Fundamental Processes of Mechanical Designers Based on Empirical Data

LARRY A. STAUFFER & DAVID G. ULLMAN

SUMMARY *The fundamental processes of problem solving during the mechanical design activity are presented based on a study of human designers. The designers in this study solve problems by applying 10 specialized actions, called operators, which are grouped into the categories of 'generate', 'evaluate' and 'decide'. These operators are applied in unique sequences of which 95% constitute the following four local methods: 'generate and test' (23%), 'generate and improve' (8%), 'deductive thinking' (33%) and 'means end analysis' (36%). Furthermore, cognitive activity is reduced to a hierarchical architecture of tasks and episodes which reflect the goal structure of the designer. These descriptions demonstrate the abilities and limitations of the human designer's problem-solving performance, yielding an insight into how the man–machine interaction of design automation can be improved.*

1. Introduction

A description of the non-routine, problem-solving performance of mechanical designers is presented. It is based on the detailed analysis of verbal and video protocol data of five mechanical designers whose actions are defined by a set of 10 operators applied to the design state in order to achieve goals. The strategies they use for applying these operators, called local methods, are also identified. While previous research, based on empirical data, has been focused on human designers [1–6], none has investigated the designer's specific problem-solving skills to the detailed level accomplished in this research. (A summary of the referenced research on mechanical designers can be found in Stauffer [7].)

A study of how humans perform mechanical design is important since the process is so complex and yet poorly understood. As a consequence, it is difficult for researchers to discern how design should be improved or how improvements should be evaluated and validated. If researchers can better understand the design process, they will have an advantage in future development of computer-aided design (CAD) and knowledge-based systems (KBS) tools. Computer-based methods need to be as flexible as the designers who use them. Part of this flexibility requires a natural interface between future tools and designers, resulting in a compatible system. There may also be some very good reasons why humans design the way they do, which can be

Larry A. Stauffer, Department of Mechanical Engineering, University of Idaho, Moscow, ID 83843, USA; David G. Ullman Department of Mechanical Engineering, Oregon State University, Corvallis, OR 97331, USA.

incorporated into these systems. An explanation of designer performance demonstrates those design tasks that require the most time and energy, so yielding the greatest potential pay-off for design-automation tools. Finally, this study helps to define the fundamental problem-solving methods of designing.

Establishing the fundamentals is an essential element of developing a design curricula [8]. With these motives in mind, this paper is focused towards identifying how an engineer processes information to solve problems in a design context.

The model of mechanical designers is based on a psychological theory of problem solving developed by Alan Newell and Herbert Simon which has three fundamental assumptions [9]. First is the assumption that a person, specifically during problem solving, can be viewed as an information processing system (IPS). This IPS theory emphasizes performance rather than personal behavior or emotion. The heart of the IPS, as shown in Fig. 1, is a processor that converts information and provides an interface between the designer's memory and his/her external environment. The processor contains operators, a short-term memory (STM) and a controller. The operators are specific to the particular task domain and are the simplest processes to be analyzed in the theory. The IPS processes information with relatively few, specific operators. The STM, sometimes referred to as the working memory, has a limited capacity, nominally seven 'chunks' of information [11], and only the most recent information used by the IPS is held for a short duration. (A 'chunk' is a meaningful unit of information that a person uses.) The controller determines the sequence of operators to be executed and provides for the integration of information. Receptors and effectors provide an interface between the processor and the external environment.

