designers. Malhotra [Malhotra 80] in another study asked experienced subjects to design a query system. An analysis of the resulting designs showed wide variation in approaches taken and in solutions. The researchers concluded that the sub-goals and solution strategies generated from 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, where subjects were to design the query system in more detail, Malhotra found the solutions were all different - in module content, data structures, and algorithms. In addition, the solutions contained errors, inconsistencies, and unwarranted assumptions. He concluded that unlike engineering, it was difficult to tell whether information system design was complete, consistent, or even met functional requirements.

In summary, the commonly held notion about the design of information systems is that it is an ordered process, performed at the beginning of a project (in the life cycle strategy), a methodology which when applied will produce the same result; that it is top-down, moving from general to specific; and that definition of requirements proceeds design solution. Research findings suggests the opposite. Design is ad hoc and

associative, the process is individual and experientially based, the products produced (by different designers) are usually different, much of design proceeds bottom-up, and solution and problem definition are intertwined.

Furthermore, there does not seem to be a common procedure for producing a design solution; different methods of problem decomposition are used, there seems to be no common mechanism for producing sub-goals, different operators are invoked, unwarranted assumptions are made, solutions are rife with errors, and there are no ways short of actually building a system to uncover errors and inconsistencies. In short, there does not appear to be convergence on one solution for any particular situation, nor does there seem to be strong problem solving models that underlay design in information systems.

Design as Portrayed in Engineering and Architecture

The art of industrial design has been defined as "selecting the right material and shaping it to meet the needs of *function* and *aesthetics*." [Archer 64][7]. These two factors, functions and aesthetics, fundamentally different in nature and likely to be in conflict, must be reconciled by the designer, and this, then, is

---

[7]*Function* is the purpose or function the finished product is to perform and this must be understood by the designer and represented in the product. *Aesthetics* are subjective considerations based on judgments that are shaped by values of the designer. It falls into two broad categories: descriptive aesthetics, which deals with empirical facts about perceivable qualities of an object and the statistics of preference; and ethical aesthetics, which is concerned with good or bad taste, or appropriateness.

the design problem.

Design is considered an art because the rules for moving from one configuration, or state, to another, the *operators* as they would be called in computer science, in either of the two domains (functions or aesthetics) are not well defined. Neither are the states.

Functions spring from a fundamental understanding of the purpose of an object, or the activity being performed. While it is quite possible to work out (by scientific methods) who likes what, in what circumstances, there are no immutable truths in aesthetics. Its essence is *choice* with the aim of *appropriateness*, and the criteria are the center of gravity of *all* prior choices. A special problem is that the designer must not only be aware of his own standards and values, but he must understand those of others, and foresee their probable future choices. In a majority of cases, aesthetics is handled more quickly and appropriately by intuition, provided there is an adequate body of prior experience to base it upon than a formal method. What tends to be missing in descriptions of information systems design is acknowledgement of the role of aesthetics, or any activities based on intuition.

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. 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 this does not follow directly from a statement of the problem. It is characteristic of creative solutions that they are seen to be apt *after* the fact and *not* before. Consequently design may be said to involve creativity and originality.

Design suggests purposeful seeking after solutions rather than idle exploration. It also implies that certain *limitations* exist, often in the form of obstacles or gaps, which constrain acceptable solutions. In information systems design, understanding the problem involves not only understanding needs, but also these constraints, and in may cases, these constraints are unstated, or implied. Thus, the need for a *fundamental* understanding of the object being designed (or the design situation).

The art of design is that of *reconciliation*. In general, design of industrial objects involve three categories of factors: human factors (motivation, ergonomics, and aesthetics); technical factors (function, mechanism, and structure); and business factors (production, economics, presentation, and support). Some of these factors, such as economics, relate to matters of fact that are 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 these crucial decisions have been made, the process proceeds with detailing of the design, for example, producing working drawings in architecture, or a working prototype in information systems[8]. 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.

There still remains the crux of the design problem, the *creative leap* from specifying the problem to finding a solution. Industrial designers appear to establish a first approximation based on *prior experience* [Archer 64]. This means finding connections between the goals, in terms of the attributes of a good solution and the facts of the situation as mediated by the designers knowledge and experience. Constraints serve to bound the problem, rule out certain solutions and provide useful clues to hidden needs or where possible solutions may be found.

