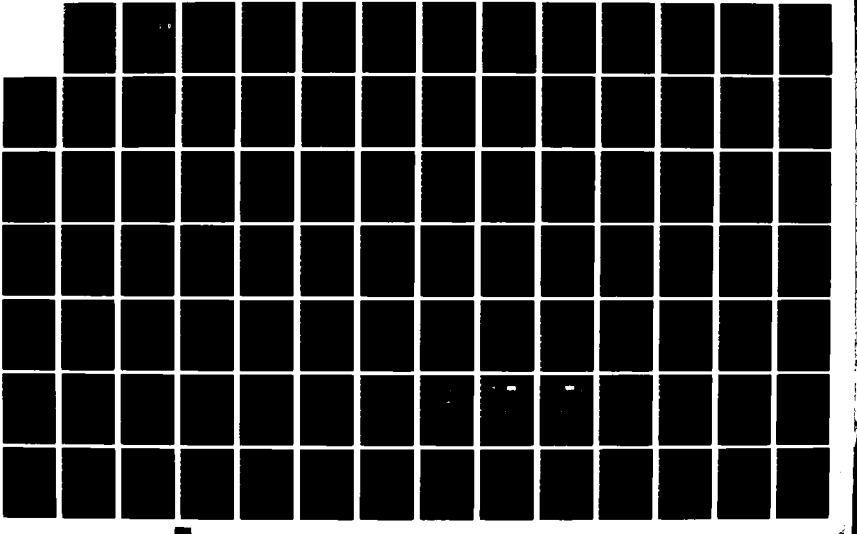
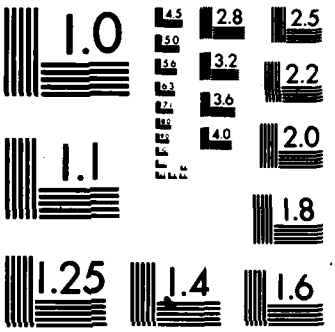
AD-A171 054 EXPERT SYSTEMS IN CIVIL ENGINEERING(U) FLORIDA UNIV 1/2
GAINESVILLE DEPT OF CIVIL ENGINEERING R A WALL 1986
N00228-85-G-3323

UNCLASSIFIED F/G 9/2 NL





EXPERT SYSTEMS IN CIVIL ENGINEERING

BY

RICHARD A. WALL JR.

N00228-85-G-3323

A REPORT PRESENTED TO THE GRADUATE COMMITTEE
OF THE DEPARTMENT OF CIVIL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF ENGINEERING

UNIVERSITY OF FLORIDA

SUMMER 1986

EXPERT SYSTEMS
IN CIVIL ENGINEERING

BY

RICHARD A. WALL JR.

A REPORT PRESENTED TO THE GRADUATE COMMITTEE
OF THE DEPARTMENT OF CIVIL ENGINEERING IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF ENGINEERING

UNIVERSITY OF FLORIDA

SUMMER 1986

DEDICATION

This report is dedicated to Marcia.
She postponed her education so I
could pursue mine.

TABLE OF CONTENTS

ABSTRACT

The fledgling field of Artificial Intelligence (AI) has found numerous applications in engineering and other disciplines. Most publicized among these are natural language recognition programs, systems that simulate cleverness (Eliza and the Rubix Cube solver, for example) and 'smart' front ends for mechanical mechanisms (robots). Unfortunately, these applications are too often seen as 'parlor tricks' or mere additions to existing technology. It has only been recently that the field has focused upon an application that will show these capabilities for the tip of the iceberg that they are. In fact, the new direction has the potential to affect the utility of computers in the same way the invention of the transistor impacted electronics. The goal of this new thrust is to 'clone' the experience, judgment and problem solving abilities of bonafide human experts into a computer program. Appropriately enough, these resulting programs are known as Expert Systems.

To understand the basic concept and structure of expert systems, it is necessary to first examine the background and fundamental theories of Artificial Intelligence. This is provided in Chapter 1, with emphasis on the methods and significance of problem representation and solution search strategies.

The utilization of these techniques is then examined, as the nominal component parts of an expert system are introduced. These are the user interface, the context, the

knowledge base and the inference engine. The architecture and function of each of these parts is dissected in turn, revealing the underlying structure of the system.

Leaving the theory behind, two operating prototype expert systems are then examined. The first, called TRALI, is a system designed to assist in the signal timing of isolated intersections. This system is studied due to its relative simplicity and functional transparency. The second system discussed acts as an expert scheduling assistant for a hypothetical construction project. Named the PLATFORM Model, this expert system demonstrates the current capabilities obtainable and points the direction of future endeavors in this area.

Costly both in terms of money and time, it is important to ensure that expert systems are developed and implemented only within those fields (domains) where their strengths are suited and their cost can be justified. Practical aspects of these decisions are discussed along with an examination of domains that are not appropriate for expert systems. Current research in the field of Civil Engineering is then discussed, followed by suggestions for appropriate applications in the area of Construction Management. Finally, an attempt is made to quantify the impact of widespread expert system use to the individual, the company and society as a whole.

CHAPTER ONE

INTRODUCTION TO THE HISTORY, METHODOLOGIES AND SIGNIFICANCE OF ARTIFICIAL INTELLIGENCE

1.1 Definition and Background

History will no doubt record that the coining of the phrase 'Artificial Intelligence' was indeed unfortunate. Far less threatening would be descriptions like 'Simulated Aptitude' or 'Synthetic Knowledge'; neither of which imparts to the layman the imagery of a machine dominated Orwellian society. However, for better or worse, the world will probably be stuck with this phrase from here on out.

The methods and goals of this field called Artificial Intelligence (AI) are not near as malevolent as one would think; unfortunately, neither are they as clear and succinct as one would desire. A classic definition, usually attributed to AI guru Patrick Henry Winston, defines AI to be "... the study of ideas that enable computers to be intelligent" (17,p.1). Were humanity to possess a viable set of criteria to define 'intelligence', then this definition may serve well. Unfortunately, such criteria do not exist. Consider the example of a child who learns to cry for its mother, yet cannot evaluate a simple Boolean expression. Is it intelligent? Likewise, what measure of intelligence can be conferred upon the computer that evaluates 4 million expressions per second but does not signal the operator when something goes obviously awry? Even beyond the question of bestowing the title of

'intelligent', there is the dispute of whether or not intelligence is absolute, or whether degrees of intelligence can be attached to different objects, actions or events.

Luckily, to operate within the field of AI, it is not a necessary prerequisite to make these weighty judgments. The act of being intelligent differs from the simulation of the act in that when attempting the latter, the researcher must first identify the principles that underlie the endeavor (17,p.3). In other words, one primary goal of the field is to understand the principles and mechanisms of intelligence. This knowledge can then be used to design and build computers that are more effective for a given application (17,p.2). It should be noted that these definitions avoid the struggle of defining intelligence, as was previously discussed. As Herbert Simon, another acknowledged expert in the field long ago pointed out, "It is not the intent (of researchers in the field) to engage in a barren lexicographic exercise, nor to bait those among us who are aroused to indignant emotion whenever terms from human psychology are used in reference to computers. We employ these anthropomorphic terms because we find them useful in defining our research goals ..." (11,p.224). In essence, the application of AI techniques and procedures confers an attempt at 'intelligence' upon a system; the fact that this intelligence has little or no connection to the 'intelligence' of philosophical doctrines is of no consequence. As the discipline matures, there will no doubt

evolve a standard by which intelligence will come to be measured. At that time, the performance of a system will be used to judge the validity of its place in the world of AI. Until then, however, the gauge will remain subjective.

1.2 Strategies and Methodologies

Although the concept of perceived intelligence dates to the 1800s and earlier, the late 1950s saw the beginning of what is currently called AI. The initial efforts were focused along the lines of a General Problem Solver (GPS). It was felt that all problems could ultimately be reduced to a point where one general solution strategy could be employed. While certain natural language understanding programs did enjoy limited success by using this approach, it was soon apparent that the larger the number of problem classes a program was required to handle, the less value its solution had on any single problem (16,p.3).

With this realization, the emphasis shifted to the development of methods and strategies that could be brought to bear on specific classes of problems. Two of the more important lines of research pursued toward this end were 1) the representation of the problem being addressed and 2) the search for one or more solutions within the state space of valid answers (16,p.4).

1.2.1 Problem Representation Strategies

Proper representation is critical, as the characteristics of a problem (or problem class) must be organized so as to fit the framework of a designated

solution strategy. A good description will make clear the important features of a problem, as well as reveal any underlying natural constraints inherent in the problem or problem class (17,p.24). Conversely, a poor description may render an otherwise easy problem unsolvable. Two important methods in this area are Description Matching (17,p.26) and Goal Reduction (17,p.33). The former allows selection of a solution strategy based primarily on a comparison of the attributes of the various strategies available. The strategy selected will usually be the one whose characteristics most closely match those of the problem. Goal Reduction, on the other hand, employs attainment of subgoals as a strategy to mold the problem characteristics to that of a solution strategy. It can be seen that both approaches strive to represent the problem in terms of the characteristics of predefined solution strategies, in one way or another.

1.2.2 Search Technique Strategies

Proper search techniques are also considered essential to problem solution. Utilization of the proper technique will allow quick and efficient identification of an answer. Alternately, the wrong search strategy may extinguish any hope of finding a solution due to control strategies that continually select the improper branches of a search tree for exploration. A classic testbed for search strategies is the 8-puzzle, as shown in FIGURE 1 on page 10. This puzzle is a square surface, containing 8 square tiles with the 9th

tile space vacant. Initially jumbled, the goal is to move the tiles, one at a time, so as to arrange them numerically around the periphery of the surface (13,p.32). The search tree for this problem is constructed so that the nodes at each successive level represent all the possible moves that can be made from the state as shown in the level above. The 'quality' of a move can be measured by a count of the number of tiles that are in the proper locations. As used in this context, a move's 'quality' is not absolute and can, in fact, be calculated by any number of appropriate algorithms. A better indicator would be to include a measure of the distance away from home for those tiles not in their proper locations. While this method would provide for a more succinct representation of the problem state, it also requires more time and effort to compute from move to move.

Fundamentally, all search techniques rely upon the existence of a 'quality' assigned to each state of the problem space. It is only by comparing successive 'qualities' that an algorithm can determine if it is converging on or diverging from a solution. Although simple in concept, this indicator has proven very difficult to implement in practice. For example, consider the positions occupied by chess pieces on a board. Given this information, a chess player has little trouble determining which side is in the better position. To date, however, no algorithm has been developed that can reduce the positions to a 'quality' number that describes who has what advantage.

This deficiency is not merely limited to chess, but occurs in any situation wherein the characteristics of the problem are even marginally dynamic.

However, for those problems whose states can be reduced to 'quality' numbers (or a reasonable facsimile thereof), there are three major categories of classic search techniques that can be employed in the quest for a solution. These three are informally known as any path, optimal path and gaming (17,p.88).

1.2.2.1 ANY_PATH_Search_Technique

The first, any path, contains strategies that are designed merely to find some solution to the problem (17,p.89). This is usually regardless of the 'quality' of the solution or the efficiency with which it was found. Techniques of this type include Depth-First, Breadth-First and Hill Climbing, to name but a few (17,p.88). These strategies are normally employed when either an unsophisticated solution is acceptable or when an initial solution is required for further refinement. In the case of the 8-puzzle example, as depicted in FIGURE 2 on page 11, this technique would find a path to any node at the (n+1) level that had a higher 'quality' number than the node from which the search began. However, the fact that the 'quality' number at the node adjacent to the node 'found' was twice as great would be of no consequence to this particular control strategy.

1.2.2.2 OPTIMAL PATH Search Technique

The techniques of the second class are designed to discover the optimum path. In most cases, the optimum path is defined as the shortest path, in terms of cost, to traverse nodes. Cost can then be defined as efficiency, expediency, link weightings or any of a myriad of characteristics that would be appropriate to a given problem class. Techniques of this type include the British Museum procedure, Branch and Bound and the A* method (17,p.88). These methods run the gambit from inefficient (in the case of the British Museum procedure that evaluates all possible solutions, and then ranks them accordingly (17,p.101)), to the highly organized and effective A* method that uses a fairly complicated control algorithm to determine its next move (17,p.113). In the case of the 8-puzzle example, these strategies would optimize the search by finding the solution to the puzzle with the least number of moves (i.e.- the greatest increase in 'quality' numbers between levels). This class of techniques is customarily employed when a solution will be implemented on a recurring basis and thereby will stand to gain continually from an optimum solution. This is contrasted by the one-time-only implementation, where the cost to determine the optimum solution is often greater than the savings that will result from its implementation. FIGURE 3, on page 12, depicts this

strategy on a generic search tree and demonstrates that the 'better' solution is achieved only after a much more intensive (and costly) search of every level.

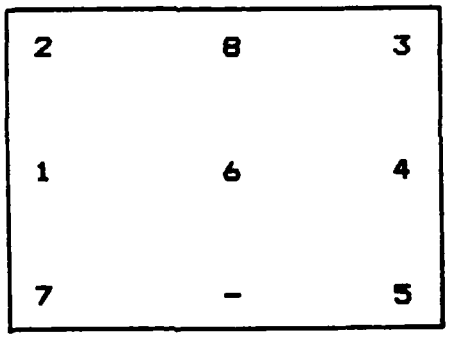
1.2.2.3 GAMING Search Technique

The third class of search techniques strives to optimize a solution in an adversarial environment. This situation is common in gaming theory, where, for each move that is made toward one's optimum position, an adversary makes a move that is away from one's optimum position (17,p.114). Examples of this predicament can be found in chess, checkers, war, etc.. Control strategies that deal with this category are the Minimax procedure, Alpha-Beta Pruning and Progressive Deepening (17,p.88). While the 8-puzzle is not germane to this class of problems, any game with two players can serve to illustrate the utility of these strategies. In this environment, every other move (i.e.- all the moves made by an opponent), are made for the purpose of decreasing one's advantage. This being the case, it is not enough to discover a path that has a high 'quality' number. The search must look beyond that level to determine the amount of 'damage' that can be done to one's position by the upcoming adversarial move. For example, two potential moves, as illustrated in FIGURE 4 on page 13, may yield gross increases in 'quality' of 3 and 6 respectively at the (n+1) level. Looking one level down to the (n+2) (opponent's move)

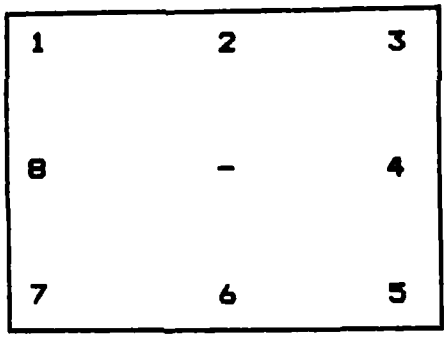
level, however, reveals that the opponent's potential moves have the capability to inflict damage of 1 and 7 respectively. Therefore, the net 'quality' of the moves, at the second level, are 2 (3 (at the (n+1) level) minus 1 (at the (n+2) level)) and -1 (6 (at the (n+1) level) minus 7 (at the (n+2) level)). From this perspective, it is obvious that the better move is the one yielding the 3 at the (n+1) level, since the potential for injury is far less than the move yielding the 'quality' value of 6.