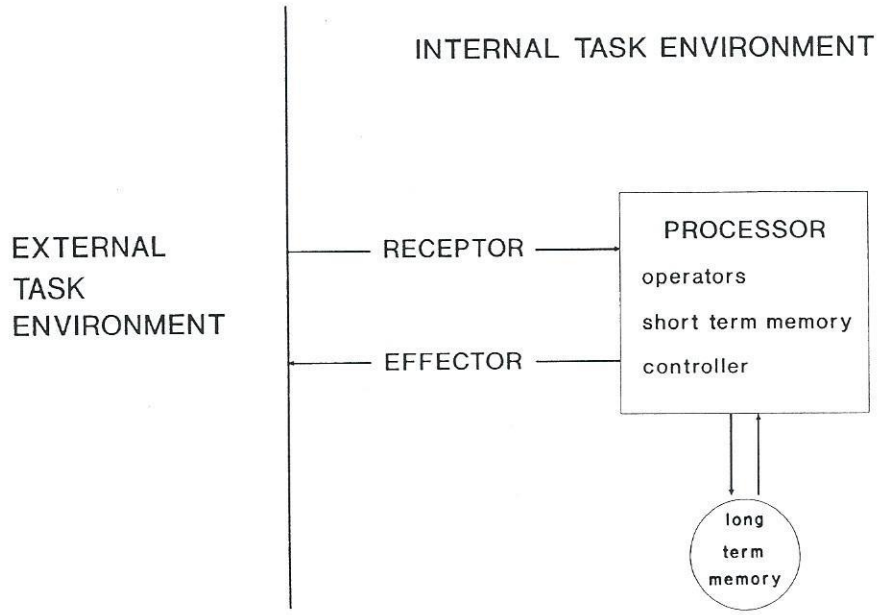

FIG. 1. Model of the designer as an information processing system.

The circle in Fig. 1 represents a person's long-term memory (LTM). The LTM refers to all the information (or knowledge) in a person's mind that has accumulated over a lifetime. The LTM stores this information by associating the relationships of

one chunk to another. While the capacity of the LTM is virtually infinite, access is slow, typically requiring 2 to 10 s to retrieve a chunk of information [11]. During problem solving, the processor accesses information from the LTM into the STM as it is needed by the operators. The execution of the operators by the processor changes the contents of the STM and thus progress is made in problem solving.

The second basic assumption is that problem solving takes place in a task environment consisting of both external and internal dimensions. The external task environment consists of those elements that can provide or receive information during problem solving—sketches, notebooks, computers, handbooks, colleagues, etc.—which can serve as a person's extended memory. The internal task environment consists of information about the problem, the desired outcome, and all other information that has been processed in the STM. All this information describes the person's knowledge about the problem and is called the 'design state'. The long-term, short-term and extended memories maintain a current record of the design state.

The third assumption is that problem-solving progress in the internal task environment can be modeled as a search, where a person moves from one intermediate state to the next in order to perform some task [12]. The search is made by the IPS when it applies operators to the present state. During problem solving, the state of the design continuously evolves when operators are applied until a solution is found.

The benefit of using the IPS model lies in its compatibility with computational structure. It is not that a person thinks like a computer, but rather, in part, that a computer may arrive at a similar solution to a problem by following this model. Modeling human problem solving as an IPS is not new: it has been applied to other areas of problem solving such as cryptarithmetic, chess [9, 13], and other well-structured problems. The IPS model has also been applied to ill-structured problems in the fields of architecture [14, 15] and software design [16], as well as mechanical design [4]. In these cases, the research goal has been similar—to identify a set of IPS elements specific to the domain in question.

2. Experimental Procedure

To gather data on the design process, experiments were conducted involving five engineers with 2–14 years of industrial design experience. Two open-ended mechanical design problems with incomplete, high level specifications were given to these designers. One problem required designing a mechanism to grasp and position a thin aluminum plate onto the surface of a chemical bath. To solve this problem required knowledge of kinematics and some actuation technology, such as pneumatics, or small electro-mechanical transducers. It was a one-off-type problem since only three mechanisms were to be built. This problem was based on a consulting contract completed by one of the authors. The other problem required the design of the contacts and housing for three button batteries connected in series to power a time clock. To solve this problem required knowledge of thin metal springs, basic electrical theory and plastic-injection molding. The project was product oriented; 1.8 million units needed to be built. This problem was developed with the cooperation of a major computer manufacturer.

Each engineer (subject) spent 6–10 h designing a solution to one of the two problems. During the work, each subject was asked to 'think aloud' as part of a technique from cognitive psychology to explore human thought, known as protocol analysis [17]. The subjects' efforts were recorded on videotape in order to provide a detailed report of the verbalizations, sketching and gesturing which took place during

problem solving. Details on the problems and experiments can be found in Stauffer [18] and Ullman *et al.* [19, 20].

3. Analysis of the Data

After the protocols were taken, the data were reduced into successively smaller segments in order to study them. (The experiments generated over 45 h of protocol data, each hour requiring roughly 40 h of study to understand the designer's problem-solving performance.) The protocols were systematically analyzed as the investigators watched the videotape, read the transcript and studied the sketches. The analysis process was iterative, requiring several passes through a section of data to understand the subject's performance. The analysis procedure was based on a process known as content analysis [21]. The goal of the research was not to understand *every* problem-solving process of each subject, but to identify the *types* of processes characteristic of the subjects. Full details of the analysis can be found in Stauffer [18].

The analysis of the protocol data has led to a hypothesis of how humans solve problems in a design context. In this paper we present an information processing model which describes the designer's problem-solving performance (Section 3.1). Sections 3.2 and 3.3. present an in-depth look at two fundamental elements—operators and local methods—which describe the actions and strategies that the subjects used in order to solve problems.

