U.S. Department of Transportation

Publication No. FHWA-TS-88-022
December 1988

Developing Expert Systems

Research, Development, and Technology
Turner-Fairbank Highway Research Center
6300 Georgetown Pike
McLean, Virginia 22101-2296

FOREWORD

This technology share report provides an overview of expert systems and problems that are amenable to solution by expert systems, and guidelines for building expert systems. The guidelines were developed by the Federal Highway Administration to address the questions faced by highway administrators such as, "Does this technology have application in highway engineering and operations?", "What types of problems can be addressed?", "How are these computer programs different from other computer programs?" Properly constructed expert systems can be very effective training aids as well as supporting tools for virtually every aspect of highway engineering and management.

Copies of the report are available from the National Technical Information Services, 5285 Port Royal Road, Springfield, Virginia 22161, (703) 487-4690.

Stanley Byington

Stanley R. Byington, Director
Office of Implementation

NOTICE

Technical Report Documentation Page

1. Report No. FHWA-TS-88-022	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Developing Expert Systems		5. Report Date December 1988 6. Performing Organization Code
7. Author(s) David Barnett, Charles Jackson, and James A. Wentworth		8. Performing Organization Report No.
9. Performing Organization Name and Address STATCOM, Inc.		10. Work Unit No. (TRAIS) NCP 3H3C0106 11. Contract or Grant No. DTFH61-87-Z-00129
12. Sponsoring Agency Name and Address Federal Highway Administration Office of Implementation 6300 Georgetown Pike McLean, Virginia 22101-2296		13. Type of Report and Period Covered Task Final Report January 1988 - September 1988 14. Sponsoring Agency Code

15. Supplementary Notes

FHWA Contract Manager: James A. Wentworth (HRT-20)

16. Abstract

Expert Systems are computer programs designed to include a simulation of the reasoning and decision-making processes of human experts. This report provides a set of general guidelines for the development and distribution of highway related Expert Systems. This expands the guidelines provided in Chapter X, Expert Systems, of the Information Resources Management Manual. Included in this set of guidelines is information on developing, distributing and maintaining Expert Systems. The general development guidelines include discussions of: (1) a representative set of problem types that are amenable to solutions using Expert Systems; (2) a description of the major components of a typical Expert System and how these components interact in a problem solving situation; and (3) a set of Expert System development guidelines and a set of guidelines for maintaining an Expert System.

17. Key Words Expert System, Knowledge-Based Expert System, Artificial Intelligence, Knowledge-Base, Knowledge Engineering.	18. Distribution Statement No restrictions. This document is available to the public through the National Technical Information Services, Springfield, Virginia 22161.

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 30	22. Price

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

DEVELOPING EXPERT SYSTEMS

1. INTRODUCTION

Expert Systems represent a technology that is gaining acceptance in many industries but have not received the prominence they deserve in the highway community. The Federal Highway Administration (FHWA) and others are beginning to consider this technology for its potential application in highway engineering. Expert systems are computer programs that include a simulation of the reasoning and problem solving processes of human experts. These programs offer a means to capture the knowledge and experience of current professionals and organize, save, and apply this information to further the highway program. These programs cannot replace the human expert or decisionmaker or trainer; but they are nevertheless a valuable tool for the transportation specialist.

This document provides guidelines for the development, documentation, and distribution of microcomputer-based Expert Systems. The guidelines are directed at expert systems developed for wide distribution, not systems developed for use by a single or small number of users. It is the intent of these guidelines to promote broad acceptance of the use of expert systems, allowing maximum flexibility to each application. The following are important elements of a successful Expert System development program:

* A firm commitment of resources and support by executive management are critical elements for assuring the success of an expert system development.

* Within the organization, there must be an influential advocate of the Expert System. Ideally this includes both a developer and a user.

* The end users must be identified and their needs and skills considered. The requirements and needs of the end user must be a major factor in the system's planning and design.

* Both the problem to be addressed and the expected output from the expert system must be clearly defined.

* There are recognized experts in the field and there is general agreement among these experts on the knowledge required to solve the problem.

* The system should be demonstrated at identified milestone points during the development process. Three demonstrations are recommended.

2. DEFINITION OF EXPERT SYSTEMS

The most distinguishing characteristic of Expert Systems-type computer programs is that they are designed to mimic the problem solving behavior of one or more experts in the field of application. In the ideal case, a given Expert System contains an exact model of the reasoning processes that the expert(s) would use in solving the set of problems the Expert System is designed to solve. This means that in the ideal case the Expert System would reach the same conclusion(s) the expert(s) would reach if faced with the same problem.

The utility of a given Expert System depends on: (1) how faithfully it mimics an expert and (2) the level of expertise the expert has in the problem domain. If the Expert System's model of the expert's expertise is "good" and the expert himself (or herself) is "good" then the Expert System will perform admirably. However, models can never be 100% accurate and no expert is omniscient. Because of this, it is important that users of Expert Systems exercise caution in interpreting the answers produced by these systems.

