TOOLS AND TECHNOLOGIES FOR EXPERT SYSTEMS A HUMAN FACTORS PERSPECTIVE

Final Report

NASA/ASEE Summer Faculty Program-1987
Johnson Space Center

Prepared by:	Navaratna S. Rajaram
Academic Rank:	Associate Professor
University Department:	University of Houston- University Park Department of Industrial Engineering Houston, TEXAS 77004

NASA/JSC

Directorate:	Space and Life Sciences Directorate
Division:	Man-Systems Division
Branch:	Crew Station Branch
Section:	Crew Interface Analysis Section
JSC Colleague:	Marianne Rudisill, Ph.D.
Date:	August 14, 1987
Contract Number:	NGT 44-001-800

ABSTRACT

It is widely recognized that technologies based on artificial intelligence, (AI) especially expert systems can make significant contributions to the productivity and effectiveness of operations of information and knowledge intensive organizations such as NASA. At the same time, these being relatively new technologies, there is the problem of transfering technology to key personnel of such organizations. This report addresses itself to the problems of examining the potential of expert systems and of technology transfer in the context of human factors applications.

One of the topics of interest was the investigation of the potential use of expert system building tools, particularly NEXPERT as a technology transfer medium. Two basic conclusions were reached in this regard. First, NEXPERT is an excellent tool for rapid prototyping of experimental expert systems, but not ideal as a delivery vehicle. Therefore, it is not a substitute for general purpose system implementation languages such as LISP or C. This assertion probably holds for nearly all such tools on the market today. Second, an effective technology transfer mechanism is to formulate and implement expert systems for problems which members of the organization in question can relate to. For this purpose, the **LI**ghting **E**n**G**ineering **E**xpert (LIEGE) was implemented using NEXPERT as the tool for technology transfer and to illustrate the value of expert systems to the activities of the Man-Systems Division.

0. INTRODUCTION AND STATEMENT OF PROBLEMS

Introduction

It is unlikely that there ever will be a universally satisfactory and at the same time comprehensive definition of artificial intelligence (AI). However, for purposes of the present document AI will be defined as the discipline devoted to the study of human reasoning capabilities. In particular, AI is concerned with formalisms for human problem solving and decision making. It can be said that to some extent all computer programs do some decision making and problem solving. What distinguishes AI programs is their ability to reason with *symbolic data.* For instance, AI programs can manipulate algebraic expressions in symbolic form, process textual information in English, reason with linguistic and other grammatical rules, use strategic information in games of skill such as chess and these are among systems that fall under the scope of AI.

Among the branches of AI expert systems have captured the greatest attention, among professionals as well as with the general public. If AI is defined as the study of problem solving and decision making, an expert system can be defined as a computer program which can solve problems and make decisions in a specific domain. In particular, there exist expert systems which can diagnose diseases, design electronic circuits, locate faults in telephone cable networks, and perform other similarly skill demanding tasks. It can thus be seen that despite their relatively recent origins, expert systems have already addressed a fairly wide range of applications. In spite of their diversity, expert systems exhibit certain characteristic features which can be summarized as follows.

1. Expert systems are *knowledge based;* i.e., an expert system achieves its often impressive problem solving power from specialized knowledge, rather than from any general form of reasoning skill.

2. An expert system has no pre-programmed algorithm built in. On the other hand, it has to select and assemble a sequence of "elementary units of knowledge" which usually depends upon the particular problem case at hand.

These and other related issues will be elaborated upon in later sections.

Statement of Problems

The research reported herein was intended to address the following specific issues, all of them of interest to the personnel of the Man-Systems Division of NASA Johnson Space Center.

1. **Expert systems for man-systems applications.** To invstigate the relevance and usefulness of the technology of expert systems within the broad context of human factors and human-machine systems, *with the help of concrete examples drawn therefrom.*

2. **Approaches and tools.** To evaluate the scope, advantages and limitations of different approaches and tools used for expert systems. In particular, to study the use of the expert system tool NEXPERT.

3. **Technology transfer.** To use problems of interest in the man-systems area as the medium for technology transfer, with the eventual goal of enabling expert systems to be integrated into such activities.

4. **Future needs and objectives.** To assess the future needs and objectives in man-systems applications as regards tools, environments, and applications in the context of relevant problems.