Designers appear to search their minds for solutions by examining all kinds of *analogies* [Archer 64]. They look at other people's design solutions to determine whether something along those lines would answer their problem. They look at phenomena and artifacts in the most unlikely fields. If this process still yields no result the designer tries to *reformulate* the problem in a manner that permits one of the solutions previously uncovered

---

[8]It is well known in architecture that in executing the detailed design conflicts arise and inconsistencies are revealed that require a rethinking of the creative phase. Often the original creative solution is abandoned and a new one conceived for the new situation.

to be used. Only as a last resort does the designer attempt deductive reasoning, proceeding from analysis of data to necessary conclusion.

In computer science terms, the industrial designer attempts a backwards, depth-first search from potential solutions (based on prior experience) to parameters of the problem, with missing data and constraints serving as cues to potential solutions, evolving the problem[9], or bounding the search. If no solution is found the designer constructs a new network composed of solutions to similar (and dissimilar) problems used by others[10]. The designer then attempts to reformulate the problem in a manner that permits use of an uncovered solution. If one is still not found, the designer attempts a forwards, breadth-first expansion of the problem to see if it leads to a solution.

Experience acts both to define the set of initial acceptable solutions and to influence how facts and sensory data are interpreted. Observers contribute to their perception of the phenomenon before them from their own experience by either addition, or subtraction. This requires a delicate balance. One needs a group of wide and rich range of experiences to stimulate flexibility and fantasy in thought in order to recognize those

---

[9]Review of Malhotra's dialogues suggests that a good portion concerns verification; obtaining feedback from the client that the designer has understood some specific aspect of the problem.

[10]I suspect that this step has a lot to do with injecting creativity into the solution as the process of attempting to understand someone else's way of thinking (why the solution works) stimulates your own thought.

aspects of the design problem that are *important*. Yet, this must be done without biasing what is observed. I believe experience serves an important role in focusing the designer's attention on key (pivotal) aspects of the problem, while permitting him to disregard the great majority of (irrelevant) data.

One of the frequently made mistakes in information systems is to presume that the objective portion of design involving, for example, documenting an existing system, constitutes *all* of the design activity. This view is incorrect because it does not recognize the creative decisions involved in defining the form the system will take and in recognizing what aspects of the problem on which to concentrate. But how shall the form of a system be described and what are the factors involved in information system design? Clearly, a new vocabulary of design is needed.

A Vocabulary for Information Systems Design

It is my belief that experienced information systems designers consider implicitly (that is, have developed refined procedures, or schemas, for) the following elements of design. No time sequencing of activities is implied; the issues presented are not necessarily resolved in the order in which they are listed. Or, are they likely to be the way people think about design. The cognitive processes involved in design seem to be associative and individual, rather than sequential. The elements presented are a *checklist* of issues that must be resolved when designing an information system.

Identification of these elements is based on my experiences

as a designer of information systems, my intuition, and my observations of industrial designers. They are presented here to make them explicit and in hope that, as such, they will serve as a new, somewhat more useful, vocabulary of design.

System Concept

Industrial designers make a distinction between a design *idea* and any one *embodiment* of it. The design idea is an invention, an abstraction, while the finished design is one of many possible embodiments of it. For example, in a patent application, the invention and a material embodiment of it are described separately. The description of the invention is interpreted literally and is deemed to cover all of the variations that the inventor wishes. The description of the material embodiment is interpreted freely and is regarded merely as an exemplar.

In order to serve as a guide in making consistent decisions and to resolve conflicts in information systems design, a system *concept* is needed. The concept is the rationale, or underlying theme of the system, for example, *minimal, or simple*. An elaboration of what the system should do is *not* the concept. The concept is a distillation of the system, its essence; analogous to the design idea used by industrial designers.

In OS/360 (IBM) the design concept was *complete*; one common operating system would support the company's complete line of computers and that system would have a complete set of features. While JCL permits almost infinite adjustment and configuration of the operating system, it is complicated, time consuming to learn,

and difficult to use. Another design concept (user friendly) would have produced a different solution, for example, TOPS-20 (DEC).

Boundary

The boundary defines what is inside the system, what is external to it, and what crosses between the two. The boundary establishes the scope of the system and, consequently, its size and complexity. If the boundary is set too wide, the system becomes so complex as not to be buildable; if is set too narrow, the system is trivial. Boundary decisions are particularly important in explaining (predicting) resistance to the implementation of a system based on an analysis of the redistribution of power.

Division of Labor

Decisions concerning the allocation of tasks between a computer and the human operator are another key design issue. A large number of combinations are possible, ranging from fully automatic, with the operator playing a role only when a malfunction occurs, to completely manual with the operator performing all tasks. In most practical systems, tasks are allocated to either computer, or human. The question then is the basis upon which this allocation decision is made, for example, by selecting the processor that is best suited to perform the task, or the one that is least loaded at the time[11].