2.1 FORMALISTIC VIEW OF EXPERT SYSTEMS

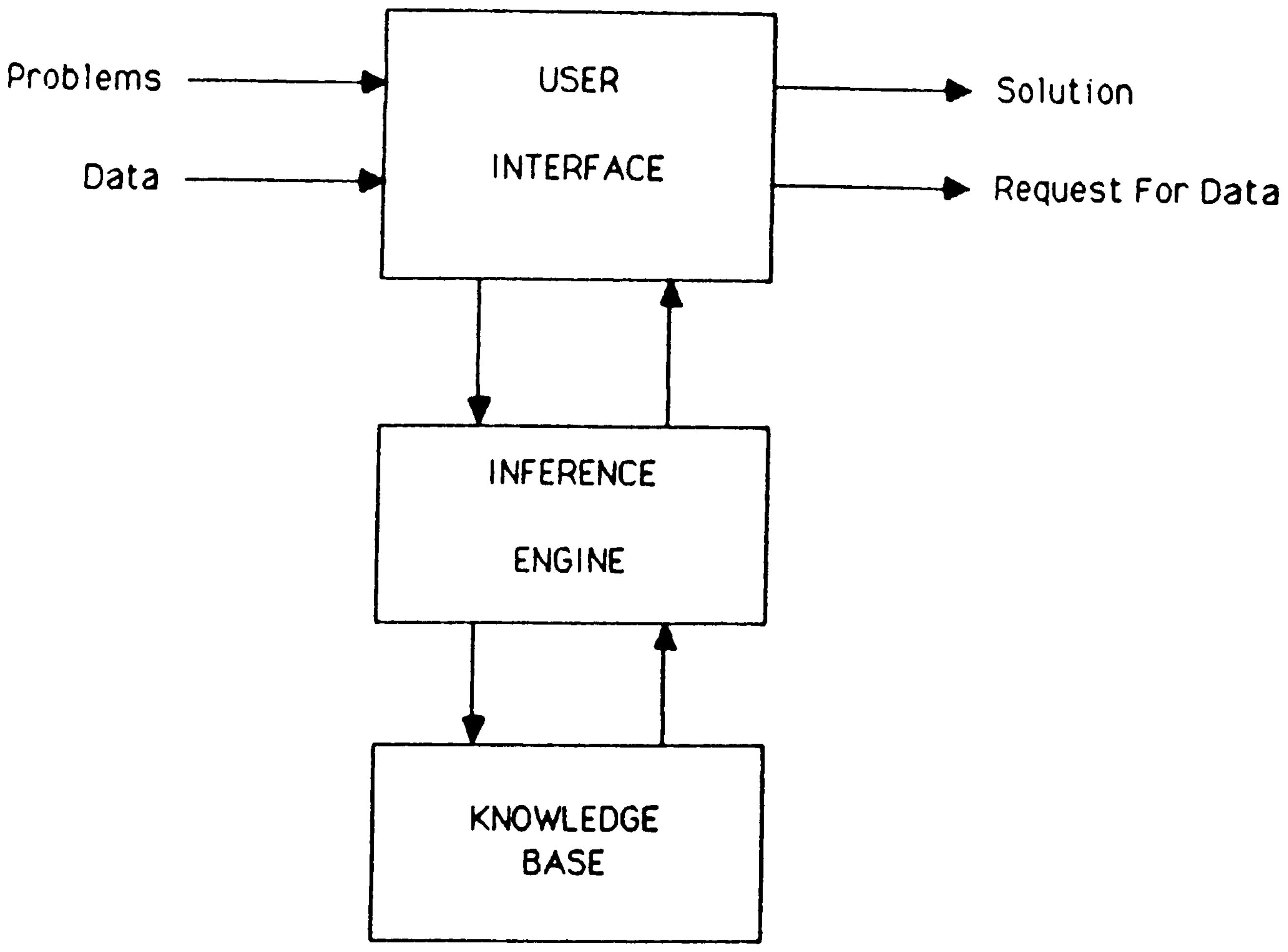
An Expert System, as does any computer program, consists of: (1) a set of inputs; (2) a set of outputs; and (3) a set of function modules which are designed to map the set of allowable inputs into the desired set of outputs. Figure 1 provides an illustration of how these various components of a typical Expert System interrelate. Using the system analysis formalism, the model of a typical Expert System includes:

(1) A problem (i.e., the set of allowable inputs),
(2) A solution (i.e., the set of desired outputs),
(3) A Knowledge Base function module,
(4) An Inference Engine function module, and
(5) A User Interface function module.

The primary purpose of the Knowledge Base module is to serve as a repository for the Expert System's domain specific problem solving knowledge and to provide this knowledge on demand to the system's Inference Engine module. The knowledge base also interfaces to the User Interface allowing the user to add, delete and modify this knowledge.

The primary purpose of the Inference Engine module is to develop solutions to the user's problem(s). In so doing it uses various types of reasoning strategies to infer solutions from both Knowledge Base supplied information and user supplied information.

Figure 1
Block Diagram Of An Expert System Model



The role of the User Interface module is to provide a means for the Expert System to receive instructions and data from users and to transmit results and requests for more information to the user. The user interface module also allows the user to change the Expert System's knowledge base and to specify control and reporting requirements.

2.2 PROBLEMS THAT ARE AMENABLE TO SOLUTION BY EXPERT SYSTEMS

There are a number of different ways to define the types of problems that can be addressed by Expert Systems. Several different categories have been discussed in the Expert Systems literature (see for examples, References 1 and 9). The categories discussed below cover the problem types that are applicable to highway related Expert System development.

2.2.1 Diagnosis

The basic problem in diagnosis is to examine a system's outputs (i.e., its symptoms) and to infer a cause (hence a remedy) for these symptoms. For example, in a pavement management system, the symptoms might be distress data (e.g., number of potholes), the cause might be weather, and the remedy might be local patching. The types of knowledge that are typically used in modelling diagnostic problems include:

* How does the system "normally" behave? (i.e., what facts do we know about the steady state behavior of the system?)

* What rules relate specific symptoms to specific causes?

* What rules relate specific causes to specific remedies?

The goal of the system diagnosis effort is to map system symptoms into either a specific cause for system malfunction or a set of candidate causes.

The specific problem solving operations that are performed during an attempt to diagnose the reason why a system is malfunctioning depends upon the type of system being diagnosed. For systems such as most hardware and software systems where the problem solver is able to test the system down at the subcomponent level, problem solving generally involves the top-down generation of inputs and the comparison of the resultant outputs to the expected outputs. This process is performed in an iterative fashion, moving down from the system level to the subsystem level until a cause for the system malfunction has been determined. Diagnostic problem solving for non-decomposable systems (such as biological systems) is normally done by attempting to match the system's symptomatic behavior with the symptoms that would be induced by some candidate set of causes of system malfunction.

Expert System-based diagnostic systems are normally designed to assist a human diagnostician by searching through the possible causes of system malfunction to generate a possible set of candidates. The human diagnostician then chooses the most likely cause from this candidate set(or collects more data).