3.1 An Information Processing Model of Mechanical Designers

The IPS model from Section 2 has been refined in order to describe the problem-solving performance of a mechanical designer (Fig. 2). This model preserves the same structure as that of the IPS model in Fig. 1 but has been tailored to reflect the uniqueness of mechanical design based on the protocols.

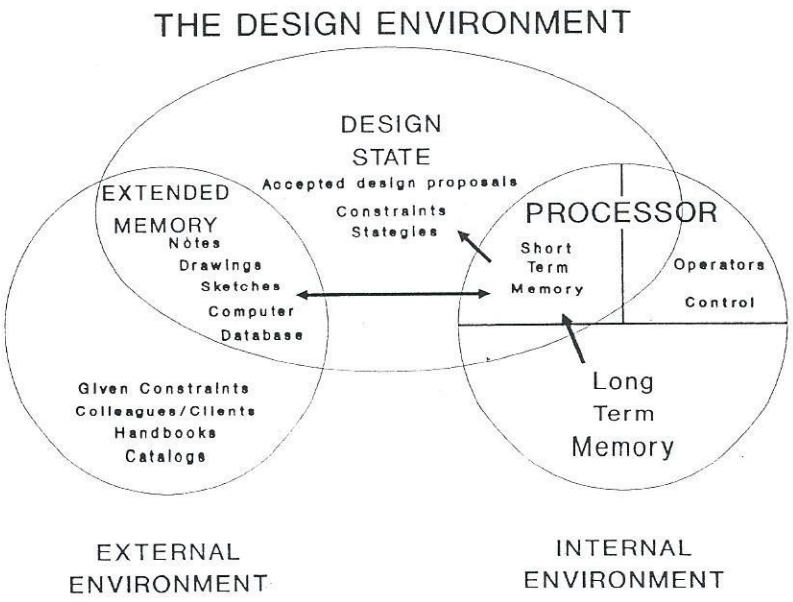

FIG. 2. Model of the mechanical designer as an information processing system.

The external environment consists of items used by subjects as an external memory, as well as manufacturers' catalogs, handbooks and the problem statement. These items and their use in problem solving have already been described in detail in other papers and will not be covered in this paper [18–20].

The internal environment contains the processor which consists of operators, a controller and the STM. The operators are the elementary information processes that the designer applies to the design state in order to change the state and thereby solve problems. The operators are discussed in detail in Section 3.2.

The controller commands the flow of information betweeen the STM and the other elements of the model. Section 3.3 describes how the operators are applied to make step-by-step changes to the design state. A description of all aspects of the controller, however, is beyond the scope of this paper. (A paper by Ullman *et al.* [22] presents a detailed explanation of the higher-level control and heuristics that the subjects used.)

Three types of information that are processed by the mechanical IPS model have been identified in the protocols: strategies, proposals and constraints. 'Strategies' are procedures for achieving a goal or simply executing the problem-solving effort. 'Proposals' are possible alternatives, form or function, for satisfying a goal—a proposed solution to the problem. As problem solving progresses, initial proposals are modified so as to specify a final solution. 'Constraints' are information that help limit or define a proposal. Constraints include the initial problem specifications and other requirements that the designer uses based on experience (i.e. standard hole sizes). Each time a new proposal is accepted as a viable alternative, new constraints are introduced to accommodate it. For example, using a certain hole size puts new limits on possible bolt sizes that can be used.

Making the distinction between strategies, proposals and constraints helps identify and keep track of information during problem solving. Design essentially starts with constraints and ends with proposals via strategies, so as to completely specify the form and function of a mechanism.

3.2 Operators

The finest information process identified in this research is an operator. Operators are the fundamental processes of design that are applied to the design state to produce a new state. Some 10 types of operators that fall under three general categories—generation, evaluation and decision—were identified and are discussed below.

(1) *Generate* operators introduce information into the STM so that it may be processed. There are two types of generate operators, 'select' and 'create'.

The select operator is applied (a) to information contained in the original problem statement, (b) to information which has previously been used in the problem-solving effort or (c) to information representative of common domain knowledge.

The create operator is used when information is generated but does not come from one of the above three sources, i.e. when information just seems to appear spontaneously, the create operator is used.

(2) *Evaluate* operators relate generated information in order to make a decision. Three evaluation type operators are identified: 'simulate', 'compare' and 'calculate'.

