**A Primer On Expert Systems For Well Log Analysis**

E. R. (Ross) Crain, P.Eng.   
D&S Petroleum Consultants / Alberta Research Joint Venture  
Calgary  Alberta  Canada

Published in CWLS Journal, March 1986. This electronic version created 15 Aug 2002. 

*Author’s Note: This paper is still relevant, as most modern AI papers in the petroleum industry demonstrate small, single-purpose expert systems that illustrate the principles described here. This work plus the LOG/MATE ESP development led to the LOG/MATE ESP ASSISTANT and INTELLOG software packages at D&S. ERC Jan 2005.*

**Abstract**This paper reviews the state of the art in expert systems, with special reference to well log analysis and related petroleum applications. Work along these lines is being carried out by a number of organizations, and what is known about these projects is described here. In addition, the terminology, methods, and limitations of expert systems are discussed to provide an adequate understanding of the subject for managers and potential users of such systems. A description of a prototype expert system for log analysis, under development by the author and his colleagues, is presented.

**Introduction**An expert system for log analysis will enable a technician to perform complex analyses which, in the past, could only be done with the assistance of a human expert. In addition, any interpretation, whether by expert or technician, would require less work to provide more complete analysis results. Further, it will allow experts to share and consolidate their knowledge and experience, for use by all analysts with access to the system.

The thrust of the research in this field is aimed at improving the use of existing computer aided log analysis systems as well as creating entirely new systems.

Successful well-log analysis is an acquired skill which is very dependent upon the experience of the analyst. The knowledge which an analyst brings to bear on a specific problem is very specific to the region being analyzed, and therefore a considerable amount of local knowledge is required for successful analysis. Much of this knowledge is available from published literature and from archives of previous work. This information is termed the knowledge base of an expert system.

A further step involves extracting analysis rules and methodology from an expert in log analysis. Rules are usually of three types: usage rules which dictate which method is the best choice for a given data set in a given area, parameter selection rules, and "what if?" or iterative rules for trying alternative methods or assumptions if results are not acceptable on the first attempt.

The knowledge base will be an integral part of future advanced well-log analysis systems. The rule base is an attempt to realize a quantum step forward in this field and contains a significant element of risk. Success is not guaranteed.

**What Is An Expert System ?**Expert systems and artificial intelligence are not knew concepts.

Researchers have worked to develop artificial intelligence since the early 1950's for a number of reasons. One is to help understand the human thinking process by modeling it with computers. Another is to make better computer hardware by modeling the computer more closely after the human brain. More achievable goals, such as making computers act more human or easier for humans to use, are also part of the AI spectrum, as are robotics and pattern recognition or artificial vision. Natural language understanding, automatic translation, and automatic computer programming are other aspects of artificial intelligence.

In the petroleum industry, well log analysis, property evaluation, reservoir simulation, drilling operations, and geologic interpretation have been attacked with AI techniques. Only limited forms of geologic interpretation, log analysis and drilling hydraulics have received any significant attention, however.

Until a few years ago, these topics were buried in the academic research environment. Now robots, expert systems for computer configurations and dipmeter analysis, as well as many consultative tasks such as medical diagnostics, are available commercially from the AI community. One pundit once explained that "If it works, it's not AI". This is no longer true.

The distinctions between conventional programming, intelligent programming, and artificial intelligence are not hard and fast. Conventional programming uses procedural languages such as Basic or Fortran to create sequential code to solve explicitly stated problems. Intelligent programming goes one step further. Here data bases are used to hold much of what would otherwise be hard code. As a result, the system is much more flexible, and program sequence or content can be modified at will by the user, as can the knowledge contained in the numeric and algorithmic sections of the data base.

Artificial intelligence software uses a process called symbolic processing instead of linear processing of variables in sequence. Although conventional computing uses symbols (variables) in describing the program, the symbols are not really manipulated by the operating system to create new symbols, relationships, or meanings. In artificial intelligence, new relationships between symbols will be found, if they exist, that were not explicitly stated by the programmer. In addition, symbols without values can be propagated through the relationships until such time as values become available, again without help from the programmer. Anyone who has had a divide by zero error while testing a program will appreciate this feature.