2.2.2 Interpretation/Classification

The basic problem in interpretation/classification is essentially an identification problem. That is, data representing features or attributes of some entity is presented. This unknown entity is identified by comparing these features to those of a set of known entities. An example of this is the problem of choosing the appropriate delivery system for a new technology. There is a known set of delivery systems (technical report, seminar, film, brochure, etc.) with known features (cheap/expensive, technical/non-technical, quick/slow, etc.). The features of the target audience for the delivery system are compared with the features of the known delivery systems to determine the most appropriate delivery system for this audience.

The kind of knowledge that is typically used by expert systems performing interpretation/classification include:

* What facts are known about the features of the known objects?

* What rules are known about what constitutes a "good match" between sets of features?

* How much confidence should we have in any given match?

Expert System which solve problems in this area are designed to model the pattern matching ability of someone who is an expert in identifying objects in the problem domain.

2.2.3 Prediction/Forecasting

The basic problem in prediction/forecasting is to forecast the future state of some system based on the existing state and a knowledge of past events. For example, the future conditions to be forecast may be accident rates. The existing conditions could include traffic volume, speed limit, road conditions, etc. The knowledge of past events might include "...when the speed limit decreased x miles per hour, the accident rate decreases y per cent".

The kind of knowledge that is typically used by Expert Systems in this area include:

* What are the constituent components of the system under study?

* How are the constituent components related to each other? How

do they interact?

* What rules govern the relationship between a given component's inputs and its outputs?

The goal of prediction/forecasting expert systems is to determine the most likely system state that will result from the current system state.

2.2.4 Design

The basic problem in design is to derive a specification of how some system is to be built. This specification includes specific components to be used and their characteristics, and the relationships and interconnections between these components. As an example, consider the problem of pavement design. The components of this design might include the base layer or the sub-base layer, and the characteristics of these components could be thickness, material, etc.

The kind of knowledge that is typically used in Expert Systems that design include:

* What facts are known about the environment in which the system will be embedded? (e.g., we might ask for the above example " What is the soil strength? Traffic load?".)

* What facts are known about the constraints that the design must meet?

* What facts are known about the candidate set of primitive building blocks that the design will utilize?

* What rules are known about how these primitive building blocks can be connected?

* What rules of composition define how modules perform which are built from the building blocks?

The goal of the system design effort is to determine some interconnection of primitive building blocks that will satisfy the constraints on the design. Currently, most Expert System-based system designers have built-in generic models of the type of system whose design is desired. In this type of Expert System, the design process basically consists of specifying the generic model based on the user's particular design constraints. Another possible type of system design is one that both discovers the generic model which can be used to implement a set of design specifications, and also specifies the model. Needless to say the task faced by this type of Expert System is much greater than that faced by a design Expert System that just has to specify a given generic model.

2.2.5 Planning

Planning involves developing a set of actions which when performed will satisfy some goal. Depending upon the goal, the performance of this set of actions can occur in series, in parallel or some mixture of the two. When formulating a plan, the planner begins with an initial state and develops a set of actions that causes a movement in stages from the initial state to the goal state. For example, when developing a work zone traffic safety plan, the goal is to ensure the safety of motorists, pedestrians and workers while inhibiting traffic flow as little as possible. The actions necessary to implement this plan include:

* placement and removal of signs
* application and removal of pavement markings
* placement, maintenance and removal of channelization devices

The kind of knowledge that is typically used by expert planning systems include:

* What facts are known about the environment in which the plan will be implemented?
* What facts are known about a candidate set of primitive actions from which the plan will be composed?
* What facts are known about the constraints on the plan?
* What rules define how a given action moves you "closer" to some goal?

Problem solving during the planning process basically involves determining, for each stage in the plan, an action that will take one "nearer" to the goal. During this plan development process candidate actions are examined to see if they satisfy all of the relevant constraints and cause a transition to a state that is in some sense "closer" to the goal state. In the absence of some measure of closeness to the goal state, planning can degenerate into an exhaustive search of either all the paths that lead forward from the initial state or all of the paths that lead backward from the goal state.

As was the case with Expert System designers, most planners have built-in generic models of the type of plan desired. The planning process basically consists of providing specific parameters for the generic model based on the particular constraints. Again this problem is considerably simpler than the problem of both deriving a generic plan and providing parameters for it.

2.2.6 Monitoring

Monitoring involves analyzing a system's output(s) and implementing an appropriate set of corrective actions if this output wanders outside of some specified bounds. This is very similar to "real time diagnosis" except the bounds quite frequently are dynamic (time-varying) and to be outside of these bounds does not necessarily represent a flaw or failure. Furthermore, the goal in diagnosis is always "cure the system"; whereas, in monitoring the goal is often external to the system. For example, in automated traffic signal control, traffic flow data would be monitored and the corrective action taken might be to change the red\green\yellow signal timings. The specified bounds might change depending on whether it is rush hour, or a holiday, or special event (e.g., Super Bowl).

The kind of knowledge that is typically used in Expert Systems which perform a monitoring function include:

* What facts are known about the environment in which the system being monitored is resident?

* What facts are known about constraints on the system?

* What rules are known about the laws and dynamics of the system? (i.e., if some action is taken, how does it effect the system?)

Expert System-based monitoring is appropriate in those cases in which the laws governing a system's behavior are not precisely known and therefore heuristic control rules must be used or in those cases in which special actions must be taken if the system does not respond to the appropriate feedback controls.

3. COMPONENTS OF EXPERT SYSTEMS

As was stated in the introduction to this document an Expert System can be modeled as an interconnection of a User Interface module, an Inference Engine module and a Knowledge Base module. Following is a more detailed examination of the role that each of these modules plays in the overall functioning of a typical Expert System.

3.1 KNOWLEDGE BASE

The Knowledge Base is where the expertise of the system is stored. Although there are exceptions, the typical module consists of <u>facts</u> and <u>rules</u>. A <u>fact</u> is simply an assertion that a relation on a set of objects is true. Examples of facts are:

* "The roadway has 8 lanes."
* "The job is located on the shoulder of the road."

* "The average speed is 50 mph."

A <u>rule</u> is an assertion that some fact(s) is true provided that another set of facts is true. An example of a rule is:

```
*  "If the user is "FHWA" and
        the user's organizational level is "administrative" and
        the user's budget is "low" and
        the user's immediacy is "can-wait"
   then
        the choice for technology transfer is a brochure."
```

The benefits of a Knowledge Base are:

* Since the knowledge is not embedded in the control flow of the program, it is much easier to delete, insert and modify.