Each of these issues will be described in the remainder of this report.

Conclusions

Two basic conclusions were reached which can be expected to have some bearing on future activities of the Man-Systems Division. Because of the importance which NASA attaches to AI and expert systems, they are discussed in some detail in the last section of this report. But those conclusions can briefly be summarized as follows.

1. AI and expert systems technologies can make a significant contribution to human-systems tasks and activities.

2. NEXPERT is an excellent tool for building experimental systems.

1. APPROACHES AND TOOLS

1.1 Structure of Expert Systems

As previously noted, expert systems are *knowledge based;* i.e., they solve problems by making inferences using *specialized knowledge* in a particular domain. This knowledge is usually expressed in the form of IF... THEN... rules. Such rules are called *production rules* in the technical literature. Thus, every expert system must have the following two kinds of knowledge.

- o *Domain knowledge* usually expressed in the form of production rules.

- o *Method knowledge or inference knowledge* about how to use domain knowledge.

It is convenient to think of domain knowledge as "know-what" and method or inference knowledge as "know-how."

In order for an expert system to function as a complete integrated system, it needs the following components.

1. Domain knowledge.

2. Inference mechanism.

3. A suitable computer representation of the problem state; i.e., a *data structure* representation of the problem. The usual term for this data strucure is *working memory.*

4. User interface which has the responsibility of converting the user stated problem statement into a computer readable data structure, i.e., the initial working memory.

Figure 1 shows the structure and organization of an expert system schematically.

The working of an expert system is best described in terms of a *cycle.* A cycle in an expert system consists of the following sequence of steps.

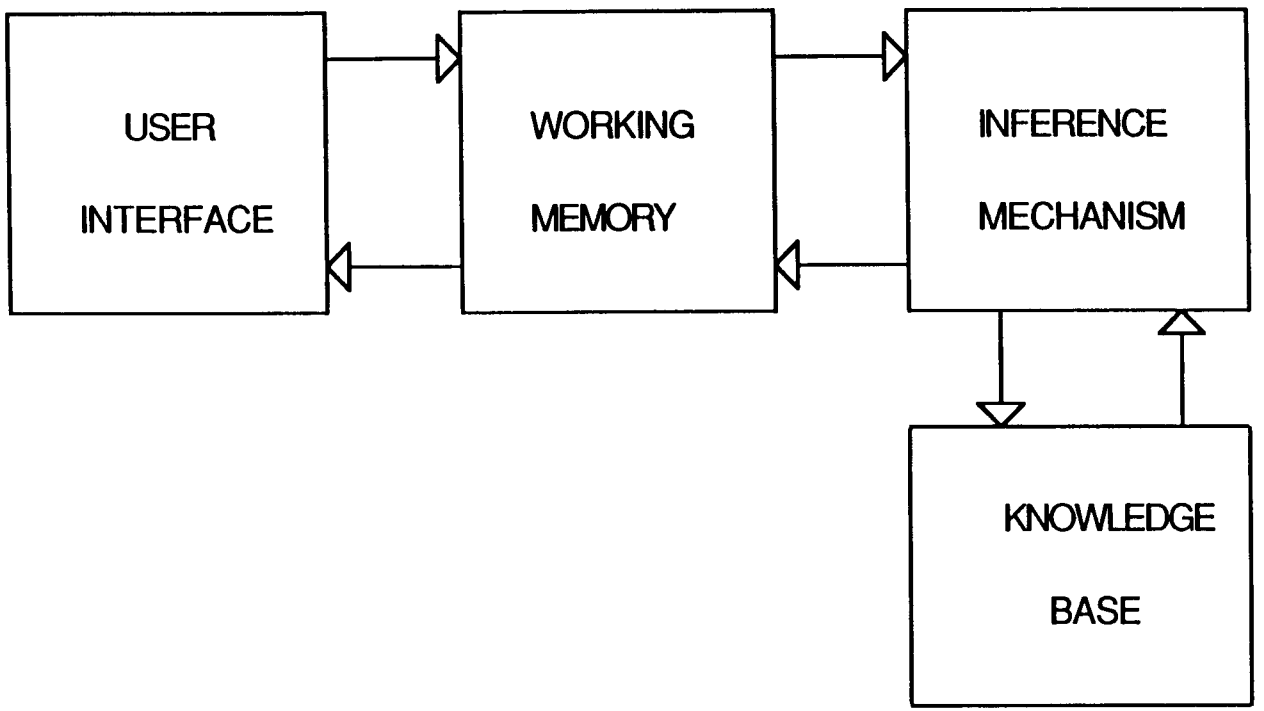

Figure 1: Organization of an expert system

1. User inputs the problem statement via the user interface. The user interface converts this input into a data structure which becomes the initial working memory.