---

[11] [Turner 84] provides a more complete discussion of this topic.

Too often the operator's job follows implicitly from the design of the computer (applications) portion of the system. It becomes the result of prior design decisions, rather than the impetus for them. Consequently, it is important to identify the tasks an operator will perform and insure that they make sense from the stand point of what is known about worker behavior, performance, and working life quality.

Most of the effort expended in design is directed at identifying the functions an application system is to perform. The trade-off is usually between functionally and complexity (cost). I maintain that these functions follow largely from prior decisions (such as system concept and boundaries) and the activities being performed[12]. This makes it all the more important that these design decisions be explicit.

System Structure

The structure of a system consists of two parts: the processing organization, representing the work organization, or flow of the system; and, the data structure, the way data elements are related. If the system is considered as transforming inputs to outputs, work organization refers to the manner in which these transformations take place. At one extreme, a unit of input can be completely transformed into output, invoking, in sequence, all of the necessary steps. Such an approach is responsive, because it permits predicting when the output will occur, but it incurs a high overhead and presents

---

[12]Or, as Davis observes, deriving the functions from an existing information system

difficulties in control. At the other extreme, the input can be held until all of the input of a particular category is assembled. This method is efficient (in terms of resources), but it is difficult to predict when output will arrive. Efficiency and functionality of actual programs depend on data structures actually selected.

Decomposition

In order to deal with the complexity of most application systems some method of decomposition (or, expansion) is needed. The approach most frequently followed in design methodologies is top-down, breadth-first expansion. Note, however, that this is just the opposite of the way industrial designers approach their problems. I suspect that information system development methodologies that support bottom-up, depth-first expansion and permit associative (ad hoc) thinking will be more successful than methodologies currently used.

Two basic strategies are followed in decomposition: functional, where the system is successively divided into parts on the basis of the business activity taking place; and data processing, based on the generic processing activity involved. The method of decomposition is highly leveraged because it influences how designers perceive the problem (its representation), what aspects of the problem receive attention (solutions and their parameters), allowable operators, and the value of the design produced.

Operating Sequence

Identification of the set of time ordered actions that must be performed in order to accomplish the purpose of the system. It is a useful check to insure that all needed functions have been defined and that those that have are used.

Performance Measures

Every system requires a control structure to monitor proper operation. Sometimes, as in file maintenance, this becomes a major portion of the system. Identifying performance measures that will be used to monitor performance is a cue in designing the control structure.

Extent of Change

Most systems represent an incremental change from some prior condition. Recognizing the extent of change imbedded in a system is another aspect of identifying the amount of resistance a system is likely to produce, and consequently, the risk involved in implementation.

Summary

These eight elements are dimensions within which an information system exists. Design is a search for conflicts among objectives and the means of resolving them, and constraints that bound the problem. These dimensions become the *space* in which design is played out.

The system concept is necessary to maintain consistency among design decisions. Boundaries establish the complexity of

the system[13]. Division of labor and system structure are basic design dimensions that establish the configuration of the application. Decomposition influences the way the designer and others perceive the system. Operating sequence, performance measures, and extent of change are cues to prompt for often overlooked factors.

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. Although execution of detailed design may influence top-level design, it addresses different issues and is much more constrained and directed.

There are two categories of design factors: subjective and objective ones. Subjective decisions concern the items discussed above. Objective decisions follow from them. The difficulty has been that we have not acknowledged, explicitly, the presence of subjective factors, with the result, that, in many cases, objective decisions appear to be arbitrary.

Implications for Research

The discussion above has been based on experience and conjecture. One obvious starting point is to search,

---

[13]Brooks [Brooks 75] has observed that management's usual response when a system has slipped schedule and over run cost is to add more manpower, which will only make the system later and cost more. The proper reaction is to trim the size of the project, which in our terms would be to make the boundary smaller and to reduce the number of functions.

empirically, for evidence supporting the presence and importance (or absence) of these notions. For example, good and poor information systems designs (bases on some objective criteria) could be compared in an attempt to establish the role a strong systems concept played (embodied in the good systems, while lacking in the poor ones). The good systems could be analyzed to see if they had selected operating points on the above dimensions that are consistent, while the poor systems may not have resolved these issues explicitly. Expert designers could be interviewed (observed) to determine the extent they consider these issues, and this could be compared with the behavior of poor (novice) designers. Although this research line is difficult from a methodological standpoint and subjective, I believe we need more detailed studies of the process of design to reveal what really goes on and to generate new conjectures for investigation.