* The knowledge representation used in the knowledge base is more likely to be readable and understandable by the user than knowledge which is encoded via traditional programming languages.

3.1.1 Knowledge Representation Schemes

In the short history of Expert System development efforts, a number of different Knowledge Representation Schemes have been developed. Because there are so many different ways to represent knowledge in an Expert System, one of the key issues that must be resolved during the process of developing a given Expert System is which one will be used to represent knowledge about the problem domain of interest. The two main criteria that are used to make this decision are: (1) how well or how closely do the knowledge structures of a given knowledge representation scheme model the given problem domain knowledge and (2) how easily can problem domain specific inferencing be performed on the knowledge structures of a given knowledge representation scheme.

What follows is a brief description of the most widely used knowledge representation schemes. These descriptions will be given with respect to: (1) how the given schemes represent knowledge; (2) how inferencing can be performed on the knowledge structures which the given scheme uses; and (3) the characteristics of applications which are best suited for representation by the knowledge structures of the given scheme.

3.1.2 Rule-Based Knowledge Representation Scheme

The Rule-Based Knowledge Representation Scheme is probably the most widely used Expert System knowledge representation formalism. Systems that use this scheme typically represent knowledge about the problem domain in one of two forms. One form consists of a collection of facts. These facts normally have either the form "object has attribute" or the form "object 1 'relation' object 2". An example of the first form is "The pavement is new" and an example of the second form is "John likes Mary".

The second method that is used to represent domain knowledge in rule-based systems is in the form of "condition-action" rules. These rules have the form "IF 'condition is true' THEN 'perform action'." The condition part of these rules consists of premises which may or may not be true. These premises (if they are true within the scope of the Expert System's knowledge) will either be stored in the facts portion of the systems Knowledge Base or will be provided by the system user (either voluntarily or more normally as a result of prompting from the system).

The action portion of condition-action rules will typically consist of either an assertion that some fact is true or a procedure that the system is to perform. An example of the former is "If the car's lights will not come on,then the car's battery is dead". An example of the latter is "If the fasten seat belt warning tone sounds, then fasten your seat belt".

The primary type of inferencing technique that is used in rule-base systems is deductive reasoning. This technique consists of either: (1) deriving new facts from existing facts and rules or (2) proving that a given fact is true by combining selected facts in the knowledge base with selected rules. The former use of deductive reasoning is the method used in so-called "forward chaining" deductive reasoning and the latter is the method used in so-called "backward chaining" deductive reasoning.

Rule-based systems that use forward chaining deductive reasoning to achieve some goal will typically iteratively combine the facts and rules in the Knowledge Base to form new facts until: (1) the goal statement has been proven true; (2) no new facts can be derived; or (3) the system has reached some threshold on the allowable fact derivations or the allowable amount of processing time used. This type of reasoning is practical when there are relatively few initial conditions. With this strategy it is not necessary that the possible outcomes, or goals, be identified when the operation of the system is initiated (as in building designs from the ground up).

Rule-based systems that use backward chaining deductive reasoning to achieve some problem goal will typically attempt to prove that the goal statement is true by assuming that it is true and then searching through the available domain specific knowledge (both that portion of the knowledge that resides in the system's Knowledge Base and that provided by the system user) to find facts and rules that will support this assumption. This strategy is practical where the number of possible outcomes, or conclusions, are known and can be readily identified.

In addition to rule based Expert Systems that use either forward chaining deduction or backward chaining deduction exclusively, there are some rule based systems that are capable of reasoning by alternately performing both forward and backward chaining deduction. Such systems provide more design flexibility than single inference technique systems but the effective use of such bi-directional systems tends to require more skill than does the use of the single direction systems.

One of the major advantages of using a rule-based knowledge representation scheme in an Expert System is that this knowledge representation scheme tends to mirror the way many experts model the knowledge in various problem domains. As a result, the use of the rule-base knowledge representation scheme often makes it easier for the Expert System's designer to capture and represent the relevant aspects of a given problem domain.

Rule-based knowledge representation is best used. to model problem domains whose domain specific knowledge can be represented as a collection of relatively unstructured facts and rules.

3.1.3 Frame-Based Knowledge Representation Scheme

The Frame-Based Knowledge Representation Scheme is probably the second most widely used Expert System knowledge representation scheme and its popularity is growing as more Expert System's designers become aware of the power this representation scheme provides for representing problem domains that consist of highly structured knowledge. Systems that use a frame-based representation typically represent knowledge as a collection of record-like objects that have special features. The record part of the frame-based representation consists of a record type, a record name and a set of record fields. Unlike standard record-type data structures, frames have several special features that support the reasoning function of Expert Systems.

One such special feature is the ability of frames to be linked together in a tree-like structure and the facility for default reasoning to be performed on elements of this tree. This default reasoning consists of allowing frames to automatically inherit characteristics from frames that exist higher up in the tree hierarchy. In other words, in a frame-based system each frame automatically includes all of the features of its ancestors (i.e., its parent frames, its parent's parent-frames) even though these features are not specifically spelled out in the given frame's definition.

For example, suppose the frame CBR (Concrete Bridge Rail) is defined as follows:

```
Frame_Name --> CBR
  Height --> 32 inches
  Joint_Spacing --> 15' 0"
```

```
          Cross_Sectional_Area --> 1.8985 sq.ft.
          Weight_per_Linear_Foot --> 284.77 lbs
```

and the frame State_Street_CBR is defined as follows:

```
        Frame_Name --> State_Street_CBR
        Frame_Type --> CBR
          Weight_per_Linear_Foot --> 155 lbs
```

Because of the existence of the inheritance mechanism, the Expert System that contained these two frame declarations would know that the object, State_Street_CBR, is 32 inches in height because its parent object (i.e. CBR) is 32 inches in height and the specification for State_Street_CBR did not state otherwise. (NOTE: The fact that State_Street_CBR weighs 155 lbs per linear foot while its parent frame weighs 284.77 lbs per linear foot illustrates that frame systems allow inheritance to be over-ridden by object specific knowledge).