1.3 Trends of Development

While interesting, the disjointed nature of the methods and techniques described above failed to bring about the long awaited revolution in artificial intelligence. By the late 1960s, the field contained a variety of sophisticated algorithms, all tuned to specific environments and problem classes. However, the line that separated high-powered algorithms from perceived intelligence had yet to be crossed. Before this could happen, new methodology had to be developed that could provide for a further limiting of the problem scope, and an infusion of knowledge about the problem to supplement, and if necessary replace the algorithmic solution strategies that were beginning to be used beyond their capabilities (16,p.4). The result of these changes became a new discipline called knowledge engineering and the product was a new line of computer programs called expert systems (16,p.5).



ARBITRARY INITIAL STATE



GOAL STATE

FIGURE 1.

8-PUZZLE configuration showing an arbitrary initial state (top) and the final goal state (bottom).

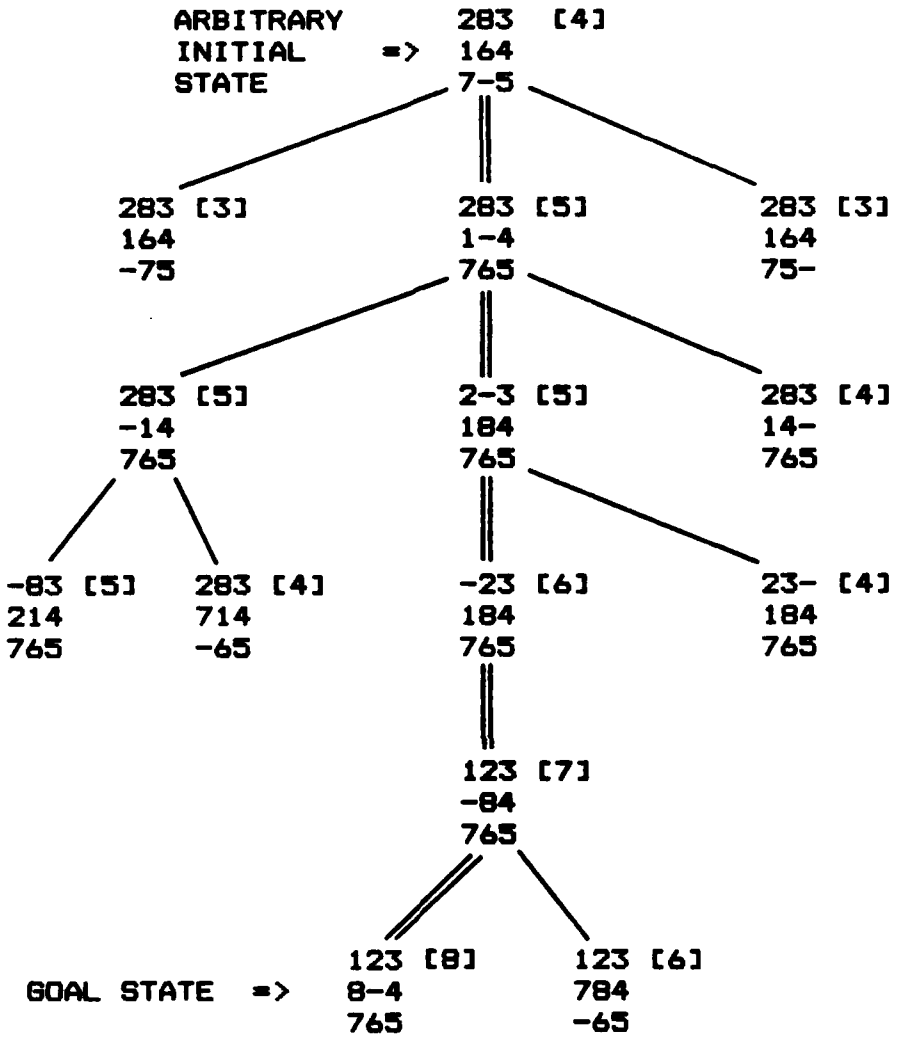


FIGURE 2.

ANY PATH control strategy for solution to the 8-puzzle. (Note: bracketed numbers ([]) indicate the number of tiles in the home position (i.e.- the 'quality' number of the state).)

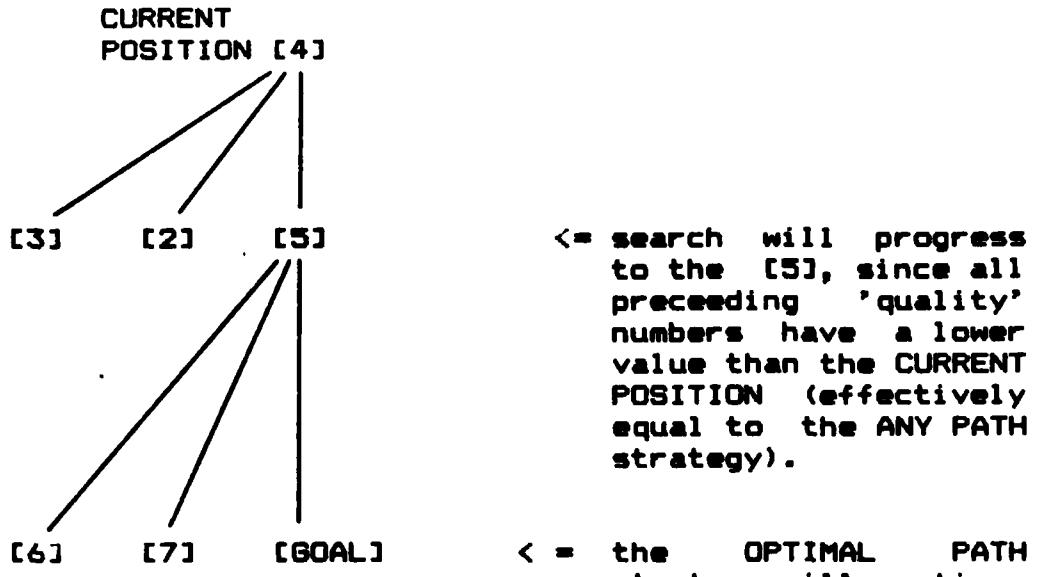


FIGURE 3.

OPTIMAL PATH control strategy demonstrating a costlier search approach, yet producing a more efficient solution path within the search tree.

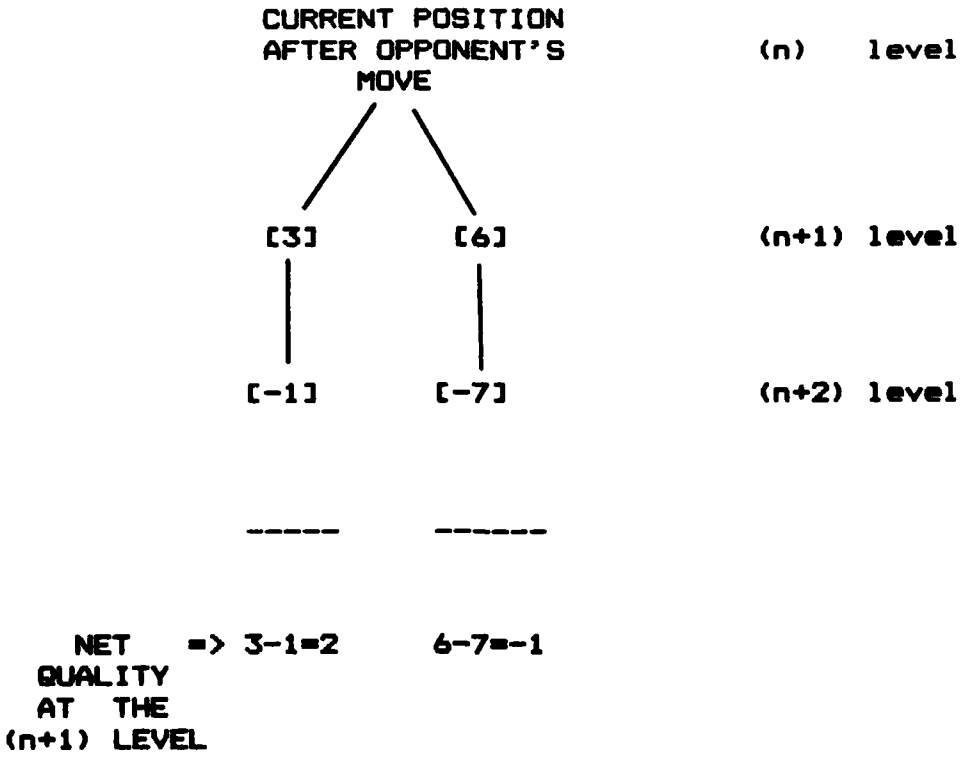


FIGURE 4.

GAMING control strategy where the 'quality' number at the (n+1) level is determined by evaluating an opponent's possible moves at the (n+2) level.

CHAPTER TWO

INTRODUCTION TO THE THEORY AND MECHANICS OF EXPERT SYSTEMS

2.1 Background

The choice of the phrase 'expert system' to describe the current level of AI technology is perhaps more appropriate than some alternatives. Unlike the imagery that 'Artificial Intelligence' conjures, 'expert system' does not seem near as wicked or insidious. Technology aside, the leaders in the field are certainly starting to comprehend the significance of semantics.

Expert Systems are not merely the aggregate of all AI methods and techniques so far developed. They transcend the current state-of-the-art by the introduction of methods and perspectives that are totally new to the field. From outward appearances, the major change is a focus of purpose; the goal now is not to imitate that intangible quantity called intelligence, but rather, to duplicate the performance of a human expert. (As such, it is recognized that the solutions will not always be correct). An added benefit to this stated goal is the introduction of a standard, by way of a human expert, against which any system can be measured. In other words, there now exists a purpose as well as a yardstick with which to gauge performance.

2.2 Definition and Perspective

This line of reasoning helps introduce the following definition for expert systems:

> An Expert System is a computer system that uses a representation of human expertise in a specialist domain in order to perform functions similar to those performed by a human expert. (2,p.1-1)

Contained within this definition are 3 clauses that help clarify the conceptual components of expert systems. The first addresses the fact that expert systems use "... a representation of human expertise ...". This mandate calls for the replacement of classic AI solution strategies with whatever is required to represent human expertise. At this time, the 'whatever' is hypothesized to be task-specific knowledge that is gained by the human expert, over the years, as his expertise matures (14,p.80-81). In other words, knowledge about the task and the domain of operation is seen to replace algorithms and the data they operate upon. The second, which speaks to the "... expertise in a specialist domain ...", has the purpose of conceding that General Problem Solver (GPS) type approaches are foreordained to failure. This translates to a need to restrict the scope of the problem classes that expert systems will face. The third and final clause involves the performance of "... functions similar to those performed by human experts". This passage defines both the goal and the required level of performance of expert systems.

2.3 Limitations

Obvious by its absence from the above list of concepts that define expert systems is any mention of the heuristics

of learning. While this is conceded to be an important aspect of expert performance, the state-of-the-art does not address this function of an expert system. The current emphasis centers on the ability to derive knowledge from rules. To reverse directions and derive rules from knowledge is a much more complicated process that has yet to find its way into the stage of conceptual development. This shortcoming, it is argued by some, may represent a fatal flaw in the basic fabric of expert systems, due primarily to the observation that, in many fields, knowledge is increasing exponentially as a function of time. The fear is that, in the absence of an ability to learn, an expert system's knowledge base may well be outdated by the time it is released to a production environment. While researchers are continuing to investigate the mechanics of expert system learning that will eventually alleviate the problem completely, there are currently a number of methods being used to minimize the impact of a system's inability to learn. The scheme most widely used is simply to restrict the implementation of expert systems to those fields where the basic knowledge is fairly stable and unchanging (i.e.- medical diagnosis, for example, where the diseases and symptoms remain the same, even as the treatments change). If implementation of an expert system is required in a dynamic field, then it is desired that the specific application be limited to an area of knowledge that is relatively static (i.e.- in the highly volatile field of

computer engineering, the Digital Equipment Corporation's expert system XCON merely configures equipment layout within the confines of a user's space). One scheme that is not considered a viable option is the 'updating' of a knowledge base with rules provided by another expert (1). It has been observed that even while two experts in a field may agree upon the final solution, their respective methods of attacking the problem may bear little or no resemblance to each other. From this, it is obvious that substituting partial methodology from one expert into the solution scheme of another is little better than playing automated Russian Roulette. For these reasons it is necessary, at least for the time being, to limit the scope of the knowledge contained within an expert system to a static 'snapshot' of the domain of operation.

2.4 Divergence from Classic Programming

Before discussing the elements that constitute an expert system, it is important to understand the differences that separate them from other computer programs. For a program to be a success in any area (i.e.- expert system, data base manager, 3 line BASIC program, etc.) it must meet certain minimum requirements. The following 3 criteria define this minimum level of performance (5,p.3):

> 1) the program must consider all possible combinations of input parameters, and be able to provide a viable output for all the potential permutations (i.e.- the algorithm must be COMPLETE).

2) the program must provide for one and only one output for each permutation of input parameters (i.e.- the solution must be UNIQUE).

3) the program must produce the correct solution for each permutation of the input parameters (i.e.- the solution must be CORRECT).

Since the task of classic programming is one of explicit representation, these requirements are relatively easy to meet only for programs that answer questions of the type "How big?" or "How many?". Systems that attempt to answer inquiries relating to "Which is best?" or "How do I proceed from here?" must resort to other strategies if the above requirements are to be fulfilled. This is due in part to the combinatorial explosion that would result if all cases were handled explicitly. (This assumes that the author had the foresight to include all possible cases, and thereby meet the requirement of COMPLETENESS ... which is highly unlikely for the type of problem under consideration).

Expert systems are designed to answer these types of questions by using knowledge, not algorithms, to fill in the blanks, direct the search and solve the problem (2,p.1-5). To accomplish this, knowledge is separated into three distinct areas: 1) the knowledge about the domain (which can include knowledge about objects, events and performance (13,p.144)) 2) the knowledge about the knowledge (usually referred to as meta-knowledge) and 3) knowledge about how to solve the problem. To put these various components into perspective, consider an expert system designed to schedule

the construction of a building foundation. The domain knowledge would consist of understanding the types of materials that are used to build a foundation, the requirements for site preparation and compaction as a function of different soil types and a comprehension of the various trade skills required to execute the activities of the foundation construction. In essence, all the information necessary to perform the mechanics of foundation construction is included in this category. The next category, knowledge about the domain knowledge, could include an awareness that recent weather conditions (i.e.- excessive rain, freezing temperatures, etc.) may alter the soil characteristics relative to those specified in the architect's report or the contract specification. Meta-knowledge of this type moderates domain knowledge. The last type of knowledge directs the utilization of the domain knowledge and the meta-knowledge to solve the problem; it is knowledge about the solution strategy. In the example, knowledge of this type could be knowing the nominal sequence of specific activities, interactions and constraints that are required to construct a foundation.

It may well be argued from the example that certain knowledge belongs in categories different from where it was placed. This may well be the case, as there are no succinct rules dictating the placement of particular knowledge into specific categories. As the field of expert systems now

stands, decisions like this are usually left to the judgment of the knowledge engineer and the domain expert (16,p.8-9).

2.5 Methods of KNOWLEDGE REPRESENTATION

Once placed within the proper category, however, the knowledge must still be represented in a fashion that allows it to be utilized by the expert system. In the absence of a viable format, the knowledge most germane to the problem may be essentially invisible to the component of the program that is formulating the solution strategy. Toward this end of usefully describing the knowledge, three methods of representation are currently being used within the field. These are 1) systems that represent their knowledge in a RULE type format 2) systems that rely on a SEMANTIC NETWORK to organize their knowledge and 3) systems that utilize FRAMES. (16,p.63).