One of the most economically attractive facets of AI is expert systems development. Expert systems apply reasoning and problem solving techniques to knowledge about a specific problem domain in order to simulate the application of human expertise. Expert systems depend on knowledge about the particular specialty or domain in which they are designed to operate. The knowledge is provided by a human expert during the design and implementation stage, hence the name expert system. Such programs most often operate as an intelligent assistant or advisor to a human user.

The term expert system sometimes has unhappy connotations, such as a computer that is smarter than a human, so the phrase knowledge based system may be used instead. I believe the human ego is strong enough to withstand the label expert system when applied to a computer program.

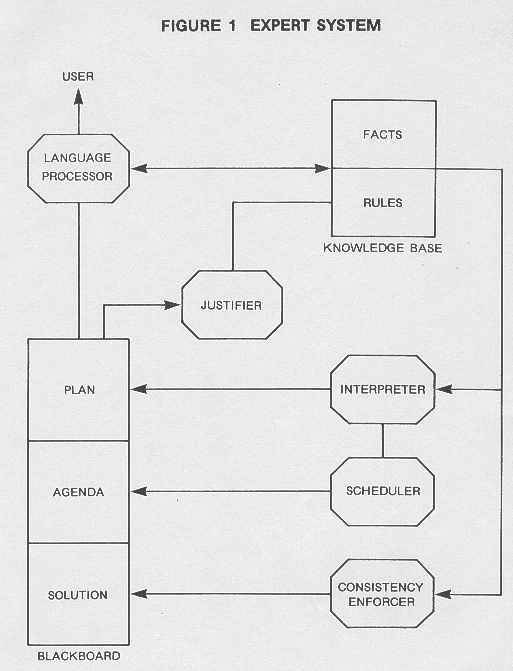
Edward A. Feigenbaum, a pioneer in expert systems, states: "An expert system is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of the field."

Thus, an expert system consists of:

1. A knowledge base of domain facts and heuristics associated with the problem,

2. An inference procedure or control structure for utilizing the knowledge base in the solution of the problem, often called an inference engine,

3. A working memory, or global data base, for keeping track of the problem status, the input data for the particular problem, and the relevant history of what has been done so far.



*Figure 1 shows a block diagram of an idealized expert system.*

The knowledge in an expert system consists of facts and heuristics. The facts consist of a body of information that is widely shared, publicly available, and generally agreed upon by experts in a field. The heuristics are mostly private, little discussed rules of good judgment that characterize expert-level decision making in the field. The rules may be difficult for the expert to verbalize, and hence are difficult to elicit or share. Some facts and/or heuristics may be proprietary to the user or user's organization, and are thus not shareable outside the organization.

In fact, one of the major uses of expert systems in business is to capture a corporation's overall knowledge base as embodied in the brains of their senior technical and executive staff. The rationale is that the expert system will not retire, get sick, die, or take trade secrets to a competitor.

As an example, the facts in an expert log analysis system are the known properties of rocks and fluids. The heuristics include mathematical rules such as Archie's water saturation equation, as well as usage rules which describe when this equation might be used in achieving the desired results. The inference engine in a conventional log analysis program is the procedural code created by the programmer. It can make only limited, predetermined types of decisions, and cannot reason or show why it took a particular path. An expert system overcomes these drawbacks to conventional programming.

When the domain knowledge is stored as production rules, the knowledge base is often referred to simply as the rule base, and the inference engine as the rule interpreter. It is preferable, when describing real problems, to separate the factual knowledge in the knowledge base into a fact or historical data base, and the heuristics on how to use the facts into a rule base. The two data bases, the rules and the facts, comprise the knowledge base. The reason for this is that facts change rapidly in time and space and heuristics evolve more slowly. Thus some logical separation is desirable. However, this terminology might confuse some AI practitioners, unless these definitions are clearly established.

A human domain expert usually collaborates with a knowledge engineer and a programmer to develop the knowledge base. The synergy between these people is important to the success of the project. The performance level of an expert system is primarily a function of the size and quality of the knowledge base that it possesses.

It is usual to have a natural language interface to communicate with the user of the system. Menu driven systems are also practical and offer considerable cost advantages, as well as ease of user training. Normally, an explanation module is also included, allowing the user to challenge and examine the reasoning process underlying the system's answers.