Another special feature of the frame-based knowledge representation scheme is the ability to attach procedures to frame fields. These procedures (when they are included within a given field specification) can be invoked either when a given frame field value is read or when the field value is changed or both. The use of this feature can aid frame-based systems in implementing heuristic searches through the Knowledge Base (by either restricting access to certain data within a frame or by constraining the system's ability to change selected frame's field values). This feature can also be used to provide a trace facility for use in debugging an Expert System.

Frame-Based Knowledge Representation Schemes support reasoning (i.e., inferencing) in a number of ways. In addition to default reasoning via inheritance, heuristic searching via procedural attachment, frame based systems will typically have a number of built in predicates which can be used to support reasoning by testing the properties of frames. Also there is nothing to prevent frame representations within rule-based systems.

One of the major advantages of using a Frame-Based Knowledge Representation Scheme is that the use of such a scheme allows the system's designer to structure knowledge in logical packages. This feature is most useful in those problem domains which have highly structured knowledge components.

3.1.4 Representing Uncertain Knowledge

The most widely used technique for Representing Uncertain Knowledge in an Expert System is 'certainty theory'. The need for such a technique results from the fact that at least part of the knowledge which a typical expert reasons with is inexact or uncertain. Sources of this uncertainty include insufficient data, random data and unreliable data.

The certainty theory method of representing knowledge associates a parameter represented by the symbol, CF (called the certainty factor or confidence factor), with every piece of knowledge in the system. CF is a measure of how certain the source of a given piece of knowledge is about the validity of the knowledge. Typically, CF can range between some lower bound L (usually 0) and some upper bound U (usually 10 or 100).

While certainty theory is in wide use in the Expert Systems field there are a number of problems with its use. Users who receive Expert System derived answers in terms of certainty factors should carefully about interpret the assumptions upon which the answers are based and thus the validity of the answers.

3.2 INFERENCE ENGINE

The Inference Engine is the problem solving component of the Expert System. Its role is to use the available rules to draw conclusions from the available facts. When the Expert System is given a problem to solve, the Inference Engine will first search the Knowledge Base to determine if it contains the solution to the problem. If the knowledge Base does not contain the solution then the Inference Engine will attempt to use the facts in the Knowledge Base, those that may be input on-line by the user and the rules in the Knowledge Base to derive a solution to the problem.

The tasks that the Inference Engine performs in attempting this derivation include:

* Selection of rules to examine
* Evaluation of rules
* Generation of new facts
* Retrieval of facts from both the Knowledge Base and the system user
* Generation of input problem solutions

The mechanism that a given Inference Engine uses to select the next rule to evaluate depends upon two main factors. These are the direction of inference being used (forward, backward or bi-directional) and whether or not heuristic measures are being used in the rule selection process.

In forward chaining systems the Inference Engine iteratively compares the available facts with the available rules and "fires" all of the rules whose premises are consistent with these facts. (A rule "fires" when its premise is a fact and hence its conclusion can be deduced.) This firing of rules will typically cause new facts to be generated and cause the Inference Engine to evaluate the system's rules again. This process continues until either a solution to the problem is attained or no new facts can be derived from the knowledge that is available to the Inference Engine. In backward chaining systems the Inference Engine evaluates the rules by assuming that the problem goal is true and working backward to determine what conditions must be true in order for this assumption to be true.

3.3 USER INTERFACE

The User Interface is the means by which the user and the Expert System communicate with each other. There are various kinds of dialogue which can take place through this interface. Examples might include:

* The user specifies a problem to be solved.
* The system requests additional information.
* The user questions why this information is needed.
* The user specifies that certain data is to be reported.
* The system reports a solution to the user.
* The user requests a modification in the knowledge base.
* The user requests an explanation of how a given solution was derived.

Although all computer programs have user interfaces (no matter how primitive), the Expert System's User Interface must be extremely sophisticated in order to allow the kind of dialogue suggested by the above examples.

Currently, there are a number of methods of implementing the function of the User Interface module. What follows is a brief description of a few of the methods that are either in wide use or are projected to be so.

The menu is the most widely used method of inputting data to an Expert System. When this method of data input is used the Expert System presents the user with a menu of input data choices. Typically the user can either select one of these choices (by manipulating a mouse pointing device or via a set of arrow keys) or type in an alternate choice.

The table/list is currently the most widely used form for outputting the results of an Expert System session. The items in this list will normally consist of a single solution (for those problems which have a single deterministic answer) or a ranked set of possible solutions (for those problems that determine their answers using probabilistic/confidence factor-based calculations).

Many Expert Systems are capable of presenting the results of their problem solving in graphical form. Such graphical results tend to be both more informative and aesthetically pleasing than are results that are presented in tabular form.

Currently, the use of natural language sentences as a means of user-to-Expert-System communication is at a relatively primitive stage. While some systems are able to respond to commands that are given using a limited subset of the English language, there are currently no widely available systems that can respond to general English sentences.

In addition to a limited ability to understand natural language input, some systems also provide their conclusions using natural language phrases. Most of these systems, however, contain these phrases in a "canned" form and are not able to compose phrases as a direct result of interacting with the user.

The next phase in natural language interface design will involve the incorporation of "knowledge based" natural language interfaces that are capable of generating and understanding a wide variety of sentences dealing with specific issues within a given problem domain. Expert Systems which incorporate this type of interface will in essence have an Expert System (the natural language interface) within an Expert System (the problem domain specific Expert System).

Another means of communicating with the user is the use of high resolution images stored on a videodisc. This technique is currently being developed by FHWA, NASA, and others.

4. BUILDING EXPERT SYSTEMS

Building an Expert System is a complex, time consuming task that requires the performance of a wide variety of different tasks. This section provides an overview of the Expert System development process and the major tasks that must be performed during this process.

There are four major categories of participants that are relevant in the Expert System building process. These are the advocate who champions the building of the Expert System, the end users of the Expert System, the domain expert(s) whose problem-solving expertise is to be modeled and the knowledge engineer who actually builds the Expert System. Although in the process of building a given Expert System the same person may at various stages of development take on different roles, it is important that it is recognized that these roles are distinct.