2.5.1 RULE BASED Knowledge Representation

Rule based systems represent knowledge in an IF-THEN format (i.e.- IF <condition> THEN <action>). This method lends itself well to quantifying domain knowledge resulting from empirical association developed over the years (16,p.63). Using this approach, many different kinds of knowledge can be represented: situation/action, premise/conclusion or antecedent/consequent, to name but a few (2,p.74). Further, represented knowledge need not be just concrete fact; rules-of-thumb, heuristics and quantifiable intuition are all fair game for representation.

One benefit of this method is the simplicity of either adding or modifying rules. Unfortunately, this can also prove to be a liability, as it becomes very easy to introduce contradictions into the knowledge base (2,p.97,100). In the absence of sophisticated control strategy that will identify this problem, it is obvious that the results could be disastrous. Another question of utility lies in the explicitness inherently required for this type of representation; how is one rule for one action any better or more efficient than an algorithmic approach? To answer this, it is important to realize that the order in which the rules are executed is not predetermined, as is the case with an algorithm. The flexibility of the program allows the parameters of the problem and the knowledge of the domain to dictate the order in which the rules are invoked. In addition, the <action> clause can even execute an algorithm that will return a value to be acted upon by another rule or set of rules. Finally, the number of rules required can be directly related to the character of the domain, the scope of its definition and the complexity of the problem. Typically, a production system that has been in development for 2 to 4 years will possess a knowledge base of 500-1500 rules (2,p.149). By way of exception, XCON, an Expert System developed over the past 10 years by the Digital Equipment Corporation (DEC) to configure VAX computer systems, has, at last count, nearly 3500 rules in its knowledge base (1). For single steps through the

solution strategy, the discrete knowledge contained within each rule can be used to good effect. However, this same discreteness masks the overall comprehension of relationships within the knowledge base. This can be of crucial importance as the system is required to select its own solution strategy based on relationships within its knowledge of the problem. Despite all the problems and shortcomings discussed above, rule based systems continue to be the most widely used representation strategy (2,p.97).

2.5.2 SEMANTIC NETWORK Knowledge Representation

The second type of knowledge representation used widely in expert systems is the semantic network. This scheme employs a network structure with nodes that correspond to objects, events, concepts, etc. connected by links (called arcs) that describe the relationships between the nodes (16,p.70). In one sense, semantic networks lend themselves very well to the comprehension of global relationships that, as discussed above, was a very important shortcoming of rule based systems. For example, the nodes 'support structure' and 'concrete block' may be connected by an AKO arc, thereby indicating that a 'concrete block' is A Kind Of 'support structure'. Additionally, the nodes 'concrete block' and '8 inch CMU' may also be connected by an AKO arc, signifying that an '8 inch CMU' is A Kind Of 'concrete block'. These two arcs then establish an inheritance hierarchy within the network that allows the inference of an '8 inch CMU' to be A Kind Of 'support structure' (16,p.70-71). While

relationships are easy to follow within the system, the construction and upkeep of a semantic network can be quite arduous. Additionally, the fact that relationships are easy to interpret does not necessarily mean that they are also easy to use; attempting to identify cause and effect connections from a network of relationships can doom a system to the plate-of-spaghetti syndrome. It is due to this, as well as the fact that not all domains lend themselves to representation in this manner, that the utilization of semantic networks is on the decline.

2.5.3 FRAME BASED Knowledge Representation

The final representation scheme utilizes a vehicle called a frame, in an attempt to incorporate the best features of both rule-base and semantic network representation. The author of this concept, Marvin Minsky, describes his creation most succinctly:

> A frame is a data-structure for representing a sterotyped situation, like being in a certain kind of living room, or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed. (16,p.73)

Conceptually, a frame is an aggregate of nodes and arcs in a semantic network that are all concerned with the same object or event. For example, FIGURE 5a on page 37 shows a semantic network where the node 'construction project' is connected to the nodes 'labor', 'material', 'equipment' and 'subcontractors' by arcs of various relationships. Even as

this portrayal makes the relationships between the component parts quite evident, it is questionable if the knowledge is represented in a fashion that allows it to be used to solve a problem. How would a computer program use these relationships to execute a project or to determine if all required elements of the project were even available? While it would no doubt be possible to trace back all the arcs and relationships, the process would be unduly difficult and very inefficient. As a solution to this problem of cumbersome representation, consider the frame shown in FIGURE 5b, on page 37, that corresponds to the example network. This frame contains all the information of the semantic network, but in a form that allows its utilization. The attributes of labor, material, equipment and subcontractors that are associated with the frame 'construction project' are known as 'slots'. Into these slots go 'values', that are the domain specific knowledge about the problem at hand. For example, from this representation the system knows that a project requires labor, material, equipment and subcontractors. If provided a list of materials, and asked to execute a project, the system could easily ascertain that it needed labor, equipment and subcontractors. A more complete frame may also include slots that provide default knowledge concerning a project; typical duration is 60 days unless union labor is used, in which case it will be 90 days. Represented like this, the structure of the frame itself contains knowledge

about the solution strategy while the slots represent the need for particular domain-specific knowledge. The variables that will fill the slots contain this domain-specific knowledge.

The benefits of this representation are many-fold. No longer need the program come to a grinding halt when all required input is not available or completely accurate (5,p.3). Building on the preceding example, the task of scheduling two consecutive projects would normally require the input of the first project's start time and duration. If the duration was not provided, the frame described above would find the default to be either 60 or 90 days, whichever was appropriate. Similarly, this organization allows the system to communicate its strategy to the user, if so directed (5,p.3). In the scenario above, assume that the system had not been provided with a duration for the first project. To solve the problem of scheduling the second project, the system may request additional information from the user, resulting in the dialogue shown below (CAPITALS denote user):

```
What will be the duration of the first project?
I DO NOT KNOW.
OK, how will the labor force be procured?
WHY DO YOU WANT TO KNOW?
To determine if union labor will be utilized.
WHY DO YOU CARE IF UNION LABOR IS USED?
If union labor is used, the first project will probably
last 90 days.  If not, then probably 60 days.
WHY DO YOU CARE HOW LONG THE FIRST PROJECT LASTS?
So I know when it will be completed.
WHY DO YOU NEED TO KNOW WHEN IT IS COMPLETED?
So I can schedule the start of the second project.
WHY DO YOU NEED TO SCHEDULE THE SECOND PROJECT?
Because that is the problem to be solved. (!)
```

This exchange demonstrates that the system itself can control the problem solving strategy that will be brought to bear, based upon the specific information (or lack of it) that is at its disposal (5,p.3). Further, it is obvious that not only is the problem of incomplete information circumvented, but the user also has the option to follow the solution strategy that the system is pursuing.

The frame approach is not the only method whereby incomplete input circumvention, strategy explanation and strategy derivation can be achieved. To be sure, these are goals that all expert systems attempt to reach in one fashion or another. The example of frames has been provided here because this approach has yielded the best results in these areas and it is the method that appears the most promising at this time for further research and development.

2.6 Component Parts of an Expert System

Regardless of the knowledge representation chosen, all expert systems consist of two primary parts; the knowledge base and the inference engine (4,p.1-10). Quite simply, the knowledge base, as already described, contains the domain-specific knowledge and the inference engine embraces the control strategy that determines how the knowledge will be used to solve the problem. For example, in a rule based system, the introduction of a new piece of knowledge (either by user input or by system derivation) may well cause the conditional clauses of a number of rules to be 'true'.

Employing its control strategy, it is then up to the inference engine to decide which rule to fire (i.e.- evaluate) (4,p.49). This selection is very important, as the action of the firing will cause a new piece of knowledge to be added to that which is already known about the problem. This, in turn, will cause other rules to prime, and the whole scenario will be acted out time and time again.

2.6.1 Support Requirements and Component Functions

Depending upon the sophistication of the expert system, a number of other 'support' features may also be present. These can include the user interface, the knowledge acquisition module, the context and the explanation module (10,p.53).

The user interface merely provides a friendly medium for man-machine interaction. When adding knowledge or altering the rule-base (using a knowledge editor), this module insulates the user from the requirement to enter syntactically correct information. When used in reverse, this friendly interface allows the system to present information in an understandable and usable format (i.e.- English responses and/or graphics as appropriate).

The knowledge acquisition module allows for the translation of the domain expert's knowledge into the strict format of the system's knowledge base. The amount of effort and time required to develop and debug an expert system is directly proportional to the sophistication of this module.

The context, while not truly a support function, is a formal repository for all information concerning the current problem. Dynamic in nature, information is constantly added or deleted as the system progresses toward a solution.

The final component, the explanation module, allows the user to query the system for an explanation as to its reasoning and strategy. Inquiries can include not only why a particular piece of information was required or how a certain fact was deduced, but can also ask why certain knowledge was disregarded. This feature is very important when both debugging a system and using it in a production environment (4,p.34).

2.6.2 INFERENCE ENGINE Strategies and Operation

The control strategies contained within the inference engine dictate which 'operator' the system will invoke to continue its search for a solution (i.e.- which competing rule to fire, when to query the user for more information, etc.) (10,p.53). The idiosyncrasies of inference engine control strategies varies significantly, each one sensitive to particular situations germane to a specific problem class. Some of those that have been implemented are briefly discussed below (10,p.54-55):

1) Means-End Analysis: the difference between the current state and the goal state is used to select an operator that has the best chance of decreasing the difference.

2) Problem Reduction: the current state is first broken down into smaller problems. An appropriate operator is then selected for each of the component parts.

3) Backtracking: this strategy retains a list of all decision points and dependencies so that an unsolvable solution path can quickly be discarded.

4) Plan-Generate-Test: similar to the British Museum Method of tree search, wherein most (or all) possible solution states are first generated, then tested until one is found that satisfies the goal state.

5) Hierarchical Planning and Least Commitment Principle: the problem is first represented as a series of dependencies, each with intermediate goal states. Operators are invoked to handle intermediate goals based on inverse dependency, with the goal being to defer decisions on highly dependent states as long as possible.

6) Constraint Handling: conceptually, this strategy attempts to determine a solution by identifying all the solution states that do not satisfy the goal state.

7) Agenda Control: each intermediate state of a problem is first assigned a priority rating. The strategy then consists of invoking operators to deal with intermediate goals based on their relative priority.

While each of the above control strategies has been used with some success, most expert systems currently under development use two other approaches either exclusively or together. These are forward reasoning (chaining) and backward reasoning (chaining) (16,p.66).

2.6.2.1 FORWARD CHAINING

The strategy of forward chaining requires the evaluation of all rules (in a rule based system, for example) whose conditional clauses are true.

Essentially, this method strives to derive all the knowledge it can, whether or not a particular derivation brings the problem any closer to solution (2,p.76-78). A system operating under this control would query the user for additional information only after it had derived all that it could, based upon the knowledge originally provided and the intermediate derivations it made to supply more knowledge to itself. As can be seen, this approach is very inefficient, in that many facts are derived that do not apply to the problem at hand (2,p.81).

2.6.2.2 BACKWARD CHAINING

Backward chaining, on the other hand, begins with a premise (theory) that the system then tries to prove by rule evaluation (13,p.198). For example, consider an expert system designed to schedule activities for a project where the current theory is 'completion delayed'. (The derivation of this theory may well be in response to a user questioning the possible scenarios that could make the project run over its estimated completion time). To arrive at 'completion delayed' for a conclusion, the system first interrogates the rule base for rules that have, as their action clause, 'completion delayed'. One possible rule may be:

IF (start delayed) THEN (completion delayed)

At this point, the system defines a subgoal of 'start delayed' and reinterrogates the rule base to see if it can prove this new subgoal. One possible rule it may find could be:

```
IF   (labor unavailable)     or
     (material unavailable)
THEN (start delayed)
```

Two new subsubgoals are defined, and the system continues its recursive process. If the program can somewhere obtain the fact that either labor or material is unavailable, then the initial theory of 'completion delayed' becomes its conclusion. However, knowledge of an ontime start with available material will invalidate the initial theory and cause the system to generate a new working hypothesis. Backward chaining is inherently more efficient than forward chaining, because all the facts derived have a direct bearing on the problem at hand (2,p.82), and no effort is wasted in deriving useless information. Typical domains where backward chaining is effective are in medical diagnosis (14,p.184) and anywhere that a small amount of 'front end' information can suggest a possible conclusion. Domains where backward chaining cannot be supported are those where no theories can be formulated ahead of time. These can include on-line monitoring and process control, to name but a few.

2.7 Implementation Languages

The mechanics required to implement these new methodologies have required the introduction of computer languages that offer greater flexibility in data representation and program control. Toward this end, the computer language of choice for programs dealing with artificial intelligence is LISP (LISt Processor). This is due to the ease of representation afforded by the list environment and the ability to manipulate the component parts of the list. Due to its recent popularity, a number of variations are now available. These include IQLISP, INTERLISP, INTERLISP-D and FRANZLISP (5,p.12). A recent entry into this list of implementation languages is C. Its strong point is the ability to migrate to different hardware environments essentially intact. This gives the program designer the ability to build the system on a machine different from the one on which the program will be implemented.

Using the implementation languages listed above, a number of expert system 'tools' have been written that provide the expert system designer with a foundation of capabilities. Divided into three categories, these tools permit the designer to trade-off flexibility for ease of implementation (10,p.56).

2.7.1 General Purpose Programming Languages

At one end of the spectrum are the General Purpose Programming Languages of LISP and PROLOGUE. While expert systems can indeed be implemented directly within these languages, no support structure for any of the component parts of an expert system exists inherent to the language. This requires the designer to build the entire system from scratch. While this requires a great deal of time and effort, it is also the environment in which the designer can obtain the most flexibility for the system.

2.7.2 General Purpose Representation Languages

One step of capability up from the General Purpose Programming Languages are the General Purpose Representation Languages. These languages, usually written in a LISP dialect, have been developed specifically for expert systems applications. Still quite flexible, they do not limit the designer to a particular control strategy or knowledge representation scheme. Examples include SRL, RLL and AGE (all from Stanford University), KEE (Intelli-Gentics Incorporated), OPS5 (Carnegie-Mellon University), ROSIE (Rand Corporation) and LOOPS (Xerox PARC) (5,p.21).

2.7.3 Expert Building Systems

At a level atop the tools previously described are programs called Domain Independent Expert System Frameworks or Expert Building Systems for short. These systems provide the complete framework of an expert system in terms of the

knowledge editor, knowledge base and inference engine (4,p.92-97). Examples of these include EMYCIN (Empty or Essential MYCIN, from Stanford University), KAS (SRI International), HEARSAYIII (USC-ISI), EXPERT (Rutgers University) and KMS (University of Maryland) (10,p.56). The benefit of these systems is obvious, in that the designer need supply only the knowledge about the domain of interest. However, if the knowledge representation scheme and inference strategies, that are essentially 'hard-wired' into the system do not lend themselves to that particular domain, then the headstart provided by the Expert Building System will soon become a glaring liability. It is therefore crucial that the knowledge engineer be conversant in not only the domain to be modeled, but also in the capabilities of the software available to assist in the endeavor.

2.8 Domains of Application

As capable and effective as expert systems are, and will come to be, it is important to understand that their application is not universal. Just as every housewife's recipe box should not be fed into the home computer, so should the implementation of an expert system be limited to domains where it can function effectively. To this end, there are six classic criteria that a domain should meet before an expert system should be considered (2,p.26):

> 1) genuine experts must exist. (this effectively nullifies the stock market and astrology from consideration).

2) the experts must generally agree about the choice of an acceptable solution.