An expert system differs from more conventional computer programs in several important respects. In an expert system, there is a clear separation of general knowledge about the problem from the system that uses the knowledge. The rules forming a knowledge base, for example, are quite divorced from information about the current problem and from methods for applying the general knowledge to the problem. In a conventional computer program, knowledge pertinent to the problem and methods for utilizing it are often intermixed, so that it is difficult to change the program. In an expert system, the program itself is only an interpreter and ideally the system can be changed by simply adding or deleting rules in the knowledge base.

There are three different ways to use an expert system, in contrast to the single mode (getting answers to problems) characteristic of the more familiar type of computing. These are:

1. Getting answers to problems -- user as client,

2. Improving or increasing the system's knowledge -- user as tutor,

3. Harvesting the knowledge base for human use -- user as pupil.

Users of an expert system in mode (2) are known as domain specialists or experts. It is not possible to build an expert system without at least one expert in the domain involved in the project.

An expert system can act as the perfect memory, over time, of the knowledge accumulated by many specialists of diverse experience. Hence, it can and does ultimately attain a level of consultant expertise exceeding that of any single one of its "tutors." There are not yet many examples of expert systems whose performance consistently surpasses that of an expert. There are even fewer examples of expert systems that use knowledge from a group of experts and integrate it effectively. However, the promise is there.

To accomplish this task, an expert system must have a method for recognizing and remembering new facts and heuristics while the system is in use, and for gracefully forgetting those which are inconsistent, incorrect, or obsolete. At the moment, most expert systems require that such changes be made off-line from actual program execution.

**The Knowledge Base**Knowledge representation in the knowledge base is an important aspect of expert system design. The three major forms of knowledge representation are production rules, frames, and semantic sets. The different methods are used for different data types and data uses. Production rules are used where IF...THEN statements define the knowledge adequately. Frames are used to represent descriptive and relational data that cluster or that conform to a stereotype. Semantic sets are most useful for defining classifications, physical structures, or causal linkages.

The most popular approach to representing the domain knowledge needed for an expert system is by production rules, also referred to as SITUATION-ACTION rules or IF-THEN rules. Thus, a knowledge base may be made up mostly of rules which are invoked by pattern matching with features of the problem as they currently appear in the global data base. A typical rule for a log analysis system might be:

IF matrix density is greater than sandstone matrix AND lithology is described as shaly sand THEN suspect a heavy mineral OR cementing agent OR suspect inadequate shale corrections OR suspect poor log calibrations

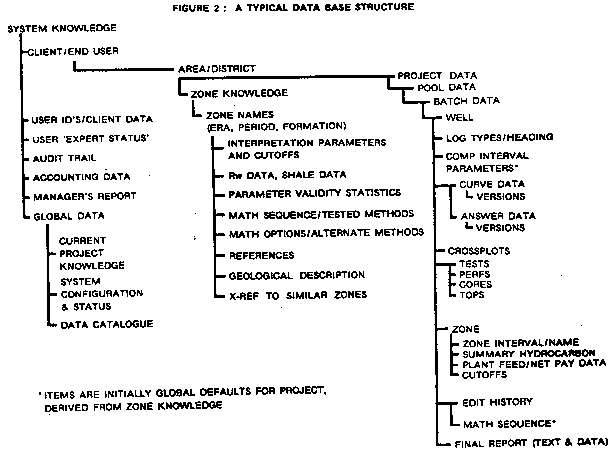
Most conventional log analysis programs contain checks and balances of this type, coded in Basic or Fortran, with appropriate action being dictated by user defined logic switches. The virtue of an expert system knowledge base is that the expert can modify this rule set with comparative ease, compared to a hard coded program. Some programs contain these rules in a user accessible data base, so the same change can be implemented easily also. In this case, the rule must be formulated mathematically, although the output may be a text message similar to the ACTION part of the rule described above.

The knowledge base may also contain large amounts of quantified data or algorithms which help quantify data. In the petroleum industry, such data may represent the physical and chemical properties of rocks and fluids, or costs and income data for different production environments, or predictive equations which quantify empirical and well accepted rules of thumb. Equations which predict porosity from sonic travel time or production rate from exponential decline are well known examples.