The role of the advocate who sponsors the development of the Expert System is to:

* Identify the need for an Expert System
* Define the problem domain
* Identify the intended user community
* Define the expected benefits that will accrue from the

intended audience using the Expert System
* Identify the expert(s) whose expertise will be modeled
* Choose the knowledge engineer who will develop the system
* Maintain (or plan for the maintenance of) the finished product

The end user is critical in the development of an Expert System and must be involved in the entire development process. The end user provides:

* Definition of the skill level of the user community
* Information on how problems are addressed in the field vs the prescribed solutions
* Advice on how the system must function (interact with the user) to be accepted by the intended users
* A cadre of supporters to test and promote the expert system once it is completed

The role of the domain expert in the Expert System development process is multi-faceted. First and foremost, the expert's problem-solving ability in the domain of interest serves as the model for the Expert System. Second and equally important, the expert must assist in quality control on the project and make certain that the Expert System faithfully represents a useful portion of the expert's knowledge. In essence, the expert must take some responsibility for insuring that the Expert System faithfully models his expertise. The expert's major task in fulfilling this responsibility is to assist in the design of a comprehensive set of test problems for use in verifying that the Expert System actually works.

The Knowledge Engineer has the task of developing a faithful model of the expert's problem solving ability in the domain of interest. Other tasks which the Knowledge Engineer must perform include:

* Implementing the model of the expert's knowledge
* Insuring that the implementation is as transparent as possible
* Documenting the Expert System
* Testing the Expert System

4.1 IDENTIFY THE NEED FOR AN EXPERT SYSTEM

The issue of identifying a need for an Expert System is one of the key issues in the Expert System development process. Identifying a problem and a specific task to perform with an expert system is difficult. There are usually numerous overlapping problems and issues in an area of interest and there is often disagreement among experts on the predominant causes.

Before an Expert System can be developed the need has to be established and the problem to be addressed clearly identified and defined. The following conditions must hold in order to justify the development of a system:

* The need for an Expert System must be identified
* The advocate, end user, domain expert, and knowledge engineer must be identified and committed to the success of the project
* Both the problem to be addressed and the output from the expert system can be clearly defined
* There are recognized experts in the field and there is general agreement among these experts on the knowledge required to solve the problem
* There is a shortage of experts (either currently or anticipated) and steps must be taken to alleviate this situation
* The effort required to develop the Expert System in the problem area can be predicted
* Resources are available to develop the system
* The domain expert must be able to dedicate sufficient time and effort to the development
* The end users can be identified

4.2 CLEARLY DEFINE THE APPLICATION FOR THE EXPERT SYSTEM

Once a suitable problem domain has been defined for the Expert System, the next task is to narrow the scope of the development effort by clearly defining the set of problems that the Expert System will be expected to solve. For example, it is not enough to specify that a "pavement diagnostic" Expert System be developed. The scope of such a problem is too broad for the current state of Expert System technology to be able to handle.

The narrower the scope, the better are the chances that the Expert System can be successfully built. On the other hand if the scope is too narrow the application becomes trivial. Since there is no deterministic method for specifying the appropriate scope for an Expert System application, judgement must be used in establishing the scope of the system. In general it is better to err on the side of too narrow a scope rather than on too broad a scope. If the scope ultimately turns out to be too narrow, the formalism of the standard Expert System model makes it relatively easy to broaden the scope by adding more knowledge to the Knowledge Base.

4.3 IDENTIFY THE EXPERT

Once a problem domain has been identified and the initial effort at narrowing the scope of the Expert System application completed, the expert whose expertise will be modeled in the Expert System must be selected.

The two main criteria that should be used to identify the expert(s) are:

* Is the candidate an expert in solving problems in the problem domain of interest and is he or she recognized as such by the potential user community? The need for the candidate to be an expert in the field is essential for the development of the expert system. The need for the expert to be recognized as such by the potential user community is primarily useful in selling the potential users on the viability of the given Expert System as a useful problem solving tool for them.

* Is the expert available and willing to spend the time (perhaps months) that will be required to build, test and field a working Expert System? The expert must be dedicated to the successful development, testing and evaluation, and implementation of the system. The failure to identify such a person and obtain a firm commitment means that the Expert System project should not be undertaken.

Other useful (but not necessary) characteristics for the domain expert to have include:

* An ability to effectively communicate
* An orderly mind
* Lots of patience
* A willingness to teach

4.4 IDENTIFY THE EXPECTED BENEFITS OF THE EXPERT SYSTEM

Prior to embarking upon an Expert System development effort, the expected benefits of such an effort must be clearly defined. There are two categories of benefits that are typically cited as reasons for developing an Expert System. One category consists of concrete, quantifiable reasons such as:

* Saving money (how much)
* Saving time (how much)
* Increasing productivity (how much)
* Enhancing a user's ability to solve a given set of problems (how much)

The other category of benefits consists of tangible but not quantifiable reasons such as:

* The developing of an Expert System can serve as a vehicle for formalizing the knowledge in a given problem domain and thereby make a contribution to the clarification and advancement of knowledge in the domain.

* The developing of an Expert System can serve as a means of combining the expertise from many experts in a given domain and thereby allowing the available knowledge about the problem to be both tested for consistency and synthesized into new and

more useful knowledge about problem solving in the domain.

4.5 KNOWLEDGE ENGINEERING

The Knowledge Engineering phase of the Expert System development process begins shortly after a suitable domain expert has been chosen. The role of the Knowledge Engineer is to build and implement a model of the domain expert's problem solving expertise for the chosen application.

A good Knowledge Engineer should have extensive experience with modeling and with implementing computerized models, as well as being thoroughly versed in the theory and use of Expert System development tools. Ideally, the Knowledge Engineer should also be experienced in systems analysis and system design, and should also have a good knowledge of modern Computer Science concepts such as top-down design, abstraction and data structuring techniques, software documentation and software maintenance. In addition, to successfully reflect the knowledge and reasoning of the expert, the engineer must also be good in expressing concepts in clear and concise English (or the language of the domain expert and the end users) and be proficient with interviewing techniques.