3) the experts must be able to articulate and explain their problem solving methodology.

4) the problems of the domain must require cognitive not physical skills

5) the tasks cannot be too difficult (i.e.- beyond the comprehension of an expert in the domain).

6) the problem should not require common sense or general world knowledge.

Once a candidate domain has proven receptive, it is still necessary to justify the tremendous effort and cost of constructing and implementing an expert system. Considerations that can provide this justification include areas where the task solution has a high payoff, areas where human experts are unavailable in the quantity required (i.e.- not enough medical doctors to service each small farming community) or unable to do the job (i.e.- in a calculation intensive environment), areas where significant expertise is being lost due to changes in employment or death or domains that possess an unfriendly or hostile environment (i.e.- inside the containment vessel at a nuclear power plant or deep water salvage or construction) (2,p.27).

2.9 Applications Case Study of PROSPECTOR

The decision to implement an expert system in a given domain has often produced results in excess of the system itself. The example of the PROSPECTOR expert system is a good case in point. Developed between 1974 and 1983 at the

Stanford Research Institute, PROSPECTOR is a rule based system that directs drilling and mining operations in search of different types of ore and mineral deposits (16,p.49-50). As with all expert systems, PROSPECTOR began with intensive dialogue between the knowledge engineer and the domain expert, in an attempt to isolate not only the knowledge used by the expert, but also the problem solving strategies normally employed. Once identified, this information was used to construct the knowledge base and the control strategy. On the first attempt to solve a problem, however, PROSPECTOR failed miserably. It was only after this failure that the knowledge engineers and the domain experts began to realize that the actual procedures used by the experts to solve problems were not congruent with those procedures believed to be in use. After much additional work, the final result was not only a working expert system, but also a more lucid understanding of the true mechanics of the domain that is now being incorporated into college texts, etc. as a replacement for the methods that people (including the experts) previously believed were correct (1).

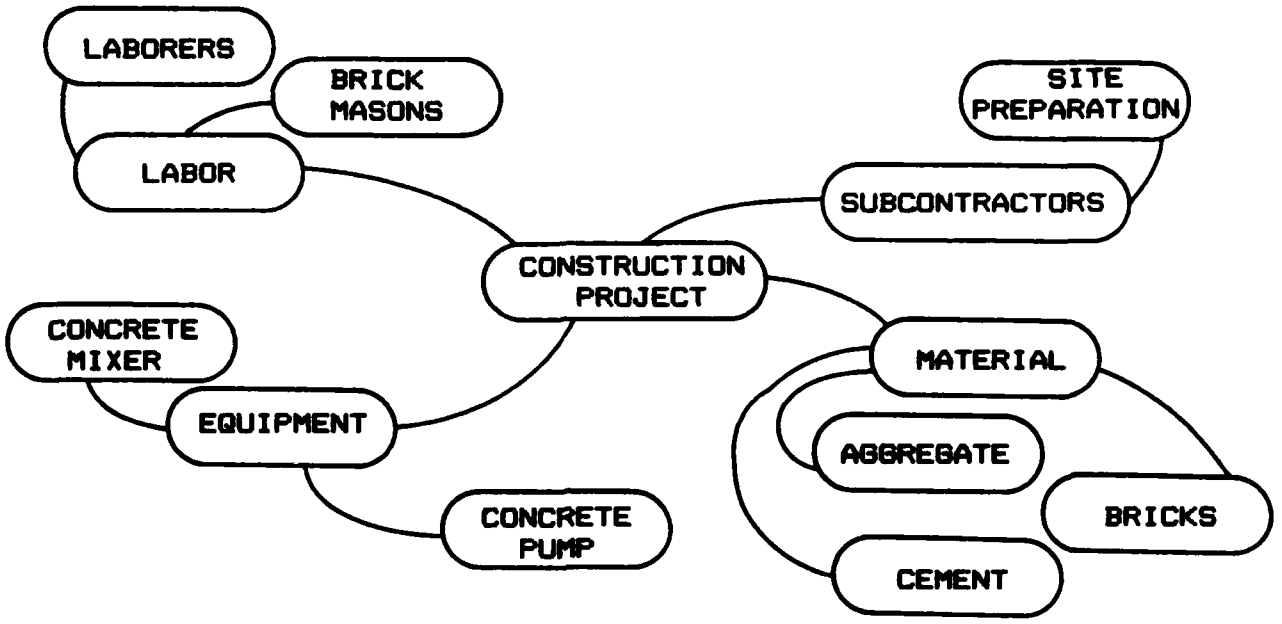


FIGURE 5a.

SEMANTIC NETWORK representing the four essential elements of a nominal construction project with representative values.

CONSTRUCTION PROJECT

```
REQUIRED ELEMENTS
   labor: trade1, trade2, trade3, ...
   material: material1, material2, ...
   equipment: equipment1, equipment2, ...
   subcontractors: sc1, sc2, ...
   duration: duration OR default (labor=nonunion): 60
                                 (labor=union)   : 90
```

FIGURE 5b.

FRAME representation for the semantic network shown in FIGURE 5a where 'CONSTRUCTION PROJECT' identifies the the FRAME, 'REQUIRED ELEMENTS' identifies the SLOTS and 'trade1, material1, etc.' identifies the VALUES.

CHAPTER THREE

CASE STUDY OF THE EXPERT SYSTEM TRALI

3.1 Introduction to the Expert System TRALI

The relative newness of expert systems, coupled with their inherently long development time has yielded a situation where no production systems exist in the field of civil engineering (10,p.57). This is not unusual, however, as the dozens of experimental systems developed across many engineering disciplines over the past ten years have produced a scant three to four true production systems that are actively engaged in field operations (2,p.149). To keep this in perspective, it is important to remember that the Concorde was not making trans-Atlantic crossings a mere ten years after the Kitty Hawk experiments.

Even though no production systems currently exist in the field, there are many small prototypes being developed for the purpose of evaluating new techniques and validating domains of implementation (7,p.294). One such prototype has been built by the Civil Engineering Department at Carnegie-Mellon University. Named TRALI, it is an expert system in traffic engineering designed to tackle the problem of isolated intersection signal timing by using a hybrid method of solution that encompasses both AI techniques and algorithm evaluation (18,p.1-2). Use of this composite structure is gaining in popularity because of the large number of domains where the union of number-crunching

algorithms and knowledge about the domain must work in harmony. For example, the entire gambit of potential civil engineering applications from cradle (design systems), through construction (project management systems) and service life (maintenance systems) will rely heavily upon expert systems that possess both of these capabilities. The architecture of TRALI demonstrates the flexibility that this combination can provide and points a direction that future production level expert systems may well travel. It is for this reason, as well as the fact that TRALI is a working prototype that it is included in this discussion.

3.2 Domain Background

The function of intersection timing is to allow all traffic movements (through and left turns) to transit an intersection in a timely manner and with minimal delay. The qualification of 'isolated' limits the intersections under study to those that are not part of an arterial network. This is important, as arterial intersections must be coordinated so as to provide for traffic progression along the route. By limiting the intersections in this way, the scope of the problem presented to the expert system has merely been simplified. Variables within the environment consist of the volumes for all the through and left turning movements, the geometry of the intersection (i.e.- the legitimate paths the movements are allowed to take) and the presence of additional required phases (i.e.-walk, don't walk, all red for intersection clearance, etc.). Parameters

that control an intersection in this regard are the cycle length and the phase distribution; the cycle length is the period of time required for all phases to be serviced and the phase distribution is the allocation of parts of the cycle to each phase (9,p.2-1 to 2-4). This is a classic problem in the field of traffic engineering, and has been the focus of many simulation and algorithmic based computer programs. While enjoying a fair amount of success, these programs have suffered from the inability to deal with situations that are not explicitly addressed in the program, as was discussed in the previous chapter. For example, an inherent shortcoming of existing programs is their inability to accommodate an intersection with more than 4 perpendicular legs (plus left turns) or any that have unusual geometry or requirements. FIGURE 6a on page 53 depicts the geometry of a conventional intersection that current software is capable of dealing with. One of the primary goals of the TRALI endeavor was to correct this deficiency by providing traffic engineers with a tool to handle intersections of unusual geometry (18,p.1). FIGURE 6b on page 53 portrays a representative intersection of this type.

The input required by TRALI is conceptually similar to that of currently used simulation software that employs algorithmic based solution strategies (the Signal Operations and Analysis Package (SOAP),for example). The exception is that the intersection is not assumed to be of a given

geometry. Rather, the various flows are characterized by an angle-in and an angle-out (18,p.3). This information is then used to abstractly describe the intersection to the component parts of the program that deal with flow conflicts, phase determination and other areas where the geometry is critical.

3.3 Solution Strategy Format

While not depending upon a preprogrammed solution strategy, the program does follow a general line of reasoning as it solves the intersection timing problem. The five main tasks nominally accomplished are 1) conflict determination 2) proposal of a phase distribution 3) determination of the optimum cycle and period lengths 4) calculation of figures of merit (measures of effectiveness that quantify the efficiency of the proposed design in terms of vehicle delay, etc.) and 5) modification to data and results at the user's discretion (18,p.3-4).

3.3.1 Conflict Determination

The first of these tasks, conflict determination, uses the information about the flow angles to identify flows that conflict and the degree to which they interfere with each other. For example, right angle flows present an obvious problem wherein the only solution is most certainly the creation of a separate phase for each flow. On the other hand, two flows whose angle-in values are fairly close may have the potential to be serviced by the same phase. Rules from the knowledge base are used to determine these

conflicts and structure the intersection description on the context, which is the short-term, working memory of the system.

3.3.2 Phase Distribution

The program then uses this information in the next step to preliminarily assign phases to the flows. It is at this point that the program selects the 'parent' flows, as those that absolutely require their own phase. Commensurate with this, TRALI attaches children (flows) to parents that exhibit similar characteristics (or weak conflicts).

3.3.3 Calculation of Cycle and Phase Lengths

The third step involves invoking certain algorithms to calculate the optimum cycle length and phase distributions for the preliminary phases determined in the previous step. While the evaluation of the algorithms involve no strategic control, the results may certainly be used by a control strategy rule to add a constraint or new piece of knowledge, and redirect the program back to a prior step for reevaluation.

3.3.4 Calculation of Solution Effectiveness

The next step involves the calculation of figures of merit or measures of effectiveness (MOEs). As with the procedures of step three, this is an algorithmic process that returns values for the average delay per lane, the average queue length and the total delay per cycle. Also like step three, values out-of-bound may trigger a control strategy that assigns further constraints or alters the

knowledge in the context, and then redirects the program to repeat steps one or two.

3.3.5 Solution Presentation and Input Modification

The last step entails the presentation of the solution to the user, and the commencement of an interactive dialogue should the user wish to query the system on how it arrived at the solution or why it chose to invoke a particular rule in the knowledge base over another. Additionally, this step allows the user to enter new constraints or to modify the knowledge base in preparation for the next evaluation. Zozaya-Gorostiza and Hendrickson (18,p.4) allude to the importance of this for sensitivity analysis (i.e.- modifying volumes, constraining phases, etc.). While this is necessary when using an experimental system such as TRALI, future production systems that are designed to optimize parameters of a numerical nature should most certainly include an indication of sensitivity to such parameters as a normal compliment to its output.

3.3.6 Advantages

In describing the rationale behind constructing TRALI, Zozaya-Gorostiza and Hendrickson develop the argument that a major restriction of current intersection analysis programs (SOAP, for one), is that the designer is required to pre-program all possible combinations of situations explicitly into the program (18,p.2). Essentially, the control strategies for all conceivable solutions must be considered and addressed before the program has a chance of success in

a production environment. This places an impossible requirement upon the designer and the program and all but ensures that the necessary condition of COMPLETENESS (as described in the preceding chapter) can never be met. In other words, the applicability of the program is reduced to only those cases foreseen by the designer. To circumvent this problem, TRALI is provided with knowledge about how to solve problems (herein called process knowledge (18,p.7)). This, in conjunction with domain knowledge and the appropriate number-crunching algorithms, allow the program to develop its own solution strategy to meet a particular situation. The representation for this process knowledge is contained in a rule base (IF-THEN format). Likewise, the domain knowledge and the meta knowledge are also represented in this format. In all, 237 rules comprise the knowledge base from which TRALI can draw (18,p.6).

3.4 System Components and Functions

Functionally, the program is broken down into four main components. These are the user interface, the context, the knowledge base and the inference engine.

3.4.1 User Interface

In TRALI, the user interface incorporates the 'friendliness' of the system. By default, both the explanation and the knowledge acquisition modules are also considered to be incorporated. However, the primitive level on which these last two modules operate require no amplification of their functions. The system 'friendliness'

consists of a menu-driven input format with response error checking (i.e.-the program will not accept a volume for a nonexistent flow). Additionally, the interface allows the user access to the context for the purpose of viewing and altering information contained therein. Since the context knowledge is nonvolatile between runs, the user can modify nearly any knowledge (user supplied, system derived or calculated) before attempting another solution (18,p.4).

3.4.2 Context

The context provides the program with a 'short-term memory' wherein intermediate knowledge is stored as the solution progresses. Likened to a blackboard (and actually named that in other systems), the function is to provide a single repository for knowledge that the inference engine can reference as it dynamically manages the control strategy. In TRALI, the context is organized by 'objects' (records), which are broken down into 'attributes' (fields) (18,p.5). Each object describes one component of the intersection under study and the system generates as many objects as it requires for the particular situation. Likewise, attributes describe the parameters of an object. For example, each traffic movement (or flow) is defined as an object. Attributes for the flow object include the flow name, the volume, the angle in, the angle out and the number of lanes. Values are placed into these attributes as they are input (from the user), derived (by execution of a rule) or calculated (as the result of an invoked algorithm).

Using this architecture, it is obvious that the inference engine has but to look at the context to determine not only what is known, but also what is not known and hence, what needs to be known. In this regard, the architecture of TRALI is particularly interesting, as the objects of the context behave very similarly to frames, in that the attributes are analogous to the slots and the variables are actually the domain specific knowledge that is input or generated. Similar to classic frame representation, TRALI is representing a certain amount of its process knowledge in the object. Unlike a frame representation, however, TRALI incorporates no default knowledge within its objects. This is probably a function of the experimental nature of this program.

3.4.3 Knowledge Base

The knowledge base, as previously discussed, contains 237 rules in an IF...THEN format. There is essentially no limit to the number of <condition> clauses a rule may possess. Likewise, any number of <action> clauses may be controlled by one rule. Further, there is no format within the knowledge base that dictates the physical ordering of the rules. This quasi-unstructured environment, commonly referred to as 'rules of equal level', allows the inference engine nearly complete control in determining and executing the solution strategy. Other expert systems, however, allow the ordering of the rules to play an integral part in the formation of the control strategy. For example, the expert

system ESIE (a relatively primitive expert system shell written in Pascal that runs on an IBM PC), uses the order of rules, as they physically appear in the knowledge base, to establish the control strategy (9,p.20). Within this architecture there is no such problem as the resolution of conflicting rules, because the first rule found, whose <condition> clauses are satisfied by the context, is considered the dominant rule and becomes the one fired. An obvious advantage is that this order dependent strategy requires a less sophisticated inference engine. The value of this must be weighed against the disadvantages; paramount of which is a lack of flexibility in adjusting the inferencing scheme from a central location. Likewise, the bookkeeping difficulties associated with an order dependent inferencing strategy in a large knowledge base would be staggering. For these reasons, all future production level expert systems will no doubt employ the concept of 'rules of equal level' in their knowledge bases.