In the petroleum environment, it is inconceivable that an expert system could be successful without extensive information of this type in its knowledge base. Much of our rule base consists of empirical rules of thumb which have been quantified by many experts, and used by large numbers of practitioners.

This information can be gleaned from literature search, from review of input data, analysis parameters, and comparison of ground truth versus output from prior work, and from manipulation of known data using the laws of physics and chemistry. Thus, a large fraction of the knowledge base does not come directly from the brain of a single expert, but is really a digest of the reference material he would use while performing his analysis. This information is sometimes called world knowledge, but it is still very specific to the domain in question.

Most existing rule-based systems contain hundreds of rules, usually obtained by interviewing experts for weeks or months. In any system, the rules become connected to each other by association linkages to form rule networks. Once assembled, such networks can represent a substantial body of knowledge, although some of it may be incomplete, contradictory, fuzzy, or plain wrong.



*Figure 2: A typical database structure.*

In this handbook, we call these networks by the generic label of ROUTINE, which is an assemblage of individual algorithms connected by conditional branching logic. The routine, with its associated computation parameter files and raw data records, constitutes the specific rule network which will be used on this data set. Unfortunately, the network must be created manually, usually by an expert, and tuned for each subsequent use, usually by a low level user with or without the guidance of a human expert.

Some computer aided log analysis systems have an extensive rule base, and can have an extensive knowledge base as well, but are not yet expert systems because they cannot perform any reasoning; they cannot chose the most likely rule network to use for a particular problem. A diagram of the data base for LOG/MATE ESP is shown in Figure 2; it has been especially designed to contain rules, facts, global data, input data, and answers, in anticipation of adding or interfacing an inferencing technique to the system.

An expert usually has many judgmental or empirical rules, for which there is incomplete support from the available evidence. In such cases, one approach is to attach numerical values (certainty factors) to each rule to indicate the degree of certainty associated with that rule. In expert system operation, these certainty values are combined with each other and the certainty of the problem data, to arrive at a certainty value for the final solution. Fuzzy set theory, based on possibilities, can also be utilized.

Often, beliefs are formed or lines of reasoning are developed based on partial or erroneous information. When contradictions occur, the incorrect beliefs or lines of reasoning causing the contradictions, and all wrong conclusions resulting from them, must be retracted. To enable this, a data-base record of beliefs and their justifications must be maintained. Using this approach, truth maintenance techniques can exploit redundancies in experimental data to increase system reliability.

Truth maintenance can be considered a form of learning and pertains to both rules and facts. The knowledge base would learn from current and past analyses based on the following criteria:

1. User's status (expert, intermediate, junior)

2. Consistency within current context ("Did I make a mistake?")

3. Consistency with historic data ("Did I forget something?")

4. Finality quotient ("Are you done fooling around?")

5. Heavy hammer override ("I want it this way, no matter what.")

6. Certainty (user's probability estimate)

7. Housecleaning and editing by user

8. Validity statistics (undefined as yet, but related to closeness to ground truth)

No system described has all these features, and knowledge updating often takes place offline from the actual use of the system.

**The Inference Engine**As indicated earlier, an expert system consists of three major components, a set of rules, a global data base and a rule interpreter. The rules are actuated by patterns, (which match the IF sides of the rules) in the global data base. The application of a rule changes the system status and therefore the data base, enabling some rules and disabling others. The rule interpreter uses a control strategy for finding the enabled rules and deciding which rule to apply. The basic control strategies used may be top down (goal driven), bottom up (data driven), or a combination of the two that uses a relaxation-like convergence process to join these opposite lines of reasoning together at some intermediate point to yield a problem solution.

The rule interpreter, or control strategy, is often called the problem solving paradigm or model in the AI literature. Other terms used are the inference engine, the solution protocol, reasoning, or deduction.

The essential difference between conventional programming and expert systems is this ability to reason or deduce; to take alternate paths, not based on pre-ordained switches, but based on logical rules and the current state of the global data base.

Different types of experts use different approaches to problem solving. Knowledge, for example, can be represented in many different ways. Similarly, there are many different approaches to inferencing and many different ways to order one's activities. Generalized models are available in the form of system building tools.