2. The inference mechanism selects a rule from the knowledge base and applies it to the working memory. This rule application changes the working memory.

3. The inference mechanism checks to see if this changed working memory corresponds to a solution state. If it does, the system reports the solution back to the user via the user interface. If it does not correspond to a solution state, the inference mechanism selects another rule and the cycle repeats.

This is an oversimplified description of the problem solving process in most expert systems. Invariably there are factors such as search efficiency and other related matters which have to be taken into consideration in even moderate sized systems. Nevertheless, it can be said that the problem solving mechanism in an expert system essentially consists of searching through the knowledge base to find the right sequence of rules in the context of the working memory.

There are basically two ways of carrying out inference in an expert system-- *hypothesis driven* and *data driven.* The hypothesis driven inferencing is also known as *backward chaining*, while the data driven approach is called *forward chaining.* In the hypothesis driven approach, the system postulates a solution and then checks to see if the working memory justifies the hypothesis. In the data driven approach, the reverse approach is taken, i.e., a hypothesis to match the working memory is searched for. Recalling that production rules in the knowledge base have the following structure, the terms forward and backward chaining are obvious.

IF <data> THEN <conclusion (or hypothesis)>

1.2 Expert System Tools: NEXPERT

Design and implementation of an expert system to work in a specific application domain calls for the integration of the following kinds of knowledge.

1. Knowledge of the problem domain or domain expertise.

2. Knowledge of the expert system design methodology, at least in the context of the problem(s) of interest.

3. Knowledge of the implementation medium, i.e., the hardware/software environment.

There are basically two choices in the the selection of implementation software.

- o A general purpose AI oriented language such as LISP or Prolog.

- o A special purpose expert system building tool such as NEXPERT.

But no matter which approach is taken, the software and the implementor(s) must be able deal with all three of the above mentioned parts of building an expert system. In addition, the user interface part of the system might become the critical component when a system has to be delivered to end-users. This report is primarily concerned with the use of the high level expert system tool NEXPERT. The relevance of general purpose languages such as LISP is discussed in the next section. Issues relating to user interfaces do not form a central concern of this research and will merely be touched upon in the final section.

Description of NEXPERT

NEXPERT is a fairly typical expert system building tool, its distinguishing feature being a quite flexible inference engine. Unlike several other tools on the market, NEXPERT permits the user to use either forward or backward chaining, or even a combination of the two. As is the case with nearly all such tools, NEXPERT is a rapid prototyping tool for building experimental expert systems. Unfortunately, like most pioneering software products, its documentation and instructions leave something to be desired, being often vague and at times downright misleading. Only some of the technical features of NEXPERT as they relate to the problem of interest are discussed here. A general introduction to the use of NEXPERT for implementing expert systems is given in a separate document. (See "An Introduction to Using NEXPERT.")

As previously noted, any tool must provide facilities for implementing the following parts which go to make an expert system: the workingmemory, the knowledge base, and the inference mechanism. With NEXPERT, the structure of the working memory is *implicit;* i.e., it is created when the rules making up the knowledge base are defined. In addition, the user has the freedom to choose any kind of inference, even with the same knowledge base. Thus, the primary focus of NEXPERT, at least from the user's point of view has to be on the use of its syntax for expressing production rules. Details of NEXPERT syntax are discussed elsewhere. (op. cit.) It suffices to note here that the following steps are involved in building the knowldge base in NEXPERT.

1. Analysis of the problem.

2. Formulation of the problem in the format of production systems.

3. Expressing the problem solving knowledge as production rules.

4. Coding these rules in NEXPERT.

A point to be noted is that neither NEXPERT nor any other software has the design knowledge needed to build an expert system. That knowledge has to be part of the system designer's expertise. This point needs some emphasis since not infrequently one hears such claims being made on behalf of expert system building tools, including NEXPERT. Such however is not the case.