3.4.4 Inference Engine

To demonstrate how the inference engine uses the process and domain knowledge to control the solution strategy, assume that the current goal of the system (also expressed as a rule in process knowledge), is to calculate the phase distribution. In order to achieve this, the goal rule informs the inference engine that it needs the volumes for all flows. The inference engine then interrogates the flow objects in the context and determines if it has

available all the necessary flows. If all flows are available, the goal rule <condition> becomes true and the inference engine fires the rule's <action> which, in this case, would be the act of invoking an algorithm to calculate the phase distribution. Conversely, if all flows were not available within the objects, then the inference engine would interrogate the rule base, looking for a rule whose <action> was the missing flow. Once found, this rule's <condition> clause would be matched against the knowledge within the context. If the required <condition> was found within the context, then the rule would be fired, and the value for the flow added to the context, which would ultimately precipitate the firing of the goal rule. On the other hand, if the <condition> was not found within the context, then the inference engine would begin another search of the knowledge base to look for a rule that had, as its <action>, the <conclusion> of the previous rule. This recursive process would continue until either the problem was solved, or the system has executed all the rules for which it had <condition> information, and therefore, by itself, could add no further knowledge to the context.

3.4.4.1 Conflict Resolution in the TRALI System

In addition to controlling the solution strategy, it is also incumbent upon the inference engine to resolve conflicts between competing rules whose <condition> statements evaluate as true. Depending upon the size of the rule base, it is not uncommon for

the addition of one new piece of knowledge to activate any number of rules. In a testing session of TRALI (18,p.9-13), the average number of rules the inference engine had to choose from was 14. The reason this number is so large is because TRALI uses a forward chaining control strategy to determine which goal rules to invoke. As discussed in the previous chapter, forward chaining results in the evaluation of rules regardless of whether or not the <action> clause brings the program any closer to the solution. In a backward chaining environment, the additional constraint of validating a goal rule (hypothesis) would have the effect of decreasing the number of candidate rules. However, even in this situation, the possibility of more than one rule's <condition> being true is quite high. When faced with this predicament, the inference engine usually uses some heuristic to break the tie. In the case of TRALI, this heuristic is probably directed by the structure of OPS5, the General Purpose Representation Language used. OPS5 has a unique feature that time stamps every new piece of knowledge that is added to the context. This utility allows for the differentiation between 'old' knowledge and 'new' knowledge. With this information at its disposal, the system nominally breaks ties by firing the rule that incorporates the 'newest' knowledge (1). Since the authors of TRALI make no reference to any particular

tie breaking scheme, it is assumed that they have chosen to utilize this function of their programming language.

3.4.4.2 Conflict Resolution in the MYCIN System

Other systems, however, use schemes that are quite different from this. In the case of MYCIN (an expert system designed to diagnose viral disorders of the blood), the heuristic used to resolve conflicts between conflicting rules is the aggregate of the certainty factors. If 2 rules are eligible to fire, MYCIN computes the certainty factor of the <action> for each rule, which is based on the certainty factors for the <condition> clauses for each rule. The rule that would provide the <action> with the highest certainty factor is judged as the 'dominant' rule and is the one fired (2,p.53).

3.5 Evaluation of Effectiveness

In their conclusions on the effectiveness of TRALI, Zozaya-Gorostiza and Hendrickson (18,p.14) point out a number of advantages their system enjoys over algorithmic based design programs, as well as a number of implementation and programming problems that were discovered during the course of the development.

3.5.1 Advantages

First, TRALI is not constrained to a predetermined geometry or the assignment of flows along predetermined routes. Additionally, the heuristics contained within the

process rules can be used effectively when the optimum alternative "... contemplates multiple competing figures of merit. For example, a good design might not be that which minimizes total delay, but one having an acceptable delay and short average queues ..." (18,p.14). Secondly, they maintain that the knowledge base can be easily updated and enriched, without the need for reprogramming in the classic sense. The merit of this advantage must be tempered with an understanding of the problems inherent when using knowledge from more than one source (expert), as discussed in the previous chapter. The third advantage addressed speaks to the architecture of the entire scheme that removes the requirement for the designer to anticipate all possible cases that may be encountered. This is an advantage not only of TRALI, but a pivotal benefit of all knowledge based expert systems.

3.5.2 Disadvantages

However, Zozaya-Gorostiza and Hendrickson also point to some problems encountered with TRALI, and expert systems in general. Specifically, they address the need for a domain expert that can identify the rules and strategies used for a manual solution to the design problem. A second problem is the nonportability of the hardware on which they chose to implement their system. This is probably due to their choice of OPS5 as a language, and the fact that the system development occurred prior to April 1986. Since then, a version of OPS5 that operates on an IBM PC has been

released. Additionally, a few other General Purpose Representation Languages that are implemented in IQLISP have recently become available. Since this language also runs on the IBM PC, there is no reason for nonportability to remain a problem in the future.

The last problem encountered speaks to the incompatibility of existing design programs and expert systems in regards to the expert system controlling the execution of algorithms resident within other languages. This was seen as a problem since system compatibility is virtually nonexistent and the OPS5 language is highly inefficient for numerical computations. Interim solutions to this problem may again lie with implementation in IQLISP or PROLOGUE, both of which are better at numerical evaluation than OPS5. Unfortunately, neither is near as efficient as FORTRAN, PASCAL or any of a number of languages designed specifically for numerical manipulations. With the current magnitude of emphasis in this field, in addition to the research being done on 5th generation hardware and software, it is highly probable that the next few years will see a language that can accommodate both the list processing requirements of the expert system and the number-crunching demands of algorithms with equal ease.

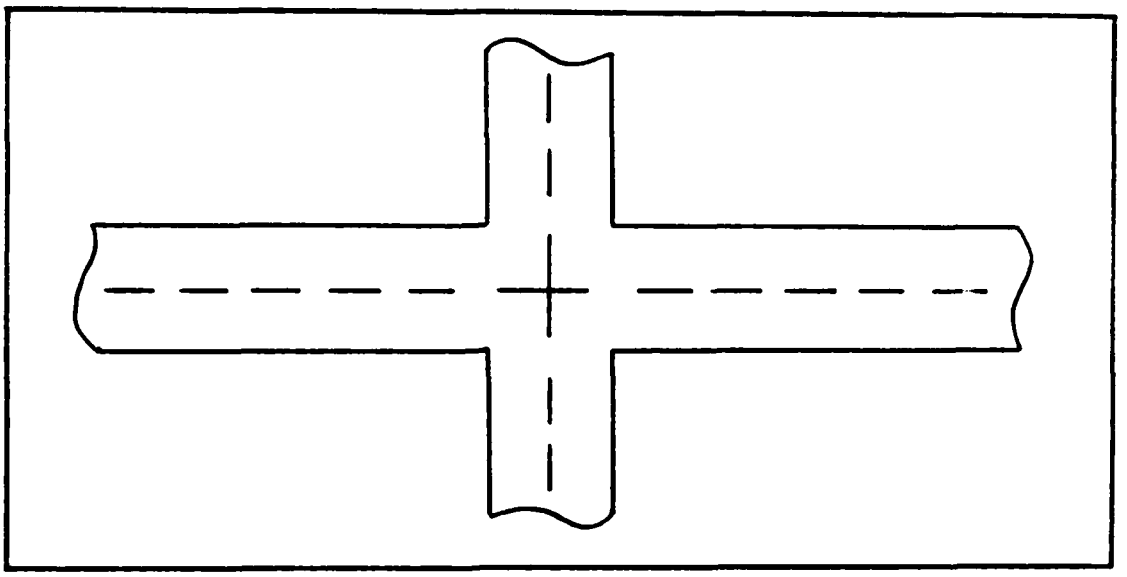


FIGURE 6a.

CONVENTIONAL INTERSECTION GEOMETRY consisting of four through flows and four left turning flows. (Right turns are assumed to occur with the through movement).

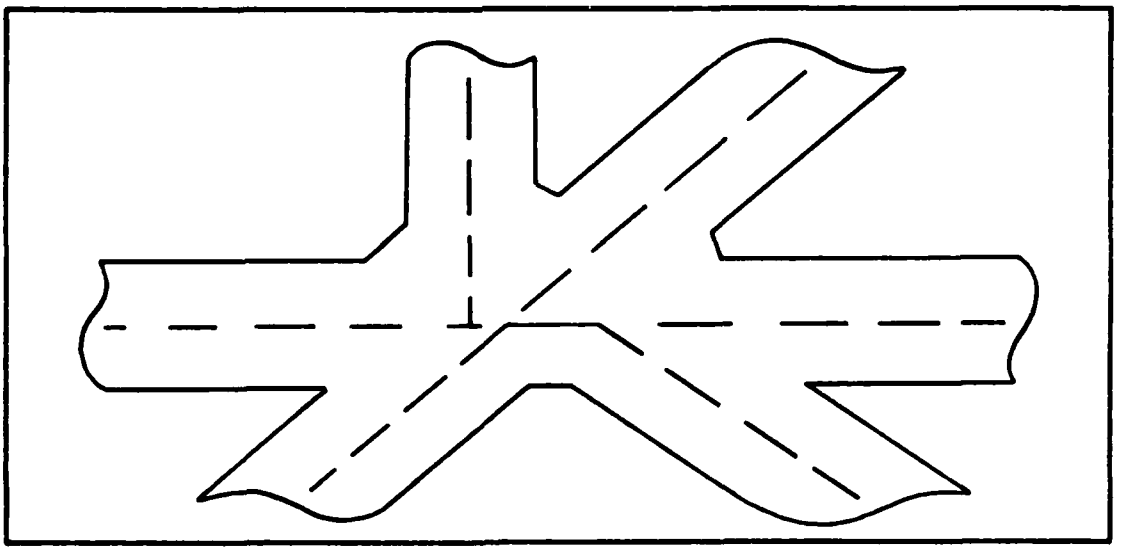


FIGURE 6b.

REPRESENTATIVE UNCOMMON GEOMETRY of the type that TRALI is designed to provide an isolated intersection timing solution for.

CHAPTER FOUR

CASE STUDY OF THE PLATFORM MODEL

4.1 Introduction to the PLATFORM Model

The next expert system to be examined was developed in the early 1980s by Stanford University and IntelliCorp for the purpose of validating certain knowledge representation structures and control strategies within the domain of project management. This program operates on a set of thirteen top level construction tasks such as design platform, cast concrete base, make deck structure and tow to site. Since there are no activity breakdowns within the top level tasks, the system is identified as a "proof of concept" testbed, and not a working prototype (6,p.61). Even so, the skill of an activity scheduling assistant that the system provides, demonstrates a very practical application in the area of construction engineering (6,p.58). Unlike the examination of TRALI, which dealt with the mechanics of the expert system program, the discussion of the PLATFORM Model will focus more on the integration of the expert system into the project management concept and the effective utilization of its capabilities.

The program itself is written in the KEE General Purpose Representation Language, which is implemented in a number of LISP dialects. This language supports an environment where a number of knowledge representation schemes can be used simultaneously. In the case of

PLATFORM, frames (herein called 'units') are used to store the domain knowledge, whereas rules (structured in an IF ... THEN format) contain the process knowledge (i.e.- control strategies) (6,p.59). This hybrid environment has more flexibility than a system where only one type of representation can be used, as the particular strengths of each representation scheme can be exploited as required (6,p.60). In addition, KEE also supports LISP functions which allow PLATFORM to engage easily in the numerical computations of CPM and PERT network evaluations.

4.2 Knowledge Representation

The basic frame used in the PLATFORM architecture is the ACTIVITY unit. As with other systems, this frame is composed of slots (attributes), into which variables (domain knowledge) are entered. Some of the nominal slots found in this unit are (6,p.62):

1) COMPLETION STATUS (complete or incomplete)
2) DESCRIPTION
3) COMPUTE EXPECTED DURATION
4) ACTUAL DURATION
5) EXPECTED DURATION
6) MOST LIKELY DURATION
7) OPTIMISTIC DURATION
8) PESSIMISTIC DURATION
9) DURATION VARIANCE
10) ON CRITICAL PATH? (yes or no)
11) SCHEDULE IMPACT CAUSES

As can be seen from the above list, the slots of this unit are generic enough to apply to any activity. Thus, by using this unit (frame) for the representation of all activities,

the system can theoretically handle projects of any conceivable size, with any number of activities, by simply generating as many units as are required (6,p.74).

The slots COMPLETION STATUS, DESCRIPTION and ACTUAL DURATION are entered by the user. The values for the MOST LIKELY DURATION, OPTIMISTIC DURATION and PESSIMISTIC DURATION, likewise entered by the user, are used by the system to calculate the EXPECTED DURATION by using the standard PERT equation (6,p.67):

```
EXPECTED DURATION = ((OPTIMISTIC DURATION)    +
                    (4*MOST LIKELY DURATION) +
                    (PESSIMISTIC DURATION) )/6
```

This algorithm is invoked only when the value of the slot COMPUTE EXPECTED DURATION signifies to the inference engine that a reevaluation is required. The slot will take on this value whenever an event occurs that alters any of the PERT durations. The variance of the EXPECTED DURATION is then entered as the value for DURATION VARIANCE.

The slot ON CRITICAL PATH? is loaded with a value whenever another LISP method, called PRINT PATHS, executes a forward and backward path evaluation through the network. This slot is binary and can have the value of yes or no, represented graphically as a "*" or " ", respectively (6,p.63).

4.3 Knowledge Utilization

Up to this point, the discussion of PLATFORM has shown no capabilities that separate it from any of the numerous

algorithmic based project management programs that are currently available. However, the final slot in the ACTIVITY unit, SCHEDULE IMPACT CAUSES, inaugurates this departure from the norm and begins to demonstrate the true power and flexibility of expert systems.

4.3.1 Weaknesses in Current Practices

Conventional project management techniques, including even those that are considered progressive, rely heavily upon CPM/PERT type networks for their decision making processes. While these do provide good information regarding durations and dependencies, they show only the end result of many decisions that were made by the project design team. Levitt and Kunz point out that this is a major flaw with current project management practices:

> The expert's knowledge about the task domain that was employed during the schedule creation is unavailable subsequently for use by other members of the project team in interpreting interim project performance or in updating the project's schedule. (6,p.57)

Effectively, the personnel charged with executing the project are given but a cryptic glimpse of the underlying reasons surrounding many of the design conclusions. The results of this lack of communication are, not surprisingly, network scheduling tools that gather dust on the superintendent's desk and only come into use when the company lawyer must substantiate a claim or enter into other litigation. This is obviously a dismal situation, as the

effort that went into the planning of the project is not available during the execution, and the project manager is forced to redesign the wheel and second guess the project planners at nearly every decision point.

4.3.2 A Knowledge Based Remedy

To alleviate this shortcoming, one of PLATFORM's basic design criteria was the inclusion of domain knowledge about the risk factors and dependencies of the activities constituting the project. This information can then be used by the system to forecast activity and project completion times, based on the performance of those activities that have been completed. The slot SCHEDULE IMPACT CAUSES is the mechanism wherein PLATFORM incorporates the expert's knowledge about each particular activity of the project. Specifically, this slot contains a listing of risk factors that were initially believed to adversely or constructively affect the activity's duration (6,p.71). For example, generic activity impacts could include:

1) LABOR PRODUCTIVITY
2) WEATHER/ENVIRONMENTAL CONDITIONS
3) MATERIAL AVAILABILITY
4) QUALITY CONTROL COMPETENCE
5) FULFILLMENT OF LEGAL REQUIREMENTS

PLATFORM stores the applicable impacts in the SCHEDULE IMPACT CAUSES slot of each activity. This provides the system with an understanding of all the factors that constitute a particular activity, and allow it to identify trends in production.