4.6 SYSTEM DESIGN AND DEVELOPMENT

Attempting to replicate expert decisionmaking with a computer requires a carefully planned, yet flexible, design and development strategy. The construction of an Expert System, from initial design to working system, consists of a number of loosely defined steps. The necessary steps are outlined below:

* Problem Analysis ... Once it has been decided that the task is suitable for an Expert System, the next step is to gather the information necessary to implement the system task. Often this step involves interviewing experts in order to gain an understanding of the processes they follow in completing the task. Through the interviews it is necessary to determine major components of the reasoning process and attempt to identify the elements that should be included in the system.

* Initial Prototype ... This step is a natural extension of the problem analysis step. The knowledge initially developed above is implemented in the Knowledge Base. Other features such as customized User Interfaces, explanation facilities, etc. are also developed. A demonstration of the initial prototype should be conducted.

* Expanded Prototype ... The initial prototype should be tested extensively by both the experts who were involved in its development and potential users. This should provide the developer with an assessment of the conclusions, ease of use, gaps in knowledge, areas where the knowledge should be

refined, etc. These findings are then incorporated to produce the expanded prototype. At least 2 demonstrations should be conducted during this phase.

* Delivery System ... Once the final changes have been incorporated into the system and thoroughly tested, it is then appropriate to produce the delivery system. The delivery system differs from the expanded prototype in that it is optimized with respect to performance, memory requirements, User Interface, etc. This may entail implementation of some or all of the modules in a language or development environment different from the one in which it was developed.

One of the advantages of this type of development methodology is that it acknowledges the difficulty in specifying meaningful requirements for Expert Systems. Thus a full set of requirements do not have to be established before development begins. Beginning with an initial set of requirements developed in the problem analysis stage, the requirements can be refined as the prototype is refined. The prototype itself is a statement of the system requirements in terms of interfaces, goals, and principal reasoning paradigms. Thus a description of the prototype can considerably ease the task of writing requirements.

4.7 VERIFICATION AND VALIDATION

In traditional software engineering, verification and validation are an integral part of the design and development process. This is recognized and accepted both by users and developers. In fact, given the sensitive functions that are performed by software today (e.g., electronic banking, air traffic control, manufacturing, etc.), it would be hard to conceive of their absence. However in the developing area of expert systems, there is relatively little discussion of these critical functions. In order for Expert Systems software to achieve the same acceptance as traditional software, these issues must be addressed.

There are many reasons for the current situation. Geissman and Schultz 6/ discuss many of these. They include:

* Expert System problems tend to lead to a "combinatorial explosion" in the number of possible states. This makes exhaustive testing an unattractive, and often infeasible option.

* The User Interface in an Expert System is extremely complex and interactive (as discussed in section 3.3). A computer program with a simple interface could perhaps be "exercised" by another test program. This is much more difficult to do with a complex, interactive interface.

* Liskov and Guttag 11/ state that the "purpose of the

requirements analysis phase is to analyze the needs of the customer and produce a requirements specification of a program that will meet those needs." These requirements are difficult to produce for an Expert System (at least in a form sufficiently detailed to test against). It should be noted that it is simple to state that the program should "perform like an expert". However, when attempting to test against this requirement, its fundamental weakness becomes more apparent.

* Green and Keyes 6/ cite "a vicious circle where nobody requires Expert System validation and verification, so nobody does it. Since nobody knows how to do it, nobody requires it."

* Most of the common techniques for tracing program execution flow were developed for procedural languages with the basic control structures of sequencing, branching and looping. The language of Expert Systems is non-procedural with control mechanisms (e.g., backtracking) which are totally alien to the typical programmer based in procedural languages (e.g., FORTRAN, BASIC, COBOL, C, PASCAL, ADA, etc.). Hence tracing program flow from the examination of the program code can be quite difficult.

* The "structured programming" methodology has been very successful using a top-down modular approach to design, implementation, debugging and maintenance. However, just as tracing program flow is difficult in Expert System languages, implementing a top-down modular design may not always be achievable.

As discussed by Geissman and Schultz 6/, bits and pieces of a verification and validation methodology currently exist, but have not been assembled and standardized due to the many applications, design paradigms, development approaches, and the stage of development and fragmentation of the industry.

Liskov and Guttag 11/ state that "... verification involves reasoning about program texts. This distinguishes verification from testing, which is always based upon observing computations. In verification we examine the program text..." Issues raised during verification include:

* Does the high-level design reflect the requirements? Are all of the issues raised during the requirements addressed in the high-level design?

* Does the detailed design reflect the high-level design?

* Does the code accurately reflect the detailed design?

* Is the code correct with respect to the language syntax?

When the program has been verified, we are assured that it has no "bugs".

Validation is a determination that the completed program performs the functions in the requirements specification and is usable for the intended purposes; i.e., the program is doing the job it was intended to do. It is virtually impossible to have an ironclad guarantee that a program satisfies its specification, so one settles for having some degree of confidence that a program is valid. Issues addressed during validation of an expert system include:

* How well do inferences made compare with historic (known) data?

* What fraction of pertinent empirical observations can be simulated by the system?

* What fraction of model predictions are empirically correct?

* What fraction of the system parameters does the model attempt to mimic?

There is no set formula or algorithm for verification and validation. However, in addition to the guidelines stated above, there are some things which can be done to make this process more effective. First of all, the developer should design for testing. One way to do this is to consistently adhere to some problem solving paradigm, (e.g., forward chaining and backward chaining are the two most common), as much as possible. The advantage of this is that once this paradigm is specified, then a great deal of verification can be done through code walk-through and examination. In traditional software engineering, adherence to certain principles, (e.g., single entry, single exit modules), restrict what features of the language the developer may choose to use. In the same way, adherence to a given paradigm may restrict or limit desirable features of an expert system shell.

In addition to the point above, the developer should also certify the Inference Engine. This does not imply that one should necessarily be suspicious of a shell's claims. However, one can have two "forward chaining production systems" that give different results with the same data. Again when consistently using a given paradigm and certifying that the Inference Engine is indeed following this paradigm, then verification can be done much easier.