The consultation/diagnosis/prescription model is a particular type of problem solving technique that is common to several different domains. The name derives from medical problems, such as diagnosing infections and recommending drugs. Log analysis is similar in many ways to the medical problem; reviewing a set of conditions (symptoms), considering various possibilities, and then recommending actions based on a qualified estimate of the probable causes. Most petroleum related expert systems use some form of consultative model.

The problem-solving model, and its methodology, organizes and controls the steps taken to solve the problem. One commonplace but powerful model involves the chaining of IF-THEN rules to form a line of reasoning. If the chaining starts from a set of conditions and moves toward some possible remote conclusion, the method is called forward chaining. An example might be building a custom tailored minicomputer, in which a list of desired features leads to a goal of a complete detailed system configuration parts list. Forward chaining usually is used to construct something.

If the conclusion is known (e.g. it is a goal to be achieved), but the path to that conclusion is not known, then working backwards is called for, and the method is called backward chaining. For example, a set of botanical descriptions ought to lead to a species name by backward chaining to find the set of conditions in the knowledge base which match the plant description at hand. Backward chaining methods are usually used for diagnostic purposes; they start from a list of symptoms and attempt to find a cause which would explain the symptoms.

The problem with forward chaining, without appropriate heuristics for pruning, is that you would derive everything possible whether you needed it or not. For instance, the description of a chess game from its possible opening moves creates an enormous explosion of possibilities. If every elementary particle in in the universe were a computer operating at the speed of light, the universe is not old enough to have computed all possible combinations.

Backward chaining works from goals to sub-goals. The problem here, again without appropriate heuristics for guidance, is the handling of conjunctive sub-goals. Conjunctive goals are those which interact with each other, and which must be solved simultaneously. To find a case where all interacting sub-goals are satisfied, the search can often result in a combinatorial explosion of possibilities too large for real computers.

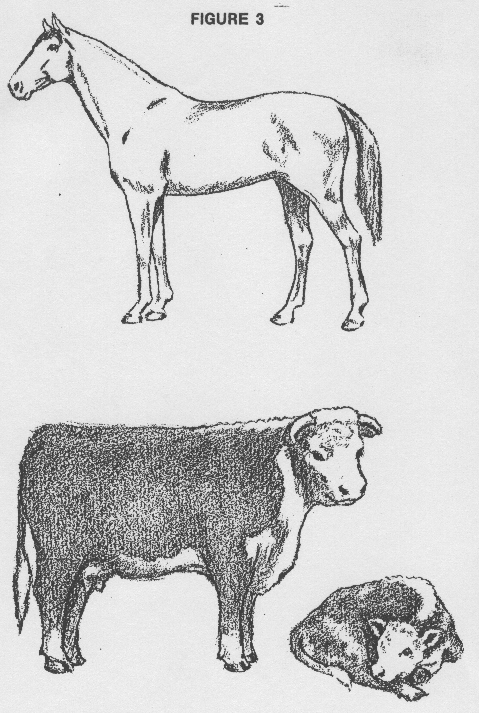
Thus appropriate domain heuristics and suitable inference schemes and architectures must be found for each type of problem to achieve an efficient and effective expert system. There are no universal, general purpose expert systems.

The knowledge of a task domain guides the problem-solving steps taken. Sometimes the knowledge is quite abstract; for example, a symbolic model of how things work in the domain. Inference that proceeds from the model's abstractions to more detailed, less abstract statements is called model-driven inference and the problem-solving behavior is termed expectation driven.

Often in problem solving, however you are working upwards from the details or the specific problem data to the higher levels of abstraction, in the direction of what it all means. Steps in this direction are called data driven. If you choose your next step either on the basis of some new data or on the basis of the last problem-solving step taken, you are responding to events, and the activity is called event driven. Log analysis falls into the data driven, event driven category.

**A Not So Trivial Example**It was not difficult to think of a knowledge base as described earlier. Many computer programs already have them. Humans work easily with tables of data or lists of procedural steps. It is much more difficult to conceive of reasoning or deduction in a computer program, although the simple examples given above suggest the possibilities.

Consider the drawing of the three animals in Figure 3. Humans with prior experience can recognize the difference between them virtually instantly, can name the species and sex, and guess their approximate ages. Some people may even be able to tell the breed of the animals. Could an expert system do the same ?