4.3.2.1 Use of the SCHEDULE IMPACT CAUSES Slot

When an activity is completed, its actual duration is placed in the ACTUAL DURATION slot of the ACTIVITY unit. This operation triggers a number of actions. First, the actual duration is compared to the expected duration to determine if the associated activity impacts represent an accelerating or a delaying trend (i.e.- actual duration less then estimated duration or actual duration greater than estimated duration, respectively). Next, the system interrogates the other activity units that completed with the same type of trend (short or long). If any are found, a match is then initiated on the impacts of the second activity, and impacts common to both are identified as accelerating trends (where both completed activities were short) or delaying trends (when both are long). (PLATFORM's lexicon specifies the former as a 'KNIGHT' and the latter as a 'VILLIAN'). With an impact so identified, the system then searches all the uncompleted ACTIVITY units for an impact match. If an unstarted or uncompleted activity is found to match, then its EXPECTED DURATION value is changed to either the OPTIMISTIC DURATION or the PESSIMISTIC DURATION, depending upon if the impact is a 'KNIGHT' or a 'VILLIAN'. Once all the activities have been handled, and their expected durations modified as appropriate, the system then invokes the CPM algorithm that performs

a forward and a backward pass on the network to update the critical path and the project duration.

4.3.2.2 Example of the SCHEDULE_IMPACT_CAUSES

The Platform Model contains two early tasks that have CONCRETE PRODUCTIVITY as an activity impact. These activities are building the graving dock and casting the concrete base. In an example run, both of these activities were caused to complete early. This had the effect of identifying the impact of CONCRETE PRODUCTIVITY as a 'KNIGHT'. When uncompleted activities were then searched, it was found that the CONCRETE PRODUCTIVITY impact existed in two other activities; Slipform1 and Slipform2. The EXPECTED DURATIONs of these two activities were revised to the OPTIMISTIC DURATION, and the CPM was evaluated. The result was a decrease in the project duration and a change in the critical path (6,p.71).

4.3.2.3 System Safeguards

It should be noted that a judgement is requested from the user at two decision points in the above described updating process. The first is at the point where the system initially identifies an impact as a 'KNIGHT' or a 'VILLIAN'. Here, the user has the opportunity to accept or reject the impact. An example of a situation that may cause rejection would be the identification of early problems with a batch plant that the user knew had been corrected, and thereby

posed no 'VILLIAN' effect on subsequent activities containing CONCRETE PRODUCTIVITY as an impact.

The second and subsequent opportunities to accept or reject the system's recommendation occur when the system proposes to change the EXPECTED DURATION value for an activity. A user rejection at this point may be for the reason that one activity, containing a certain impact, is being serviced in a different way than the other activities also containing the impact. For example, if a certain inadequate batch plant is identified as a 'VILLIAN', then the system will recommend the alteration of the EXPECTED DURATIONs of all activities that contain the impact BATCH PLANT CONCRETE PRODUCTIVITY. The user would most likely concur with those recommendations that were concerned with activities receiving concrete from the inadequate batch plant. However, activities obtaining their supplies from other batch plants would obviously not be affected, and therefore, the user would no doubt disagree with the recommendation to alter their EXPECTED DURATIONs. While it is recognized that this potential problem could be alleviated by entering separate impacts for all the batch plants being used, it is also important to recognize that some impacts of slightly different character (i.e.- batch plants A and B, for example) will always need to be combined into a somewhat broad, single impact. Because of this, it is

necessary that the user be allowed a voice in the process, whenever the system is making decisions based on these broad factors.

With these safeguards in place, it is obvious that the system can be effectively monitored and will not produce schedules that bear no resemblance to the real world situation it is modeling. In fact, as the designers intended, PLATFORM operates very much like a scheduling assistant, in that the system accumulates information and presents it in a fashion that allows informed decisions to be made.

4.3.3 Supplemental Benefits

In addition to employing domain knowledge to assist the project manager in identifying trends and accessing their impact, PLATFORM's basic approach has the added benefit of eliminating some of the conceptual problems that have long plagued PERT methodology (6,p.66). Specifically, one basic assumption of PERT that is universally known to be untrue, is that activity durations are independent. In its approach to project updating, PLATFORM not only ignores this assumption, but actually capitalizes upon the dependencies that exist between activities; as its foundation, PLATFORM assumes that activities have highly correlated durations. It uses this assumption, along with the domain knowledge of risk factor assignments to each activity, to weave an intricate web of interdependencies. The resulting model

represents the real world situation with an accuracy and a flexibility that a purely algorithmic approach can never hope to achieve.

4.4 System Integration and User Interface

Another basic design criteria of PLATFORM appears to have been the user interface. Unlike other systems that require the user to decode cryptic output, this expert system displays its results in the form of dynamic, graphical representations of the ACTIVITY units (6,p.67).

4.4.1 ACTIVITY Graphics Representation

FIGURE 7, on page 68, shows an example activity image where five of the unit slots are displayed. These slots are (from top to bottom) a critical path indicator, the activity name, an indicator of the schedule performance (ACTUAL DURATION or updated EXPECTED DURATION measured against the initially planned duration), the ACTUAL DURATION and the current EXPECTED DURATION. Containing this information, the graphical image bears a functional and aesthetic similarity to the nodes on a precedence diagram. Extending the comparison of the precedence diagram one step further, FIGURE 8 shows an 'Image Panel' that reproduces the information contained within selected slots of all the ACTIVITY units. Functionally, this 'Image Panel' allows both the system and the user to transfer information. When the system is communicating to the user, the graphical images represent windows to the ACTIVITY units, showing

realtime changes in the slot values. Conversely, when the user is communicating with the system, modifying slot values on the graphical image (by use of the keyboard or system mouse) will have the effect of changing the same slot values within the ACTIVITY unit. Effectively, the expert system is communicating with the user through the medium of an automated precedence network that is presented on a monitor screen.

4.4.2 NETWORK Representation Prior to Start

FIGURE 8, on page 69, shows the 'Image Panel' at a time prior to the project start. Note that nine of the thirteen images contain an asterisk in the CP slot, indicating those activities that are on the critical path. The performance 'dials' of all the activities are shown in the NORMAL position because no activities have yet been completed. (Recall that at least two activities must complete before the system attempts to identify 'KNIGHTs' and 'VILLIANs': a necessary prerequisite before the inference engine can alter the scheduled performance of an activity). The lower third of the images contain information from the DURATION slots. All the images contain a 'NIL' in the lower left hand corner (ACTUAL DURATION slot). This is a LISP value for a variable that contains no data. The numbers opposing the 'NIL' slots (lower right hand corner), are the initially planned EXPECTED DURATIONs in months. Note also the bar graph, part

way up the right hand side of the screen. This graphic shows that the project duration time is initially estimated to be 27 months.

4.4.3 NETWORK Representation Subsequent to Start

FIGURE 9, on page 70, shows the same 'Image Panel', but at a later time when four activities have completed. These are the Project Start (leftmost activity), the Build Graving Dock activity (up and to the right of Project Start), the Cast Concrete Base activity (adjacent to Build Graving Dock) and the Design Platform activity (below and to the right of Project Start). The durations and schedule performances for these activities are tabulated below:

ACTIVITY	EXPECTED DURATION	ACTUAL DURATION	SCHEDULE PERFORMANCE
Project Start	0	0	Normal
Graving Dock	14	11	Short
Concrete Base	6	4	Short
Design	7	8	Long

The 'Image Panel' in FIGURE 9 mirrors this progress, with the 'dials' showing the schedule performance. Additionally, the images for the activities Slipform1 (adjacent to the Cast Concrete Base activity) and Slipform2 (the second image to the right of the Slipform1 activity) show a schedule performance of 'short' and an EXPECTED DURATION that equals the OPTIMISTIC DURATION input for the activities (one month, in both cases). This is due to the system's identification

of a 'KNIGHT' in the CONCRETE PRODUCTIVITY impact for the Build Graving Dock and Cast Concrete Base activities, as discussed earlier. The ramification of identifying this 'KNIGHT' is to decrease the EXPECTED DURATION for activities that shared the impact. The net result is twofold: 1) to change the critical path (note the new locations of the asterisks) and 2) to decrease the entire project duration from 27 months to 21 months. The intermediate bar of the duration graph in FIGURE 9 shows a duration of 22 months. This value was provided before the system searched for other impacts and, hence, represents the projected duration due only to the acceleration of the completed activities (6,p.68-73). Additionally, the user interface is continually active, thereby allowing the user to query the system at any point for its strategy and methodology.

4.5 Evaluation of Effectiveness

The ramifications of PLATFORM's success are twofold: 1) the domain of project management is validated as a viable realm for the implementation of AI systems and 2) the function of an 'intelligent' scheduling assistant can be accomplished by using construction task knowledge and project management knowledge within the knowledge base of an expert system (6,p.73). While PLATFORM deals with only thirteen top level tasks, the methodologies and control strategies employed could easily be extended to handle the volume and detail of an actual construction project. Along

these lines, Levitt and Kunz identify a number of areas that need to be addressed commensurate with such an undertaking (6,p.74-75). These are:

> 1) the difficulties in graphically displaying the 'Image Panel' precedence network for large projects with numerous activities.
>
> 2) the requirement to input large amounts of project data from numerous and diverse sources.
>
> 3) the degradation of system processing speed as the number and complexity of activities and rules increases.

As with the TRALI expert system, PLATFORM's utility lies not as a domain expert, but rather, as an expert assistant. This is not so much a breach of faith with the goals of AI research, as it is an admission that the embryonic stages of development will, of necessity, yield systems of a less capable nature.

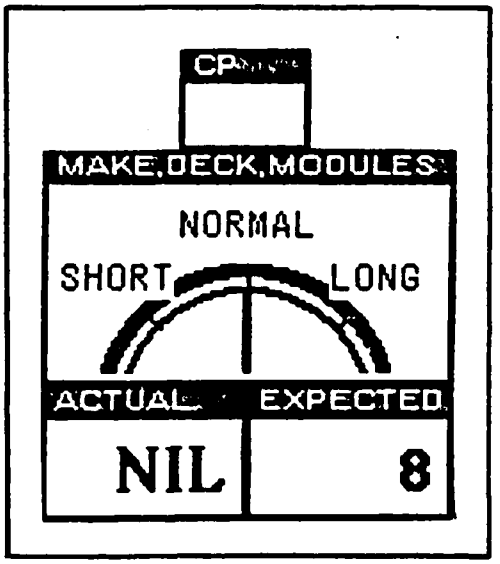


FIGURE 7.

GRAPHICAL IMAGE OF THE ACTIVITY unit, showing the slots ON CRITICAL PATH?, ACTIVITY DESCRIPTION, a measure of schedule performance, ACTUAL DURATION and EXPECTED DURATION (6,p.67).

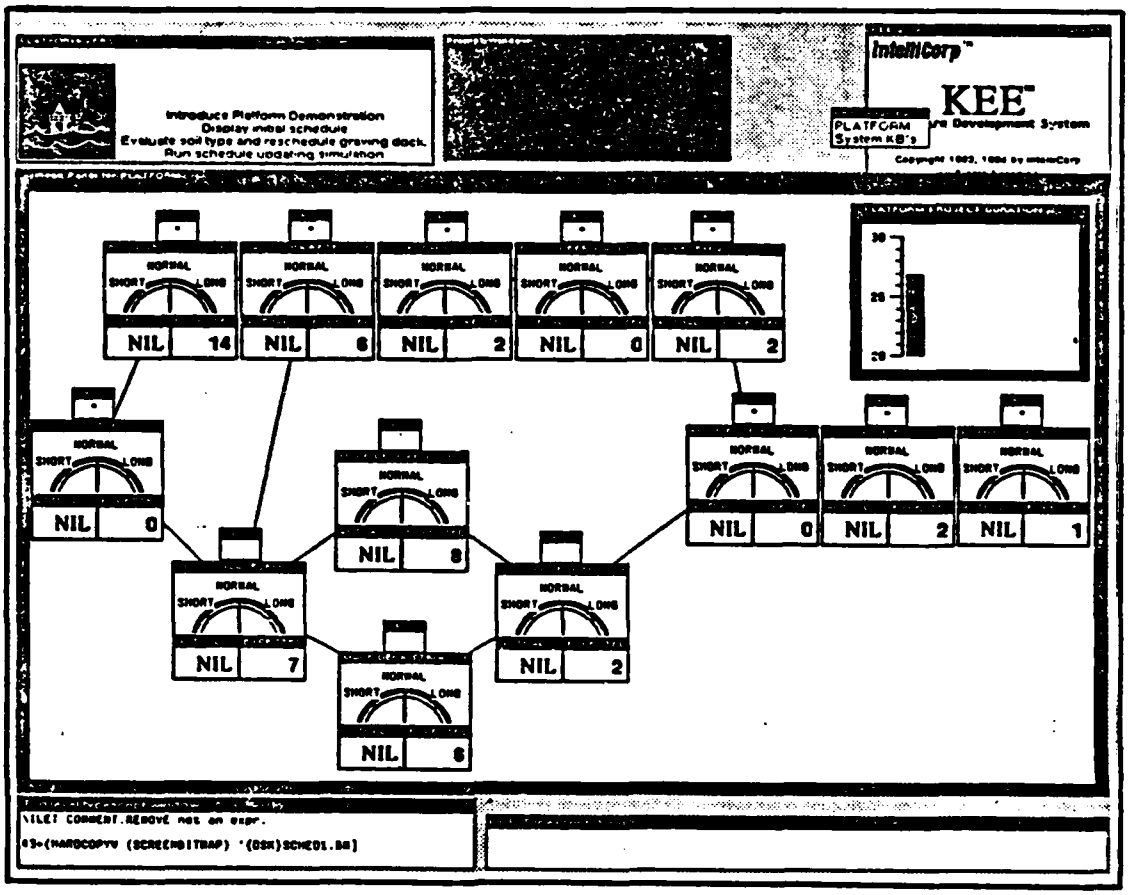


FIGURE 8.

PROJECT PRECEDENCE NETWORK prior to job commencement. Note that the dials for scheduled performance all indicate normal duration (6,p.69).