Verification and validation are not fixed steps in the development process. Instead they should be thoroughly integrated into development from the very beginning and remain on-going. The near future should see a more formal approach taken as the expert systems field matures.

4.8 SOFTWARE TOOLS

Expert systems development tools or "shells" are available to aid Knowledge Engineers (and others who play the role of Knowledge Engineer) in the development of Expert Systems, a number of software vendors have developed Expert System development shells. These shells are in essence Expert Systems without the Knowledge Bases. These shells typically provide a built-in User Interface and a built-in Inference Engine. When using one of these shells to build an Expert System all the user need provide is a model of some domain specific problem solving expertise. Most of these shells provide software tools for coding this model into a form that is compatible with the Knowledge Base representation that the shell uses.

4.9 DISTRIBUTING AND MAINTAINING EXPERT SYSTEMS

Once the Expert System development effort has been completed, the tasks of distribution and maintenance begin. Although there are no fixed rules governing these tasks, there are some general guidelines which can make these tasks easier to perform.

4.9.1 Distribution

There are several major criteria that a developer should follow in order to facilitate the distribution of a given Expert System.

* Identify the potential user community prior to undertaking the development of an Expert System. This should insure that the Expert System actually solves a real set of problems and thereby, make it easier to sell the designated user community on the suitability of the tool.

* Develop the system using standard hardware and software. Although there are a number of exotic pieces of Expert System hardware and software, the cost of these items is often so high that it is unreasonable to expect potential users to procure them just to use an Expert System. As a rule of thumb, any Expert System that is developed in the near term should run on industry standard personal computers using MS-DOS.

* Use development software that does not require distribution licenses or where an unlimited distribution license can be purchased for a reasonable fee. There are a number of Expert System development shells that require that a developer pay a fee for each system that is distributed to a user. If the user community is large or if it is expected that several future versions of the system will be developed, the amount of money spent on user fees can become exorbitant.

* Distribution must be accomplished as specified in Chapter X,

Expert Systems, of the Information Resources Management Manual.

4.9.2 Maintenance

The task of Expert System maintenance is one that should be planned for from the inception of the Expert System development project. Maintenance can be facilitated by following a few good development rules. These include:

* Make sure that the Expert System design is as transparent as possible. Since the maintenance phase will probably not be handled by system designers, it is important that the structure of the Expert System be as straightforward and clear as possible. The developers should avoid the use of cryptic names for objects and knowledge structures within the system. The developers should also avoid the use of overly complex and obscure software structures, even though their use may provide some type of performance benefits. One of the guiding principals in the development effort should be "keep it simple."

* Make sure that the system is well documented. This documentation should be produced as the system is developed, not after it is finished. The Knowledge Engineer should clearly identify where the system's knowledge resides (e.g., in the Knowledge Base in the form of facts and rules and in the Inference Engine in the form of heuristic search techniques). The Knowledge Engineer should also document the inference procedures that the system uses in producing solutions. There must also appear as part of the documentation an explicit model of the problem solver that exists within the Expert System. The documentation should also provide a comprehensive and well documented test procedure for the system. In addition,the Expert System itself should contain an extensive set of both user "Help" text and text which explains how the Expert System produced a given solution. These latter two items should be produced during the development phase and not added after the system has been built. One of the guiding principles that developers should use is "a poorly documented system will have a short useful life."

* Each version of a given Expert System should have a version number. This will make it easier to provide users with up-dated copies of the system.

* Establish a mechanism for soliciting, receiving and acting upon feedback from the user community. This will facilitate the identification and removal of "bugs" in the system and will also make it easier to retro-fit the system to satisfy specific user community needs after the system has been

fielded.

* The above items must be complied with to meet the intent of Chapter X, Expert Systems, of the Information Resources Management Manual.

REFERENCES

1. Barr, Avron & Feigenbaum, Edward A.
   The Handbook of Artificial Intelligence, Volumes 1,2
   William Kaufman, Los Altos, California (1981)

2. Brownston, Lee et al
   Programming Expert Systems in OPS5
   Addison Wesley, Reading, Massachusetts (1986)

3. Cercone, Nick & McCalla, Gordon
   The Knowledge Frontier
   Springer Verlag, New York, New York (1987)

4. Frost, Richard
   Introduction to Knowledge Based Systems
   Macmillan Publishing, New York, New York (1986)

5. Galambos, James A. et al
   Knowledge Structures
   Lawrence Erlbaum Associates, Hillsdale, New Jersey (1986)

6. Geissman, James R. & Schultz, Roger D.
   "Verification and Validation of Expert Systems"
   AI EXPERT, Vol. 3, No. 2, Feb. 1988

7. Haton, J.P.
   Fundamentals in Computer Understanding: Speech and Vision
   Cambridge University Press, Cambridge, England (1987)

8. Hayes-Roth, Frederick et al
   Building Expert Systems
   Addison Wesley, Reading, Massachusetts (1983)

9. Klahr, David et al
   Production Models of Learning and Development
   The MIT Press, Cambridge, Massachusetts (1987)

10. Kowalik, Janusz S.
    Knowledge Based Problem Solving
    Prentice-Hall, Englewood Cliffs, New Jersey (1986)

11. Liskov, Barbara & Guttag, John
    Abstraction and Specification in Program Development
    The MIT Press, Cambridge, Massachusetts (1986)

12. Minsky, Marvin
    Semantic Information Processing
    The MIT Press, Cambridge, Massachusetts (1968)

13. Nilsson, Nils J.
Principles of Artificial Intelligence
Morgan Kaufmann, Los Altos, California (1980)

14. Pearl, Judea
Heuristics
Addison Wesley, Reading, Massachusetts (1984)

15. Shapiro, Alan D.
Structured Induction in Expert Systems
Addison Wesley, Wokingham, England (1987)

16. Winston, Patrick H.
Artificial Intelligence
Addison Wesley, Reading, Massachusetts (1977)

☆U.S. Government Printing Office : 1989 - 617-000/88639

DOT-T-89-11

TECHNOLOGY SHARING

A Program of the U.S. Department of Transportation