A second line of research would investigate the design process, in more detail, at the cognitive level. While there have been no studies of information systems designers to determine the way that problems are represented and operated on, work has been done in understanding how people represent problems in other domains. Chi [Chi 81], in studying the representation[14] of physics problems in relation to the organization of knowledge in experts and novices, has shown that the quality of problem representation influences the ease with which a problem can be solved and the quality of the resulting solution. Her results show that the categories into which experts and novices sort

---

[14]An internal cognitive structure constructed by a person to stand for, or model a problem.

problems are different, although both are able to construct an enriched internal representation of it. Experts appear to categorize problems by underlying physics principles, a kind of deep structure, while novices categorize problems by their surface structure. With learning, advanced novices began to categorize problems by principles with gradual release from dependence on the physical characteristics of problems.

Chi's notion is that a problem can be at least tentatively categorized after some gross preliminary analysis of its features. After a potential category is activated, the remainder of the representation is constructed with the aid of knowledge associated with the category as an internal schema[15]. For experts, the schema includes potential solution methods. She concluded that experts perceive more in a problem statement than do novices. They have a great deal of tacit knowledge that can be used to make inferences and deviations from the surface features of the problem. Their selection of an approach (principle) to apply to solving a problem appears to be guided by this *derived* knowledge. The actual cues used by experts are not the labels themselves but what they signify.

The findings of Chi's study are consistent with the notions of the information systems design process set forth here.

---

[15]A schema is the category and its associated knowledge. That is, interpretation and processing rules consisting of both declarative and procedural knowledge, relating to the category. In Chi's study, the category was equated to the label a person used to access a related unit of knowledge and the knowledge was expressed as a network and production rules.

Problems and solution methods are bound together in a schema: bottom-up (data-driven) recognition of problem categories followed by top-down application of processing rules. This would be a reasonable explanation of the patterns found in Malhotra's dialogues. Chi's work suggests that we should be more interested in the ways designers represent problems and the operators they appear to apply in executing designs. Finally, to the extent the parallel holds between solving physics problems and designing information systems, if general principles of design exist they have not been recognized. We must continue the search.

Conclusion

I have argued that there is not much understanding of the process of designing information systems. Design is much more ad hoc and intuitive than the literature would lead one to believe. Rather than being separate, solutions and problems are interrelated, and solutions are an integral part of problem definition. It is incorrect to think that a problem has only one proper solution; there are many. Consequently, notions of closure and completeness must be re-thought. A good portion of information systems design involves aesthetics, yet there is no discussion of the aesthic in the field. Rather than pretending that it does not exist, it would be far better to acknowledge the importance of aesthetics and make it a central subject of attention and research. Subjective does not mean arbitrary. We should refrain from attempting to quantify subjectivity, although we certainly must understand its components.

There needs to be more awareness of the top level factors that drive detailed design. These design dimensions should be

made explicit and they should receive the same amount of attention that we lavish on such detailed design issues as data structures. In research, we need to understand how designers represent and manipulate problems. If we focus the energy and attention on these issues that they deserve, I'm confident that a major contribution will be made.

References

[Ackoff 67] R. Ackoff. Management misinformation systems. *Management Science* 14(4):b-147-156, 1967.

[Alexander 64] C. Alexander. *Notes on the Synthesis of Form.* Harvard University Press, Cambridge, MA, 1964.

[Archer 64] L. B. Archer. Systematic methods for designers. *Design* , 1964.

[Brooks 75] F. P. Brooks. *The Mythical Man-Month.* Addison-Wesley, Reading, MA, 1975.

[Chi 81] M. T. Chi, P. J. Feltovich and R. Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science* 5:121-152, 1981.

[Davis 82] G. B. Davis. Strategies for information requirements determination. *IBM Systems Journal* 21(1):4-30, 1982.

[Keen 81] P. G. W. Keen. Information systems and organizational change. *Comm. of the ACM* 24(1):24-32, 1981.

[Malhotra 80] A. Malhotra, J. C. Thomas, J. M. Carroll and L. Miller. Cognitive processes in design. *Journal of Man-Machine Studies* 12:119-140, 1980.

[McCracken 81] D. D. McCracken. A maverick approach to systems analysis and design. *Systems Analysis and Design: A Foundation for the 1980's.* North Holland, New York, 1981.

[Turner 84] J. A. Turner and R. A. Karasek, Jr. Software ergonomics: Effects of computer application design parameters on operator task performance and health. *Ergonomics* 27(6):663-690, 1984.

[Turner 85] J. A. Turner. *The process of system design: Some problems, principles and perspectives.* Technical Report GBA 86-101, New York University, Center for Research in Information Systems, 1985.