The simulate operator represents information at the proper level of abstraction in order to relate it. Since information can exist in any configuration, from an initial concept in the mind to an actual physical form, the role of the simulate operator is crucial. For example, one of the subjects had to relate a proposal (the height of a battery housing) to a constraint (that the battery he was holding in his hand would fit

into that housing). In order to relate these two chunks of information, he had to represent them at the same level of abstraction either by measuring the height of the battery and entering that value in his STM, making a physical model of the housing, or doing something in between, like making a sketch of both.

The compare operator determines compatibility: whether a proposal is satisfied by constraints, whether one proposal or strategy is better than anotner at satisfying a constraint, or whether two constraints are compatible.

The calculate operator infers new information by combining existing information, such as a sum, by adding two numbers. The information need not be mathematical—it could be qualitative. Rather than determine compatibility, this operator combines information uniquely.

(3) *Decision* operators are applied primarily to the results of the evaluate operators to describe the subject's subsequent actions. There are five types of decide operators: 'accept', 'reject', 'suspend', 'refine' and 'patch'.

The accept operator is applied if the evaluation result is satisfactory. The new information is then added to the solution state. The reject operator is applied if the evaluation is unsatisfactory. Just how a designer determines satisfaction is not clear from the protocol data, but it is obvious that some type of qualitative reasoning occurs [23]. Both of these operators can terminate a decision.

The suspend operator is used to terminate a decision without coming to a definite conclusion. The designer often applies the suspend operator because he or she is not prepared to make a decision, or to seek additional information.

The refine and patch operators are used to alter information. Refine is applied in order to make the proposal, constraint or strategy more specific and less abstract. Patch adds to or combines information without making it less abstract. These two operators are often followed by more generate and evaluate operators before being terminated by the accept, reject or suspend operator.

3.4 Local Methods

Mechanical designers solve ill-structured problems—those whose structure lacks definition [24]. Such problems require weak methods for solutions such as generate and test, heuristic search, and means end analysis [25]. In contrast, well-structured problems can be solved with more powerful methods solution. For example, many optimization problems can be represented as mathematical function, which are solvable by powerful methods such as linear programming.

In this research, the weak methods that describe how the subjects applied operators to solve individual problems are called 'local methods'. These methods were identified by stripping the specific descriptions from the proposals, constraints and strategies in order to reveal the unique sequences of operators that the subjects applied while making a decision. While at least a dozen weak methods have been identified [11, 26, 27], only four methods had significant use in the protocol data and are described here.

Generate and test (GAT) is the simplest method of problem solving with the exception of simply recalling a solution. GAT is the generation of solution to a problem and the testing of it to see if it is satisfactory. Knowledge of the desired solution is not needed with GAT, but only the knowledge of what is acceptable. GAT could be used to open a combination lock, if the combination were unknown, by randomly or systematically dialing all combinations—the method will work eventually.

Figure 3 shows the group of operators that are associated with the GAT method. The lines in the figures signify both flow of information and flow of control. The

sequence requires the generation of a proposal and a constraint. The proposal is compared with the constraint and either accepted or rejected.

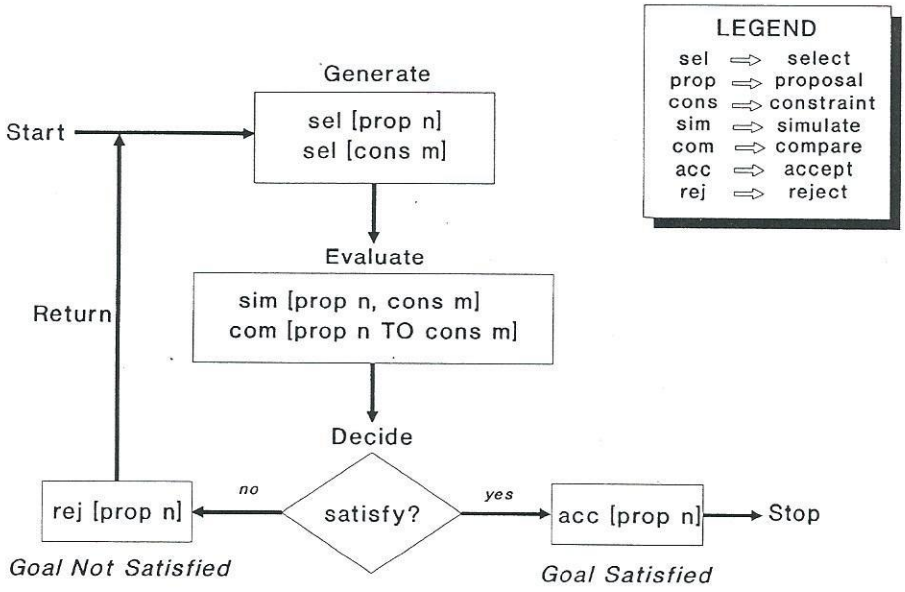

FIG. 3. Generate and test method.