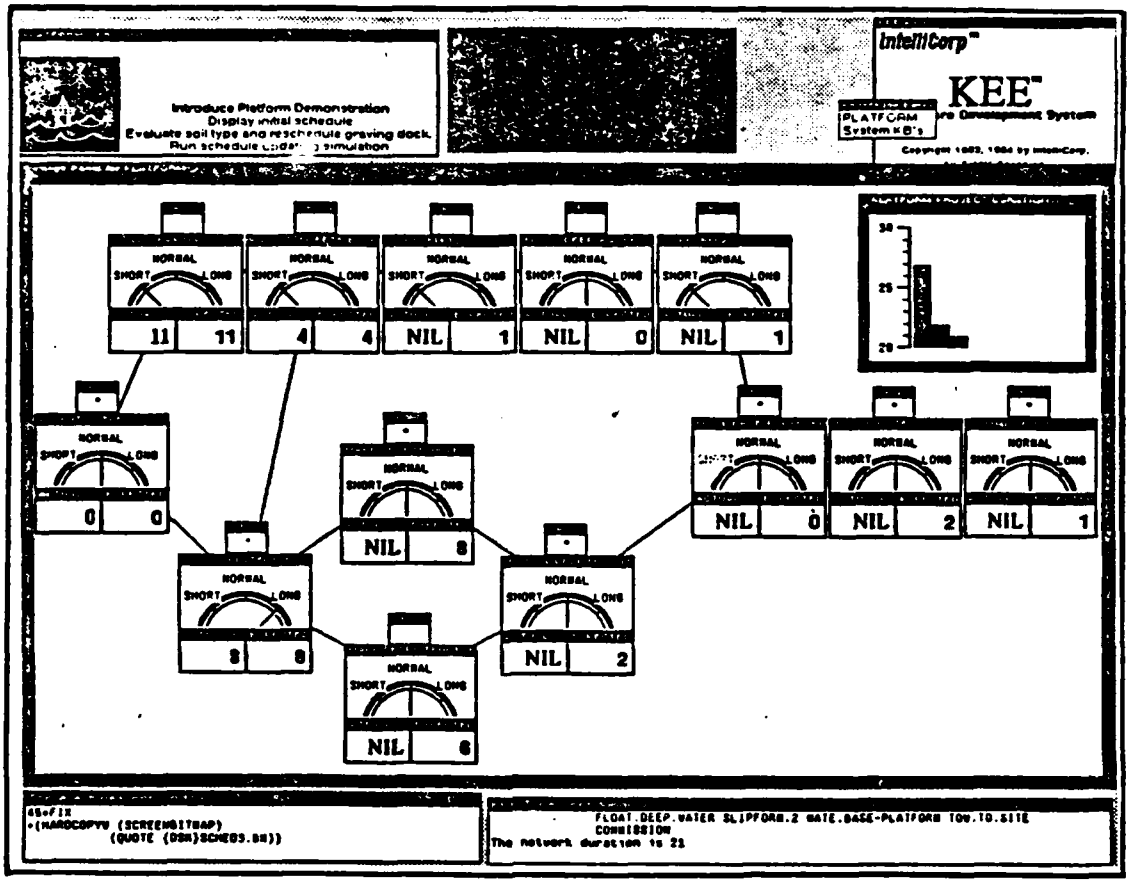


FIGURE 9.

PROJECT PRECEDENCE NETWORK after completion of four activities. Note the changes made to the scheduled performance of the uncompleted activities (Slipform1 and Slipform2) due to the identification of 'KNIGHTs' and 'VILLIANs' (6,p.72).

CHAPTER FIVE

CONCLUSIONS AND RECOMMENDATIONS FOR FURTHER RESEARCH AND DEVELOPMENT

5.1 Perspective on Enthusiasm

When a child receives a new bicycle, the first thing that friends want to do is ride it. A similar phenomenon is evident whenever a new class of computer program is introduced; persons from all imaginable fields attempt to fold, spindle and mutilate the computer technique to fit their particular application. The advantage of this is that the new procedure is applied to numerous problem classes. The disadvantage is that these applications to new problems are often at the hands of persons unknowledgeable about the strengths and limitations of the technique. This appears to be the case as the concept of expert systems begins to emerge from obscurity. Such is the fervor to find applications, in fact, that journals in the field are literally teeming with ideas for expert system utilization. Unbridled enthusiasm in this area, however, can quickly lead to dismal failure. As Stansfield discovered, after an attempt to construct an expert system that would act as a commodity market analyst:

> After a significant effort ... I am forced to the conclusion that an intelligent, real-world system of the kind envisioned is currently out of reach. [Specific problems encountered were] the complexity of the real-world domains, and the difficulty of describing the ways the experts deal with them. (12,p.591)

5.2 Characteristics of a Suitable Domain

One of the cardinal rules for expert system applicability that was apparently overlooked in the above endeavor was that which requires a genuine domain expert to exist. Recall the six necessary domain criteria discussed in chapter 2:

> 1) GENUINE EXPERTS MUST EXIST. In the absence of this necessary condition, expert systems would be required to extend domain knowledge and understanding. No computer program, however sophisticated, currently possesses this capability. In addition, the undertaking of 'cloning' solution strategies into the program's knowledge base presupposes the existence of those strategies. Domains that do not meet this criteria, for example, would include stock market speculation and commodities trading, as discussed above.
>
> 2) THE EXPERTS MUST GENERALLY AGREE ABOUT THE CHOICE OF AN ACCEPTABLE SOLUTION. While the problem solving strategies and methods of different experts do not necessarily have to match, accord on the final solution indicates a domain wherein the problems are solvable. As discussed, however, care must be taken not to include different expert's strategies within the same knowledge base. Domains that would be excluded from consideration, based upon this criteria, may include the problems of nuclear arms control and the national budget deficit.
>
> 3) THE EXPERTS MUST BE ABLE TO ARTICULATE AND EXPLAIN THEIR PROBLEM SOLVING METHODOLOGY. A domain that satisfies the first two conditions does not automatically meet this one. Recall the example of the PROSPECTOR expert system that was discussed in chapter one; the experts were unable to articulate their actual problem solving methodology, since they were not conscience of the actual mechanisms that they employed. The fulfillment of this constraint is a function not only of the domain, but also of the personalities and dispositions of the domain experts. An expert

system designed to manufacture Coca-Cola, for example, would probably be a failure since the experts in the field would be quite reluctant to divulge their trade secrets.

4) THE PROBLEMS OF THE DOMAIN MUST REQUIRE COGNITIVE, NOT PHYSICAL SKILLS. Used in this context, cognitive denotes a broad range of skills that run the gambit from meditative problem solving to vision and robot manipulator interfacing and control. In other words, the problem should not be to accomplish the task, but rather, how to accomplish the task. To this end, the activities of brick laying and telephone pole erection would not be good candidates, whereas the domains of foundation design and telecommunications system planning would.

5) THE TASKS CANNOT BE TOO DIFFICULT. As with the requirement for a domain expert, this constraint mandates that a solution exist and that the discovery of the solution be possible. A classic example of this criteria is the 3 bears analogy; the task should not be too difficult (a plan for world peace, for example), not too easy (taking the square root of a number), but just right.

6) THE PROBLEM SHOULD NOT REQUIRE COMMON SENSE OR GENERAL WORLD KNOWLEDGE. This criteria speaks mainly to the size and complexity of knowledge bases as well as to the early failure of General Problem Solver (GPS) type programs that attempted to deal with a broad range of problem classes. While it would be possible to build a system that would at least simulate common sense, the knowledge base size and depth this would require is well beyond the capabilities of current systems.

5.3 Justifications for Implementation

The above criteria describe domain characteristics that are considered important to the success of production level expert systems. Due to the high cost of development and implementation, however, these should be viewed as only necessary conditions and not sufficient unto themselves. The added dimension needed is justification; in what

situation and under what circumstances will the development of an expert system be justified? While this is not a principle consideration within a research and development environment, it is nonetheless of paramount importance to the practitioners within a domain. With this in mind, five possible justifications are presented (2,p.27). The existence of any one, when combined with the domain characteristics described above, should be sufficient cause to commence the implementation of an expert system.

1) THE PROBLEM SOLUTION, WHEN FOUND, SHOULD HAVE A HIGH PAYOFF. Simple economics dictates that the solutions provided by an expert system have a payback sufficient to cover the cost of the system.

2) HUMAN EXPERTS ARE UNAVAILABLE TO PERFORM THE TASKS. When the demand for a certain expertise exceeds the supply, expert systems may be employed to make up the difference. This would be especially attractive in a domain where the time required to develop a human expert was considerable.

3) HUMAN EXPERTS ARE UNABLE TO PERFORM THE TASKS. This justification speaks to those domains where human experts do exist, but certain problems within the domain, due to their complexity, defeat the application of the human expert. Problems of this nature include those that require an enormous number of calculations with the commensurate bookkeeping tedium.

4) SIGNIFICANT EXPERTISE IS BEING LOST WITHIN THE DOMAIN. This situation could occur for a number of reasons: an economic climate that compels experts to move to different fields or the lessening of importance of a field such that human experts are not replaced as fast as they are leaving. Whatever the reason for the loss of expertise, an expert system may well be justified in this situation.

5) DOMAINS THAT EXIST IN HOSTILE OR UNFRIENDLY ENVIRONMENTS. Prime candidate domains for this justification include space travel, the interiors of nuclear reactor containment vessels and deep water mining and salvage operations. To be of any value in these areas, the expert systems would obviously need to be controlling some physical apparatus, and not just passively solving problems.

5.4 Applications in Civil Engineering

As previously observed, journals in the field of Civil Engineering are literally teeming with ideas for expert system applications. Among these are equipment diagnosis and repair, structural diagnosis, site investigation, environmental sensing, quality control, structural design, operations planning, construction planning and equipment monitoring, to name but a few (10,p.57-8).

Even though no production level expert systems yet exist, there are a number of prototype systems currently under evaluation. In addition to the two described in previous chapters, the fields of sensor interpretation and structural design have also produced systems with some interesting capabilities.

5.4.1 Sensor Interpretation

In the field of geotechnical interpretation, an expert system prototype called CONE has been developed with the capability to interpret cone penetrometer data. From the raw data provided by the penetrometer, the CONE system infers soil stratigraphy parameters about the various layers of soil tested (10,p.60). With this, the system uses its

knowledge base to classify the soil layers, infer structural parameters and develop trend lines for the area under study.

CONE is a rule-based system that is written in OPS5. Currently it is limited in scope to off-site analysis. However, an interesting proposal has been made to implement the system within a microprocessor environment, attached to the physical cone penetrometer. This approach is a natural step in the progression of expert system utilization, as it is undertaking to put the expert system's power to use at the time and place where it can be of most benefit.

5.4.2 Structural Design

The area of design boasts a number of expert system prototypes. One of these, named HI-RISE, operates as an engineering assistant for the design of high rise buildings. This system, written in PSRL and utilizing a frame based knowledge representation, is one of the most extensive systems yet developed in any field (10,p.61).

Given basic parameters about a structure to be designed, HI-RISE develops a number of competing alternative designs, ranks them according to a set of preliminary criteria and presents the one with the highest ranking to the user (10,p.60). One of the interesting features of this system is its ability to interface with other expert systems and knowledge bases (called knowledge modules) during the course of the design/selection process. As one example, a smaller expert system, called HI-COST, is employed to

develop cost estimates for the designs of HI-RISE. These estimates are then returned to HI-RISE for use in the ranking process.

5.5 Applications in Project Management

When considering all the arguments concerning domain criteria and development justification, another area within the field of Civil Engineering that emerges as a potentially qualified candidate domain is the area of construction engineering, specifically project monitoring and management. Satisfying the domain criteria, there is no doubt that expert project managers and superintendents exist, nor is there generally much disagreement about the choice of an acceptable solution to a problem. Further, the problem classes germane to this domain are not of a highly theoretical or difficult nature and generally tend to require cognitive skills, at least at the decision level addressed by the expert system. The only domain criteria that this field appears not to meet, at least on the surface, is the one mandating that the solution not require common sense or general world knowledge. As is well known, a large percentage of problems in project management require these exact ingredients for a solution. Fortunately, this is not a fatal problem: the field of project management is diverse enough to allow expert system application in subareas that do not require common sense, general world knowledge or creativity for the solution. A prime example

of this was seen in the expert system PLATFORM, where the program's only stand-alone capabilities were the bookkeeping and matching of activity risk factors and the computations associated with a PERT network. To be sure, the system could offer conclusions and recommendations it developed based upon the knowledge it contained. However, recall that the user was required to accept or reject a recommendation at each decision point that required the application of common sense or general world knowledge. Used in this fashion, as an expert assistant, a system's ability to comply with the 'common sense' criteria is not essential.

This is good news, as the justification that speaks to the high payoff potential of a solution is certainly applicable to this domain. Considering the tremendous monetary losses that poor project management produces, and the huge profits that good project management can yield, the eventual introduction of production level expert systems into this arena is a given. It is only a question of how soon and in what areas.

5.5.1 Cost and Time Control

The evolution of expert systems within the domain of project management will no doubt be driven by simple economics; those systems that provide the best cost-to-benefit ratio will be at the forefront of development and implementation efforts. With this in mind, the subareas of cost control and time control emerge as particularly good candidates for expert system development,

since a small improvement in efficiency can yield a disproportionately large payoff. McGartland and Hendrickson (7,p.298) develop the argument that the close association of these two areas would make them inseparable within an expert system. Specifically, the system envisioned would analyze activity costing and completion milestone data to forecast completion times and final costs. If the methodology of the PLATFORM Model was also included, then the system would be able to anticipate problems with unstarted activities based on the project performance to date. Finally, the 'intelligence' of the system could be used to trap input errors and question information that did not appear 'reasonable'.

To accomplish these objectives, periodic information about each activity would be required by the system. Depending upon the level of definition desired, daily or weekly input would consist of the following:

1) estimated percent complete
2) cost to date
3) actual labor used to date
4) actual material used to date
5) actual equipment used to date

This information would be compared with the estimated cost and completion information for each activity that was either input at the start of the project or updated by the system during the course of an earlier run. Using these empirical values the system would then interrogate its rule base to determine the significance and effect of each. Possible

conclusions and recommendations that the system could be requested to make may include the following (7,p.301):

> 1) Recommendation for improvements in resource utilization and resource leveling strategies based upon project experience to date and past trends.
>
> 2) Updating of the remaining schedule based on the same experience to date and past trends.
>
> 3) Prediction of problems that may occur during future phases of the project.
>
> 4) Suggestions to remedy the problems identified above.

While a system of the kind herein described would not be able to manage a project by itself, the aggregate of these capabilities would, in fact, provide the project manager with an 'expert assistant' in the area of cost and time control. This would have the effect of allowing the project manager to concentrate on the supervision and common sense aspects of the project.

5.5.2 Purchasing and Inventory Control

Another area of project management that promises a high payback potential for expert system implementation is purchasing and inventory control. Like the dynamics of cost and time, the correlations between purchasing and inventory mandate that both be included in the same expert system. The objective of an expert system in this area would be to minimize the overall project material cost by comparing the cost of purchasing the materials early, and storing them in inventory, to the cost of not having the materials available when they are needed (7,p.303).

Information required by this system, for each item of material addressed, would include the consumption rate, the storage cost, the delivery time, the delivery probability, and the cost to the project if the material were unavailable. The knowledge base of the system would then use this information to recommend reorder points and suitable inventory levels.

If this system were integrated into the cost and time control expert assistant described previously, the resulting capabilities would surpass the sum of the two. Purchase and inventory control could then be tied to specific activities within the project. A change in a particular activity start time or duration, either detected by the system or recommended as a change, would cause an appropriate modification in the purchasing and inventory strategies in use for the materials required by the activity. Additionally, the resource of material could be dynamically leveled, based upon the slack time for activities that the cost and time control system determines.

Add to this combination an expert system that controls hiring and manpower, and it becomes obvious that as expert system concepts are applied to more and more subareas, the aggregate capability may theoretically approach that of the human project manager, minus the components of common sense, general world knowledge and creativity.

5.5.3 Integration of FUZZY LOGIC

Of the three components described above, only the area of creativity appears to be unapproachable at this time. Current research in the area of Fuzzy logic is beginning to produce methodologies that mimic applied common sense and the application of general world knowledge; decisions made on a 'gut' feel, or those made in the face of competing, conflicting or contradictory information.