1.3 Use of General Purpose Languages

As previously noted, any general purpose language can be used as a medium for building expert systems. The choice of the language is nearly always determined by the needs of the environment in which the finished system is required to run. If the expert system under consideration is required to interface with existing databases and other external files, it is best to choose the implementation language with that consideration in view. As is invariably the case with information systems (of which expert systems are a special case) it is the structure of the data that has to interface to the problem which dictates the design including the choice of the language.

This basic fact, namely that data are by far the most important part of any information system should decide the choice of the language for expert systems also. In the latter case, this requirement translates into defining the strcture and the needs of the working memory which is nothing more than a computer representation of the problems of interest.

Once this design and implementation decision is made, the steps involved in building an expert system are the following.

1. Choice of the data structure needed to represent the problem states.

2. Formulation of the knowledge in the format of production rules.

3. Choice of the appropriate inference mechanism(s).

4. Coding of the production rules and the inference mechanism.

As can be seen, there is some additional work involved in using a basic language such as LISP instead of a special purpose expert system building tool such as NEXPERT. But the additonal work involved in coding the inference mechanism should not be exaggerated. On the other hand, there is greater flexibility in the choice of data structures for the working memory, thereby ensuring that the tool does not limit the choice of applications. In particular, existing tools can almost never be used for delivering finished expert systems to end-users. These are among the factors which have to be taken into consideration in choosing a particular language or tool for implementation. Summarizing, it can be said that NEXPERT, like others in its genre is suitable for building experimental expert systems, but not as a delivery vehicle.

2. CASE STUDY: AN APPLICATION IN HUMAN FACTORS

The objective of the case study chosen was twofold. First, to explore and demonstrate the use of the expert system technology, and in particular, the usefulness of the tool NEXPERT for building experimental expert systems. Second, to use this particular case study as a means for technology transfer to the Man-Systems Division in the sense to be described later.

2.1 Description of Expert System LIEGE

LIEGE (**LI**ghting **En**G**i**neering **E**xpert) was the expert system designed and implemented using NEXPERT to demonstrate the feasibility of both the technology and the tool (NEXPERT) for practical problems of relevance to human factors. In particular LIEGE recommends the appropriate light intensity in international units (Lux) for a specific task in a specific area of interest. Thus, LIEGE has the following capability.

Input: TASK AREA
TASK DESCRIPTION

Output: RECOMMENDED INTENSITY

Limitations of space preclude the inclusion of a detailed description in the present document. Such a description can be found elsewhere. (Ref. 1) Here, it suffices to know that the knowledge base in NEXPERT is organized according to the taxanomy given in table 1.

LIEGE uses a forward chaining or data driven inferencing mechanism. The "data" input to LIEGE are AREA and TASK which are chosen from the options menu which forms part of its knowledge base. Thus the user does not have to know which are the permissible inputs and which are not. The user however does need to know how to load the corresponding knowledge base which has been called LIEGE1.KB.

Intensity is only one of the parameters which go into the lighting design task. LIEGE has been designed with possible future extensions in mind. When fully implemented, its knowledge base will be according to the structure in figure 2.