Generate and improve (GAI) is a modification of the GAT method. This method is a type of hill-climbing used in many optimization procedures. In GAI, new positions are sought on an imaginary 'hill' and the best so far is kept. The GAI method is more general than hill-climbing in that new 'positions' on the hill can be searched from some earlier 'base' position or from the present position. Thus, the method has two features—the ability to generate new 'positions' and the distinction of maintaining the 'best so far' position. Just how the subject determines which proposal is best and when the best is good enough is not understood. GAI is a method of optimization, in a sense, because there is a search for a better solution, though not necessarily the best solution.

Figure 4 shows the GAI method identified in the data. Beginning from the 'base' of proposal n, new 'solutions' such as proposal $(n + i)$, are searched for on the hill with the refine or patch operators. The process continues by generating new constraints and proposals in order to find a good solution.

The third and most frequently used local method is another variation on the GAT method, known as *means-end-analysis* (MEA). MEA is actually a type of heuristic search—a method in which a search is conducted through an exponentially expanding solution space using heuristics to narrow the search. The MEA method is characterized by a comparison of the present state and the desired state followed by employing the proper operators needed (the means) to eliminate the difference.

The MEA method is shown in Fig. 5. The diagram begins with the generation of a proposal and a constraint. This information is evaluated via the compare operator, and then a patch or refine operator is used to achieve the desired state. Unlike GAT, this method requires knowledge of some 'ideal state' that the designer is striving towards.

A final method is *deductive thinking*. In this method, a decision is made via reasoning and inference. This method assumes that there is a system of logic that

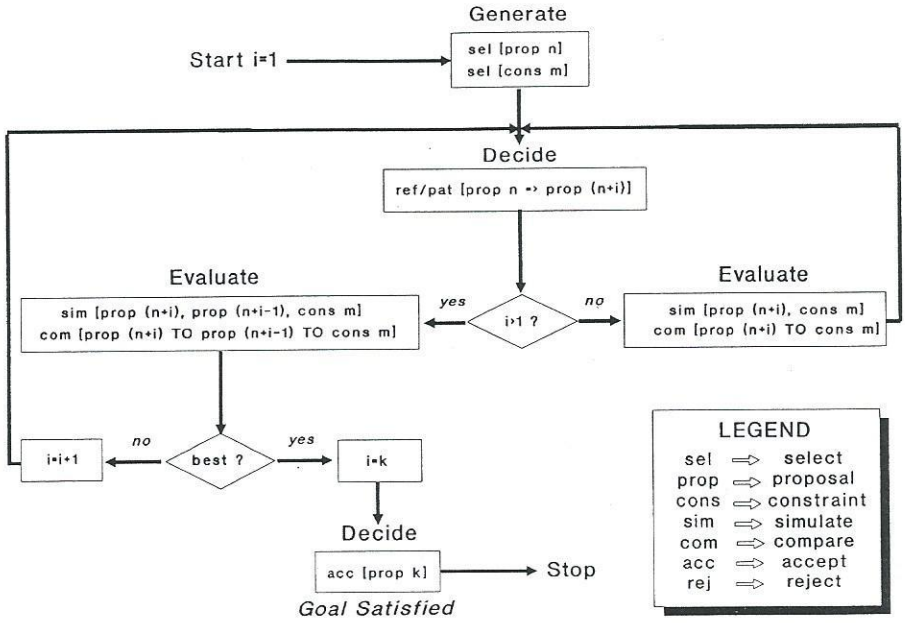

FIG. 4. Generate and improve method.