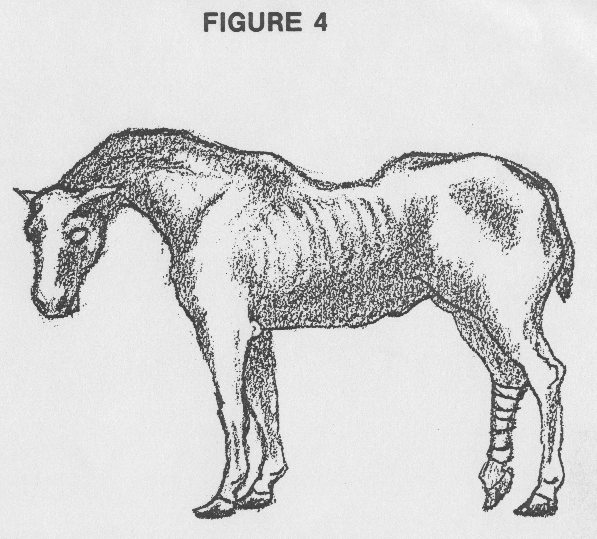
*Figure 3: Drawing of three animals.*

First, try writing down a list of descriptive features that you know for each of these three animals. Do not rely solely on the characteristics in the drawing. Include enough information so that none of these animals could be mistaken for a zebra or a dog. Then check off on each of your lists the observable features of each animal in the illustration. Does your checklist identify each animal uniquely ? Keep improving your list until there is no doubt. You may need a number of conditional statements, using "AND" and "OR" to make identification positive, or even some numerical procedures or probabilities to handle extreme cases.

We have just described the process of extracting knowledge from an expert and using inferencing to draw conclusions. Backward chaining in an expert system would check the checklists, and a reasonable pattern match would generate an answer as to the animal's species, along with a statement as to its probable chance of being correct.

In this case, to emulate the human brain's ability to do pattern recognition, we had to resort to a brute force listing of pattern features, a semi-quantitative description of the animals. Various heuristics would be needed in a real program to account for the fact that you cannot "see" all around the animal in a drawing, and must make assumptions about symmetry and hidden features. After all, this may only be a drawing of a picture of an animal on a billboard, and not a real animal at all,

Now try the animal in Figure 4 on your checklists. Did you identify the animal right away or did you need further updates to your knowledge base? Did any of your updates create conflicts or contradictions? This process describes the "expert as tutor" mode of operation.



*Figure 4.*

Expert systems are not good at pattern recognition from outline drawings such as these, but do better on quantized lists of facts and relationships as described in our example. Real pattern recognition is coming - especially in military and aerospace applications such as target identification and response strategies.

To complete this exercise, consider the possibility of having more data, such as X-rays of the animals' skeleton, autopsy and dissection results. or even a drawing or photograph of other views of the animal. This information would make identification much easier, and allow the programmer to create many new rules, and to add to the factual data base.

These sets of extra data are analogous to extra well logs or extra non-log data, such as core, test, and production history information. Obviously, with more facts to work on, and more rules to evaluate, an expert system to determine animal species or the production to be expected from a well, will do a better job. Thus integration of various disciplines in a common knowledge base is a natural outcome of expert system research.

**Languages and Tools**Tools are software products or techniques used by knowledge engineers to create expert systems. They can be considered as extended languages or environments (like an operating system) which are especially tailored to the requirements of AI.

There is not, however, a one-to-one match between software techniques and problems. One programmer may approach a constraint satisfaction problem using a tool based on backward chaining; another knowledge engineer, faced with the same problem, might choose a tool that relies on forward chaining. However, few knowledge engineers would probably choose to use a backward chaining tool to tackle a complex planning problem, because it is known to be an inappropriate model.

When choosing a tool, you want to be very sure that the specific tool chosen is appropriate for the type of problem on which it is to be used. Unfortunately, since knowledge engineers do not understand how to handle most of the problems that human experts routinely solve, and since there are only a few tools available, many types of expert behavior cannot be conveniently encoded with any existing tool.