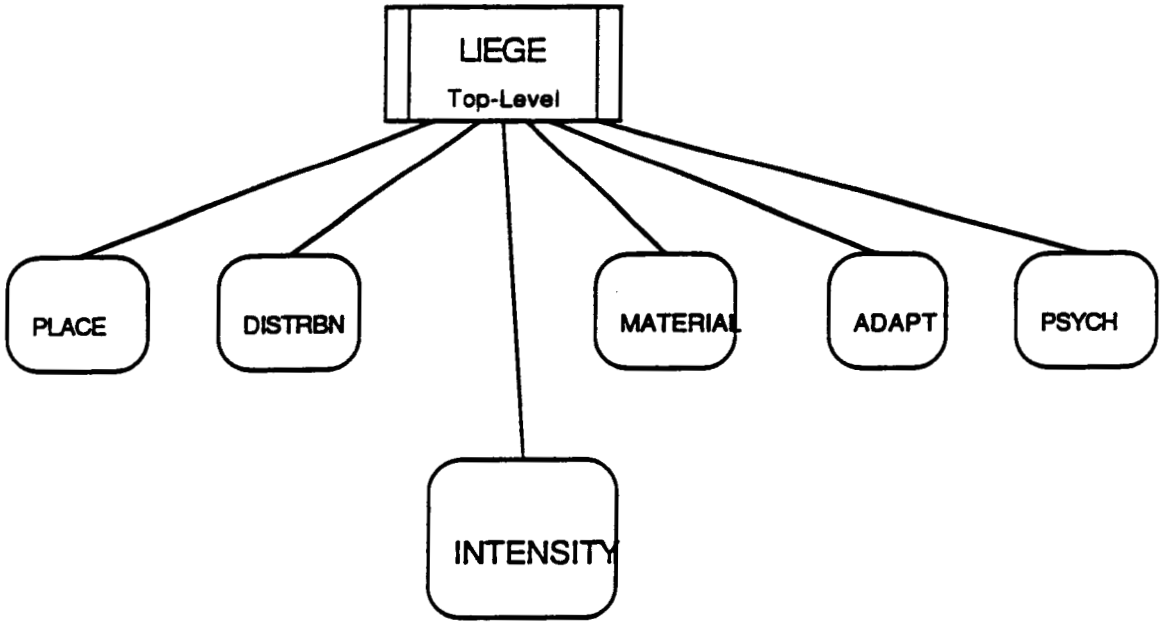

Figure 2: Potential modules in lighting design

Table 1

AREA	TASK
General	All (i.e., ambient lighting)
Translation paths	Hatches Handles Ladders
Wardroom & galley	Reading Recreation Dining Food preparation
Personal hygiene	Grooming Shower Health care
Waste management	All
Crew quarters	Reading Emergency lighting
Health maintenance	First aid Surgical I.V. treatment Exercises
Emergency lighting	Unspecified

As previously noted, LIEGE is a production system. Thus the taxanomy given in table 1 has to be converted into production rules and then encoded in NEXPERT. Here is an example rule.

IF *AREA is health maintenance*
and TASK is surgical
THEN *INTENSITY 920 lux is confirmed*

2.2 Application Development for Technology Transfer

At this time, expert systems is a technology in transition; i.e., it is a technology whose potential for contributing to the efficiency and productivity of knowledge intensive organizations is recognized, but is yet to realize that potential. A basic problem in implementing expert systems for specific applications is what is known as the *knowledge acquisition bottleneck*.

Knowledge Acquisition Bottleneck

From the discussion so far, it is evident that designing and implementing an expert system for a specific application calls for the analysis of the problem in fine detail, and then expressing the skills needed to solve the problem, i.e., *domain knowledge* in the format of AI, in particular as production rules. At this time, the great majority of domain experts lack design knowledge about expert systems. As a result, there exists a communication gap between technologists who know how to build expert systems, and application specialists with domain knowledge, i.e., between those with "what knowledge" and those with "how knowledge." This is known as the *knowledge acquisition bottleneck.* This has to be addressed whenever we want to exploit a new or emerging technology such as AI. Once the existence of this bottleneck is recognized, one can take either of the following two approaches in attempting to overcome it.

- Familiarize knowledge engineer with the domain knowledge to the point that he/she acquires sufficient depth in the latter turn this knowledge into expert systems; i.e., turn the knowledge engineer into a domain expert.

- Train the domain expert in the use of tools and techniques of building expert systems; i.e., turn him/her into a knowledge engineer capable of generating expert systems from his/her own domain knowledge.

Experience has shown that the second approach is by far the more effective and realistic. In this approach, an AI specialist functions as an instructor and a consultant on specific problems for which expert systems are being built by domain experts. This was the approach followed during

the present project as the means of technology transfer. Obviously, such technology transfer has to take place before any organization can efffectively use a new technology like AI and expert systems. Thus, the following course appears to be the most productive and at the same time the most pragmatic means of technology transfer.

- o Identify problems of interest within the organization, problems which the personnel in the organization can relate to.

- o Train and assist domain specialists in these areas to turn part of their expertise into expert systems. This calls for the assistance of at least one experienced knowledge engineer, but the "leverage" from this expertise will be high.