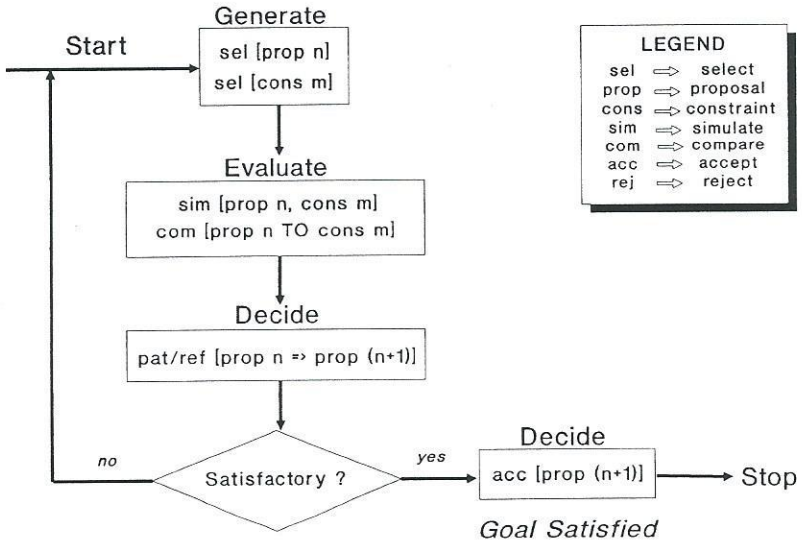

FIG. 5. Means end analysis method.

allows a person to deduce new information from existing information. Problem solving in this fashion seems to be in conflict with the search paradigm discussed so far in this research. Simon [12] addresses the conflict between search and reasoning in great detail, contending that problem solving can be viewed as both search and reasoning.

As a search, a space is assumed in which 'treasures' (states) are hidden. The problem solver searches the space with operators, moving from one state to another

until a treasure is uncovered. As reasoning, a system of logic is assumed whereby a person accumulates more and more information by inference until the problem is solved. Simon says, 'In no sense are the metaphors (search and reason) mutually exclusive; metaphors seldom are. The same problem-solving algorithm can be viewed, now as a search, now as reasoning'.

Figure 6 presents the deductive method used in the protocol data. In all cases, deductive thinking utilizes the calculate operator since, by definition, calculate is the inference of new information.

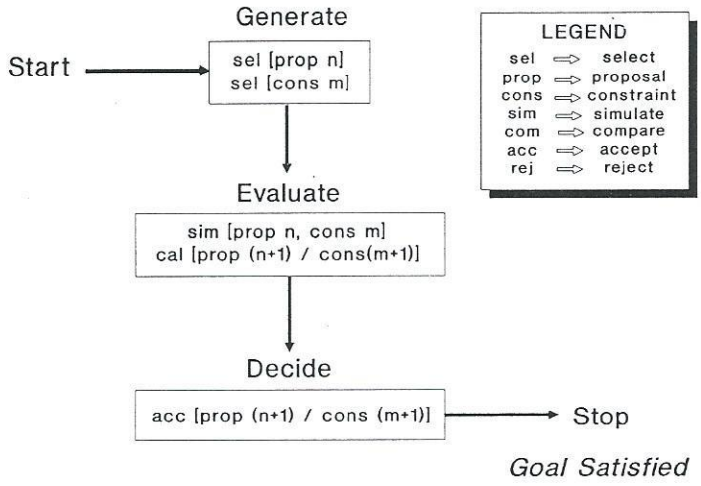

FIG. 6. Deductive thinking method.

The four local methods that are used to solve problems reveal some of the duties of the controller and put the operators into context with one another, demonstrating the ways in which they depend on each other during problem solving. (A description of other duties of the controller, beyond the local methods, can be found in Ullman *et al.* [22].)

4. Demonstrating the Processes

A model of human performance, detailing the operators and local methods used by mechanical engineers to solve problems in a design context, has been presented in Section 3. In order to demonstrate this model, sections of protocol data were randomly selected that were representative of problem solving at all stages of the design process. Some 19 sections were randomly selected and analyzed as described in Section 2.

An example of a section of data and its analysis for subject S1 is presented below. S1 is designing a solution to the problem that deals with electrical contacts. The section of protocol data shown below refers to the 14th decision out of 16 which were made to design the middle electrical contact, roughly 3 h 20 min into the experiment. The text is a transcription of the subject's verbalizations during this episode. Text in parentheses contains information of explanation to the reader and the asterisks in the text represent pauses by the subject of approximately 5 s. The reader must remember that the transcript is not sufficient on its own to understand performance, for many of the subject's actions can be found only in the videotape.

Decision 14

> and this nickle plate, just those two areas, and the rest of the material would be made out of, say copper alloy, which is a good electrical conductor. * A beryllium copper is commonly used for springs, but you don't really need a spring force on these (middle contacts), so it's not really a question, if we just use good copper for its conducting properties.

The operators that the subject used to achieve the goal of this decision—to specify a material for the contact—are coded and listed below. The time in parentheses is elapsed time into the experiment (hours:minutes:seconds); the difference in times gives the duration of the decision process. The operators are documented by the operator type followed by square brackets which contain the state of the information being operated on.