Nguyen describes the fundamental concepts of Fuzzy logic and their application in the realm of non-numerical problem solving:

> The notion of fuzzy sets ... deals with certain sets that may admit partial membership. A fuzzy set is thus a set with members having a continuum of grades of membership, from 0 to 1. Fuzzy set theory [a subset of Fuzzy logic] is particularly suitable for application in the modeling of classes of problems involving fuzzy or imprecise data ... for which the information may involve uncertainty of a subjective type, such as vague description, human errors, omissions and mistakes. (8,p.232,240)

In other words, a fuzzy set can be described as the set of possible solutions to a problem, where the members of the set are the individual solutions themselves. For example, the set of solutions to the situation where an activity's actual duration is exceeding its estimated duration may include the following members:

1) hire more labor
2) rent more equipment
3) divert labor from another activity
4) divert material from another activity
5) move to next activity and finish later
6) do nothing and absorb the excess time

All of these members (solutions), and many more, have partial membership in the solution set. The degree of membership is dependent upon the criteria used to judge the members. In this example, the criteria may well depend upon the reason for the delay: if shovel availability is less than estimated, then solutions dealing with labor and material will have low grades (near 0), while solutions that address the equipment problem will enjoy greater membership (a higher grade). The advantage of this structure is that a solution can be dynamically selected from a preexisting set, based upon the magnitude and importance of other factors.

The field of artificial intelligence has yet to capitalize on Fuzzy logic to any great extent. The expert system CONE, as previously described, does make use of this methodology to describe the heuristics of expert judgment in its inferencing scheme (10,p.60). However, the lack of widespread use is only indicative of the embryonic nature of both fields. With time, Fuzzy logic will no doubt become an integral part of expert system methodology, thereby making the component of creativity the sole remaining responsibility of the human user.

5.6 Consequences for the Practitioner

Expert systems hold the potential to herald a revolution greater than that introduced by the microcomputer. This is because expert systems will allow the true capability and potential of microcomputers to be utilized; for the first time, there will be application programs available that actually assist the user, and do not simply regurgitate the input data in a disguised form.

For users in the construction industry, and other areas as well, this revolution will bring about a variety of benefits. Among these will be (3,p.132):

> 1) Shorter decision time, both in the field (project management) and in the office (designing, scheduling, etc). This is not because the program is making the decisions, but rather because it is screening out those factors that are irrelevant to a decision and thereby preventing the user from wasting time and attention.
>
> 2) Augmented professional judgement of the employed human experts, in that the expert systems will be available to offer 'second opinions' on critical decisions. Likewise, an expert system could also be employed as a 'knowledge based spreadsheet' (similar to Lotus 123, for example), to perform 'what if' analyses of a broad reaching nature.
>
> 3) The sharing of corporate expertise, as the expert's technical knowledge and reasoning are made available to the draftsmen, engineers and junior project managers. Additionally, this environment would infer an increase in the ability to train inexperienced professionals.

It is important to remember, however, that the acquisition of these capabilities is not without cost. In building an expert system tailored to a particular environment, the

price could easily run in excess of a few hundred thousand dollars for the hardware, software and knowledge acquisition. It is due to this, as well as the requirement to assemble and maintain a staff of experts during the development, that most companies will not implement expert systems until stand alone, off the shelf programs become available at a reasonable price. While this is not currently the situation, the marketplace will no doubt soon boast a number of generic expert system applications. Since these programs will very likely run on IBM PC compatible microcomputers, whose numbers will have greatly proliferated, the only cost to the user will be the capital cost of the program, the loading of any knowledge particular to the specific company and program maintenance/updating costs. FIGURE 10, on page 87, shows the inverse, logarithmic relationship of knowledge based system development cost, as a function of time in years. From this, it is obvious that expert systems will soon become very affordable.

The possibility of this evolutionary profile for expert systems suggests implications that should be considered by future users. As discussed above, the price and availability of 'packaged' expert systems, in a number of disciplines, will soon make them available to nearly anyone. The effect of this may be a dramatic increase of competition in the marketplace. In construction management, for example, simple, labor intensive jobs may soon be bid, and

won, by anyone who has an 'expert scheduler' program and the ability to hire enough labor. While the first 'expert assistants' for sale may not be very capable, the evolution of the field will do nothing but add more job types, of increasing complexity, to the list of those that an expert system can manage.

A corollary to the above scenario suggests the reduction of staff and middle management positions, due to the intrinsic ability of expert systems to function well at that level. On the plus side, this would mean lower payrolls, less hiring problems and a lower turnover rate. On the other hand, fewer middle management positions implies that fewer persons would be trained for the higher level positions, and that there would be a resulting smaller pool from which to choose the top management personnel (3,p.134).

While these scenarios may not evolve exactly as stated, the general impacts are clear. The widespread introduction of expert systems will most certainly change the complexion of the way businesses operate and, in all probability, the way that society as a whole runs.

5.7 Timetable for the Future

FIGURE 11, on page 87, depicts a look into the crystal ball, for a hint at the future of expert systems. Whether or not the forecast is off by a year or two is inconsequential. The reality is just around the corner, waiting to let the human race tinker with yet another Pandora's Box.

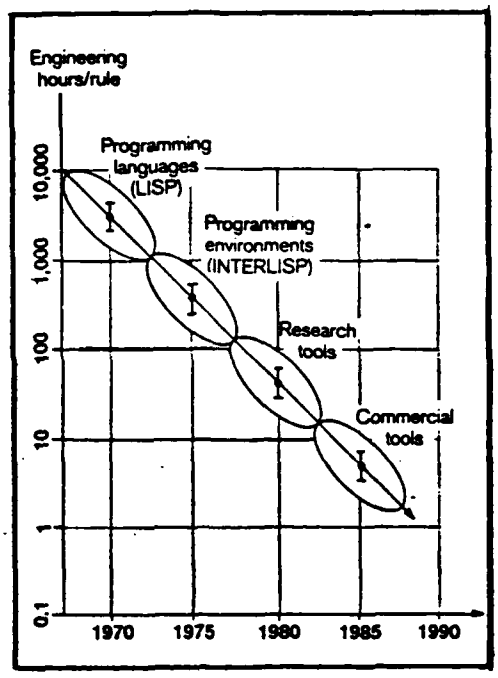


FIGURE 10.

The DECLINING COST of expert system knowledge base development as a function of time in years (4,p.9).

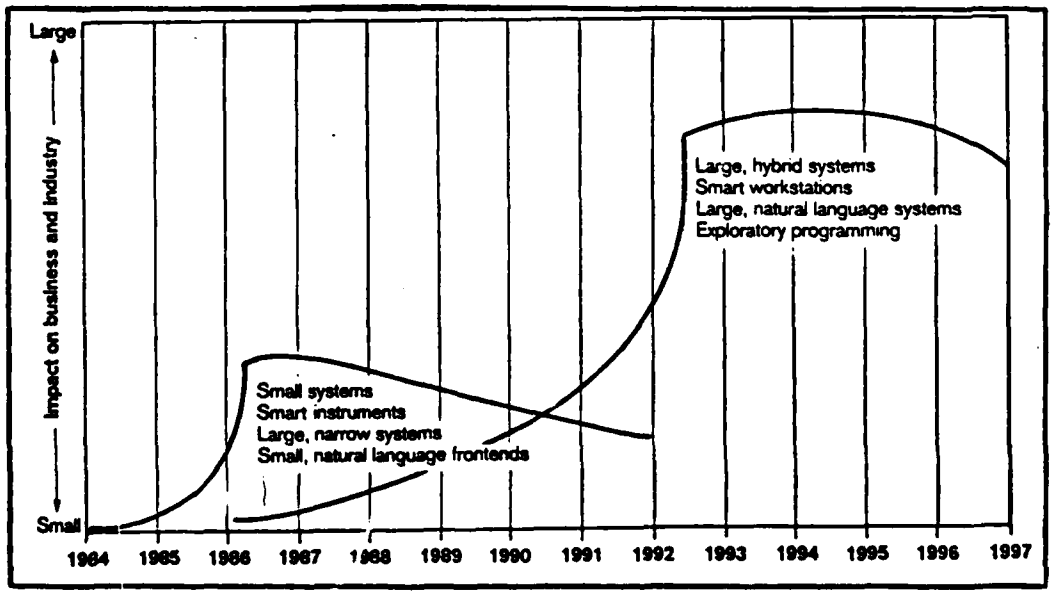


FIGURE 11.

THE TWO IMPACTS OF EXPERT SYSTEMS. The first impact deals with expert systems in the research and development environment, whereas the second impact demonstrates the accelerating effect of the marketplace (4,p.10).

REFERENCES

1. Dankel, Douglas, D., Lectures from CAP-6627 (Expert Systems), University of Florida, Spring Semester, 1986.

2. Dankel, Douglas D., Lecture Notes for CAP 6627 (Expert Systems), instructor xeroxed class notes, University of Florida, Spring Semester, 1986.

3. Elliot, Robert K. and John A. Kielich, "Expert Systems in Accounting", Journal of Accountancy, September 1985, p.126-143.

4. Harmon, Paul and David King, Expert Systems, John Wiley and Sons, Inc., New York, 1985.

5. Hendrickson, Chris T., Daniel R. Rehak and Steven J. Fenves, Expert Systems in Transportation Systems Engineering, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, 1985.

6. Levitt, Raymond E. and John C. Kunz, "Using Knowledge of Construction and Project Management for Automated Schedule Updating", Project Management Journal, Vol. XVI, No. 5, December 1985, p.57-76.

7. McGartland, Martin R. and Chris T. Hendrickson, "Expert Systems for Construction Project Monitoring", Journal of Construction Engineering and Management, Vol. 111, No. 3, September 1985, p.293-307.

8. Nguyen, Van U., "Tender Evaluation by Fuzzy Sets", Journal of Construction Engineering and Management, Vol. 111, No. 3, September 1985, p.231-243

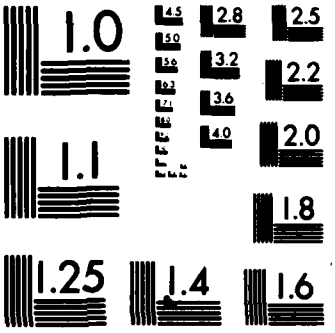
9. Reasor, Edward, ESIE - The Expert System Inference Engine - Knowledge Engineer's Manual, software support documentation, Lightwave Consultants, August 1985, p.20-21.

10. Rehak, Daniel R. and Steven J. Fenves, "Expert Systems in Civil Engineering, Construction Management and Construction Robotics", The Robotics Institute 1984 Annual Research Review, ed. Purvis M. Jackson, Robotics Institute, Carnegie-Mellon University, Pittsburgh, 1985, p.51-66.

11. Simon, Herbert A., "Artificial Intelligence Systems that Understand", The Scientific DataLink Index to Artificial Intelligence Research 1954-1984, Scientific DataLink, New York, 1985, p.224.

AD-A171 854 EXPERT SYSTEMS IN CIVIL ENGINEERING(U) FLORIDA UNIV 2/2
GAINESVILLE DEPT OF CIVIL ENGINEERING R A WALL 1986
N00228-85-G-3323

UNCLASSIFIED F/G 9/2 NL



12. Stansfield, James L, "Conclusions from the Commodity Expert Project", The Scientific DataLink Index to Artificial Intelligence Research 1954-1984, Scientific DataLink, New York, 1985, p.591.

13. The Handbook of Artificial Intelligence, ed. Avron Barr and Edward A. Feigenbaum, Volume 1, HeurisTech Press, Stanford, 1981.

14. The Handbook of Artificial Intelligence, ed. Avron Barr and Edward A. Feigenbaum, Volume 2, HeurisTech Press, Stanford, 1982.

15. U.S. Department of Transportation, Federal Highway Administration, SOAP84 User's Manual, Publication FHWA-IP-85-7, Turner-Fairbank Highway Research Center, McLean, January 1984.

16. Waterman, Donald A., A Guide to Expert Systems, Addison-Wesley Publishing Company, Inc., Reading, 1985.

17. Winston, Patrick Henry, Artificial Intelligence, 2nd ed., Addison-Wesley Publishing Company, Inc., Reading, 1984.

18. Zozaya-Gorostiza, Carlos and Chris Hendrickson, An Expert System for Traffic Signal Setting Assistance, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, 1985.

BIBLIOGRAPHY

"Artificial Intelligence", Engineering News-Record, March 28, 1985, p.20-23

Dankel, Douglas, D., Lectures from CAP-6627 (Expert Systems), University of Florida, Spring Semester, 1986.

Dankel, Douglas D., Lecture Notes for CAP 6627 (Expert Systems), instructor xeroxed class notes, University of Florida, Spring Semester, 1986.

Elliot, Robert K. and John A. Kielich, "Expert Systems in Accounting", Journal of Accountancy, September 1985, p.126-143.

Harmon, Paul and David King, Expert Systems, John Wiley and Sons, Inc., New York, 1985.

Goff, Kenneth W., "Artificial Intelligence in Process Control", Mechanical Engineering, October 1985, p.53-57

Hendrickson, Chris T., Daniel R. Rehak and Steven J. Fenves, Expert Systems in Transportation Systems Engineering, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, 1985.

Levitt, Raymond E. and John C. Kunz, "Using Knowledge of Construction and Project Management for Automated Schedule Updating", Project Management Journal, Vol. XVI, No. 5, December 1985, p.57-76.

King, Michael S., Steven L. Brooks, and R. Michael Scheafer, "Knowledge-based Systems", Mechanical Engineering, October 1985, p.58-61

McGartland, Martin R. and Chris T. Hendrickson, "Expert Systems for Construction Project Monitoring", Journal of Construction Engineering and Management, Vol. 111, No. 3, September 1985, p.293-307.

Nguyen, Van U., "Tender Evaluation by Fuzzy Sets", Journal of Construction Engineering and Management, Vol. 111, No. 3, September 1985, p.231-243

Reasor, Edward, ESIE - The Expert System Inference Engine - Knowledge Engineer's Manual, software support documentation, Lightwave Consultants, August 1985, p.20-21.

Rehak, Daniel R. and Steven J. Fenves, "Expert Systems in Civil Engineering, Construction Management and Construction Robotics", The Robotics Institute 1984 Annual Research Review, ed. Purvis M. Jackson, Robotics Institute, Carnegie-Mellon University, Pittsburgh, 1985, p.51-66.

Simon, Herbert A., "Artificial Intelligence Systems that Understand", The Scientific DataLink Index to Artificial Intelligence Research 1954-1984, Scientific DataLink, New York, 1985, p.224.

Standfield, James L., "Conclusions from the Commodity Expert Project", The Scientific DataLink Index to Artificial Intelligence Research 1954-1984, Scientific DataLink, New York, 1985, p.224,591.

The Handbook of Artificial Intelligence, ed. Avron Barr and Edward A. Feigenbaum, Volume 1, HeurisTech Press, Stanford, 1981.

The Handbook of Artificial Intelligence, ed. Avron Barr and Edward A. Feigenbaum, Volume 2, HeurisTech Press, Stanford, 1982.

The Handbook of Artificial Intelligence, ed. Paul R. Cohen and Edward A. Feigenbaum, Volume 3, Heuristech Press, Stanford, 1982

U.S. Department of Transportation, Federal Highway Administration, SOAP84 User's Manual, Publication FHWA-IP-85-7, Turner-Fairbank Highway Research Center, McLean, January 1984.

Waterman, Donald A., A Guide to Expert Systems, Addison-Wesley Publishing Company, Inc., Reading, 1985.

Winston, Patrick Henry, Artificial Intelligence, 2nd ed., Addison-Wesley Publishing Company, Inc., Reading,1984.

Zozaya-Gorostiza, Carlos and Chris Hendrickson, An Expert System for Traffic Signal Setting Assistance, Department of Civil Engineering, Carnegie-Mellon University, Pittsburgh, 1985.

END

DTIC

9—86