The second of these approaches was followed during the project reported here. This was greatly assisted by the lighting design problem which workers in the division could relate to. Suggestions for furthering this initiative are made in the next section.

3. CONCLUSIONS AND SUGGESTIONS FOR THE FUTURE

The goals set at the beginning of the present project were met or exceeded. Nevertheless, the project is best regarded as the beginning of a new direction rather thas as the culmination of an effort. With this perspective, the following conclusions and recommendations are presented for future reference.

Conclusions

1. NEXPERT is an excellent tool for building experimental expert systems, especially useful for rapid prototyping. The learning period needed to put NEXPERT to practical use is quite short. Further, its versatile inference facilities make it especially attractive.

When all these factors are taken into consideration, NEXPERT is at least as good as any other tool on the market, even though many of them cost several times as much as NEXPERT and require expensive and hard to use special purpose workstations.

2. At the same time, NEXPERT should be seen as a tool for building experimental systems and prototypes rather than as a delivery vehicle. In this regard, it is not a substitute for LISP or C.

By this observation, it is not at all being suggested that NEXPERT is "worse" than LISP, Prolog or C for implementing expert systems, but that it is a completely different vehicle, being a special purpose tool rather than a general purpose language. In other words, it is a case of the proverbial comparison between "apples and oranges."

3. There are many problems which fall under the category human-machine systems which can potentially benefit from the infusion and integration of AI and expert systems technologies. One of the more intriguing and potentially useful applications is the area of expert system monitoring and design of crew safety.

Suggestions for the Future

The conclusions just presented suggest possible options for the future. In particular, they indicate what needs to be done so that AI and expert systems can move beyond the experimental stage and begin to make material contributions to the relevant operations and tasks at hand.

1. Additional problems in the human-machine systems area should be identified and formulated in the framework of AI and expert systems.

2. Tools such as GCLISP and GoldWorks which enable expert systems to be built and delivered for end users should be used for the purpose so that actual systems can be delivered and their benefits evaluated.

The steps just outlined can make a significant contribution towards incorporating this new technology into existing operations. In this regard, general purpose tools such as LISP and C can be used to build and deliver systems to end users. These tools should be chosen with the specific goals in view; i.e., the application should drive the tool selection and not the other way, as is too often the case. In additon, the following technology alternatives are worth examining for possible future use.

1. Advanced computer architectures for learning machines, such as the ACA and the Connection Machine architectures.

2. Object-oriented-programming for general purpose prototyping and for building integrated user environments.

3. Expert systems with "linguistic" rule selection formalisms for dealing with uncertain situations.

The final technology option above is a crucial advance in which basic research is currently being done and has considerable future potential for the design of more flexible systems. In particular, systems with such linguistic capabilities instead of simple pattern match in the inferencing stage can lead to systems a step beyond the current state of the art. Such expert systems can deal not only with uncertain situations, but also with unforeseen circumstances, possibly even exhibiting some "common sense."

The goal of common sense reasoning by machines has so far proved somewhat elusive. It is the belief of the present author that the linguistic approach just outlined offers an alternative to logic based approaches currently being investigated. Although this is clearly a basic research area, it seems likely that such research might yield practical benefits surprisingly soon. This and other technologies are worth exploring for potential benefits.

Acknowledgements: The author is grateful for the opportunity for working with the Man-Systems Division, particularly to Dr. Marianne Rudisill. He is also grateful to Mr. Charles Wheelwright for his generous help with the knowledge base for LIEGE. Finally, he is grateful to Ms. Jeri Brown for generous help with the resources and support.

4. REFERENCES

1. Rajaram, N.S. (1987) *An Introduction to Using NEXPERT,* NASA Johnson Space Center (In preparation).

2. Rudisill, M.L. and Burns, M.J. (1986) *Human Factors of Expert Systems,* NASA Johnson Space Center.

3. *NEXPERT Fundamentals,* Neuron Data Inc., 1985.

4. *Man-Systems Integration Standards vol.1,* NASA-STD-3000, Johnson Space Center.

5. Charniak, E.J. and McDermott, J. (1984) *Introduction to Artificial Intelligence,* Addison-Wesley.