(3:19:37–3:20:24) specification—material for contact
select [constraint 15: nickel contacting surface]
select [proposal 20: material of contact (except for plated surface) is unknown]
select [constraint 16: contacts must have good conductance]
select [constraint 17: copper is a good conductor]
compare [proposal 20 TO constraint 16, 17]
refine [proposal 20 => proposal 21: material of contact is copper]
select [constraint 18: beryllium copper is commonly used for springs]
compare [proposal 21 TO constraint 18]
patch [proposal 21 => proposal 22: material of contact is beryllium copper]
select [constraint 19: contact does not need spring force]
compare [proposal 22 to constraint 19]
reject [proposal 22]
accept [proposal 21]

In this process, S1's goal is to specify a material for the electrical contact. The design state begins without a material stipulation and ends with a decision to use plain copper. The local method used in this process is GAI. Two proposals were developed: basic copper and beryllium copper. Then constraint 19 was generated and the better of the two was accepted.

The cognitive activity demonstrated in the randomly selected sections of data represent 203.9 min, or roughly 3.5 h, of problem solving in which 1120 operators are applied. With these operators, the subjects use 183 local methods of which 95% can be characterized as GAT (23%), GAI (8%), deductive thinking (33%), and MEA (36%). (Note that these percentages reflect the frequency of the methods and not the time spent applying them.) The methods required an average of 5.5 operators and roughly 1 min to complete. For a comparison, Akin [11] describes four methods used in architectural design: GAT, hill climbing, heuristic search (MEA) and induction. He found that such methods were used 94.5% of the time in his data (Fig. 7). A discussion comparing this work with Akin's work with respect to this figure can be found in Stauffer [7].

5. Conclusion

The IPS model provides a description of the problem-solving process of a mechanical designer. The operators describe the actions designers take to process information and the local methods describe how designers achieve the goals that make problem solving

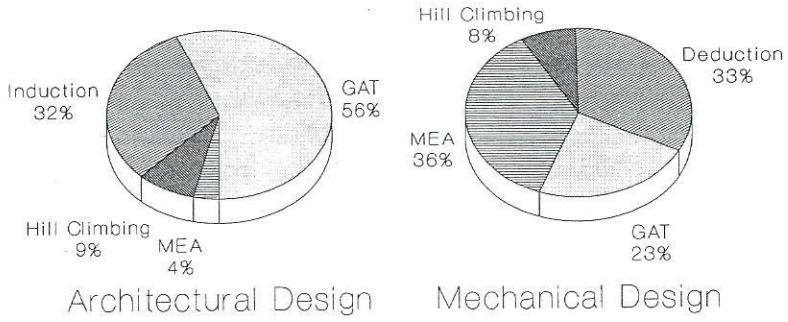

FIG. 7. Frequency of local search methods.

possible. The achievement of this research is significant. The operators identified herein describe the design activity at a more detailed level than do operators identified in past research on mechanical design (in Lewis's research, for example). Furthermore, the problem-solving strategies of mechanical designers have never been identified nor quantified in any past research—hence, the comparison with Akin's work in architectural design. This research, then, establishes a datum with which further work in mechanical design can be compared.

What is the implication of this research on design automation? Foremost is that it gives a detailed description of how a mechanical engineer solves problems in a design context. This performance is very dynamic yet somewhat repetitive at a local level. The same operators are used to achieve the same types of goals over and over, whether the task is at the conceptual or the detailed level. Alternative proposals are rarely investigated and if they are, then it is not in any great detail. Designers do not need to make deep or parallel searches to find a solution. Solutions are found by applying a wealth of knowledge to isolated problems rather than by developing a global procedure for the design task. The essence of mechanical design, then, seems to rely very little on problem-solving techniques but rather on a richness of knowledge. While the number of operators and local methods needed to describe problem-solving performances are relatively few, the knowledge of possible proposals and constraints that are used is endless.

The IPS model demonstrates how a designer functions, and intelligent CAD tools must be cognizant of these functions in order to interface naturally with mechanical designers. The designer's strengths should be exploited and weaknesses supported. Present design automation tools are not flexible enough. While research in the area will continue to provide more automation for the designer, compatibility with the user must also be addressed. In one sense, design automation is a 'human factors' problem. Just as the human body, its form and function, must be understood in order to design more useful hand tools or automobiles, human problem solving must be understood in order to develop more useful tools that interface with the mind. A computer tool that is developed with regard to such aspects will certainly have a competitive edge over tools that ignore human performance.

REFERENCES

[1] MARPLES, D.L. (1961) The decisions of engineering design, *IRE Transactions on Engineering Management*, EM-8, pp. 55–71.