Thus in most cases, those who want to employ knowledge engineering techniques have a choice. They can focus on problems that are well understood and ignore those for which there are no available solutions at this time. Or they can develop a sophisticated knowledge engineering team and try to build a system by creating a unique set of knowledge representation, inference, and control techniques in some general-purpose AI language or environment such as INTERLISP, PROLOG, or perhaps OPS5. This is clearly too expensive for most small to medium sized companies, but is the approach taken, for example, by Schlumberger for their Dipmeter Advisor and other AI projects.

Most companies have decided to focus on solving problems for which there are already established tools. Given the large number of available problems with significant paybacks, this is certainly a reasonable strategy. Companies that have decided to develop a team capable of creating unique knowledge systems have usually built that team while working on some fairly well-understood problem, as the author and his colleagues are doing with the LOG/MATE ESP Assistant project, described later.

The tools used by the expert system community involve specialized computer languages and system building tools, as well as specialized hardware architecture, often called LISP machines after the dominant language used in the USA. The other popular language, used mostly in Europe and Japan, is Prolog. Other specialized languages, such as OPS5 written in BLISS, are used in limited areas.

The conventional languages, such as Basic and Fortran and many others, have been successfully used to create expert systems. The AI community tends to downplay these successes, and insist on using LISP. It should be remembered that LISP was invented at a time when Fortran could not handle strings of characters at all. Much invention has since taken place and extended Basic and other languages handle user defined functions, recursion, and text strings quite well, all deficiencies which LISP was supposed to overcome. LISP is also very difficult to read, and programmers often cannot understand or debug each other’s code, in contrast with structured extended Basic which can be composed so as to read well in pseudo English.

TABLE 1: EXPERT SYSTEM TOOLS

System Name Available On Language Supplier

Small Systems  
AL/X Apple II Pascal U. of Edinburgh Edinburgh, Scotland  
ESP Advisor IBM PC Prolog Expert Systems King of Prussia, PA, 19406  
Expert/Ease IBM PC Pascal Expert Software DEC Rainbow San Francisco, CA, 94114  
EXSYS IBM PC C EXSYS Inc. Albuquerque, NM, 71941  
Insight IBM PC Pascal Level 5 Research DEC Rainbow Melbourne Beach, FL, 32951  
M.1 IBM PC Prolog Teknowledge Palo Alto, CA,94301  
OPS5+ IBM PC C Artelligence Dallas, TX, 75240  
Personal Consultant IBM PC C Texas Instruments TI PC Dallas, TX, 75380  
Series-PC IBM PC Lisp SRI International Menlo Park, CA, 94025  
  
NOTE: Approximately 30 small systems are available in the price range of $100 to $1000. Only the best known are listed here.

Medium Systems  
Expert IBM Fortran Rutgers University New Brunswick, NJ, 08903  
KES IBM PC Lisp Software A&E DEC VAX Arlington, VA, 22209 Apollo, Xerox Symbolics  
OPS5, OPS5e DEC VAX Lisp Carnegie Mellon University Pittsburg, PA, 15213  
RuleMaster IBM PC C Radian DEC, HP-300 Austin, TX, 78766  
S.1 DEC VAX Lisp Teknowledge Symbolics Palo Alto, CA,94301 Xerox  
TIMM IBM PC, IBM Fortran General Research DEC VAX Santa Barbara, CA, 93160

NOTE: Medium sized systems cost $1000 to $10,000.

Large Systems  
ART DEC VAX Lisp Inference Corp LMI Los Angeles, CA, 90045  
KEE Xerox, LMI Lisp Intellicorp Symbolics Menlo Park, CA, 94025 TI Explorer DEC, HP-300  
LOOPS Xerox Lisp Xerox Research Palo Alto, CA, 94304  
Knowledge Craft Symbolics Prolog Carnegie Group Inc Pittsburg, PA, 15210

NOTE: Large systems cost $10,000 to $80,000, except LOOPS which is $300 with a Xerox workstation.

In addition to the true languages, the system building tools can be divided into three groups:

1. Small system building tools that can be run on personal computers. These tools are generally designed to facilitate the development of systems containing less than 400 rules and are not discussed further here.