[2] RAMSTROM, D. & RHENMAN, E. (1965) A method of describing the development

of an engineering project, *IRE Transactions on Engineering Management*, EM-12, pp. 79–86.

[3] MITROFF, I.I. (1967) A study of simulation-aided engineering design, *Doctoral Dissertation* (University of California, Berkeley, CA).

[4] LEWIS, W.P. (1981) The role of intelligence in the design of mechanical components, *Man-Machine Communication in CAD/CAM* (Amsterdam, North-Holland).

[5] WALDRON, K.J. & WALDRON, M.B. (1987) A retrospective study of a complex mechanical system design, *NSF Workshop on Design Theory and Methodology*, Oakland, CA, February (New York, ASME).

[6] HALES, C. (1987) Analysis of the engineering design process in an industrial context, *PhD Thesis* (University of Cambridge). (Republished by Gants Hill Publications, Eastleigh.)

[7] STAUFFER, L.A. (1989) The commonality of design in diverse domains, *Proceedings of the IME International Conference on Engineering Design*, Vol. 1 (Bury St Edmunds, Mechanical Engineering Publications), pp. 447–466.

[8] DIXON, J. (1991) Engineering design science: the state of education, *Mechanical Engineering*, Vol. 3 (New York, ASME).

[9] NEWELL, A. & SIMON, H.A. (1972) *Human Problem Solving* (Englewood Cliffs, NJ, Prentice-Hall).

[10] MILLER, G.A. (1956) The magical number seven, plus or minus two: some limits on our capacity for processing information, *Psychological Review*, 63, pp. 81–97.

[11] AKIN, O. (1986) *Psychology of Architectural Design* (London, Pion).

[12] SIMON, H.A. (1983) Search and reasoning in problem solving, *Artificial Intelligence*, 21, pp. 7–29.

[13] ERNST, G.W. & NEWELL, A. (1969) *GPS: a Case Study in Generality and Problem Solving* (New York, Academic Press).

[14] EASTMAN, C.M. (1968) Explorations of the cognitive processes in design, *PhD Thesis* (Carnegie-Mellon University, Pittsburgh, PA).

[15] AKIN, O. (1979) An exploration of the design process, *Design Methods and Theories*, 13 (3/4), pp. 115–119.

[16] ADELSON, B. & SOLOWAY, E. (1984) A cognitive model of software design, *Rep. No. 342* (Department of Computer Science, Yale University, MA).

[17] ERICSSON, K.A. & SIMON, H.A. (1980) Verbal reports as data, *Psychological Review*, 87 (3), pp. 215–251.

[18] STAUFFER, L.A. (1987) An empirical study on the process of mechanical design, *Ph.D. Thesis* (Oregon State University, Corvallis, OR).

[19] ULLMAN, D.G., STAUFFER, L.A. & DIETTERICH, T.G. (1987) Preliminary results on an experimental study of mechanical design, *NSF Workshop on Design Theory and Methodology*, Oakland, CA, February (New York, ASME).

[20] ULLMAN, D.G., STAUFFER, L.A. & DIETTERICH, T.G. (1987) Toward expert CAD, *ASME Computers in Mechanical Engineering*, 6 (3), pp. 21–29.

[21] KRIPPENDORFF, K. (1980) *Content Analysis* (Beverly Hills, CA, Sage Publishing), p. 121.

[22] ULLMAN, D.G., DIETTERICH, T.G. & STAUFFER, L.A. (1988) A model of the mechanical design process based on empirical data: a summary, *Proceedings of the AI in Engineering Conference*, Stanford University, CA, August.

[23] BOBROW, D.G. (1984) *Qualitative Reasoning About Physical Systems* (Amsterdam, Elsevier).

[24] SIMON, H.A. (1973) The structure of ill-structured problems, *Artificial Intelligence*, 4, pp. 181–201.

[25] POLIKAROV, A. (1985) Methodological problems and approaches in artificial intelligence, in: BIBEL, W. & PETKOFF, B. (Eds) *Artificial Intelligence: Methodology, Systems, Applications* (Amsterdam, Elsevier).

[26] NEWELL, A. (1969) Heuristic programming: ill-structured problems, in: ARNOESKY, J. (Ed.) *Progress in Operations Research* (New York, Wiley), pp. 363–414.

[27] AKIN, O. (1978) Artificial intelligence and pattern recognition in computer aided design, *Proceedings of the IFIP Working Conference in Computer Aided Design*, France, March 17–19 (Amsterdam, North-Holland).