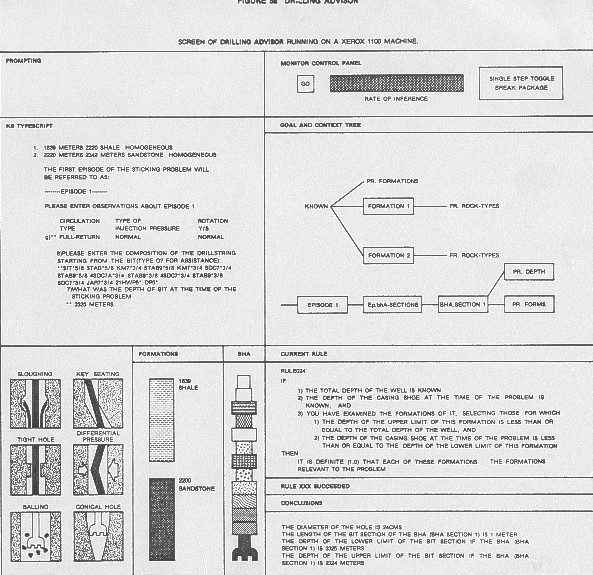
2. Large, narrow system building tools that run on LISP machines or larger computers and are designed to build systems that contain 500 to several thousand rules but are constrained to one general consultation paradigm.

 In this handbook, we call these networks by the generic label of ROUTINE, which is an assemblage of individual algorithms connected by conditional branching logic. The routine, with its associated computation parameter files and raw data records, constitutes the specific rule network which will be used on this data set. Unfortunately, the network must be created manually, usually by an expert, and tuned for each subsequent use, usually by a low level user with or without the guidance of a human expert.

Some computer aided log analysis systems have an extensive rule base, and can have an extensive knowledge base as well, but are not yet expert systems because they cannot perform any reasoning; they cannot chose the most likely rule network to use for a particular problem. A diagram of the data base for LOG/MATE ESP is shown in Figure 2; it has been especially designed to contain rules, facts, global data, input data, and answers, in anticipation of adding or interfacing an inferencing technique to the system.

**Petroleum Industry Examples**DRILLING ADVISOR is a prototype knowledge system developed for the French oil company Societe Nationale Elf-Aquitaine (ELF) by Teknowledge Inc. The system is designed to assist oil rig supervisors in resolving and subsequently avoiding problem situations such as being stuck in the hole. DRILLING ADVISOR was developed by means of a tool called KS300 and is a backward chaining, production rule system.

Currently the knowledge base of DRILLING ADVISOR consists of some 250 rules. Approximately 175 of those rules are used in diagnosis, and the other 75 rules are used in prescribing treatment. Results to date are very encouraging. The system has successfully handled a number of difficult cases that were not included in the set used during its development. Current plans call for extending the capabilities of DRILLING ADVISOR and for integrating it into the actual drilling environment. A sample of the control screen is shown in Figure 5.

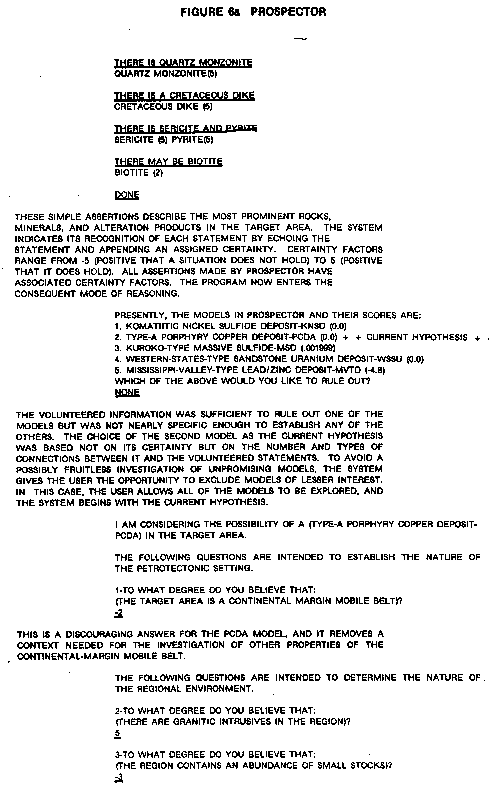


*Figure 5: Control screen for Drilling Advisor*

PROSPECTOR has one foot in the world of research and the other in the world of commercial applications. It was developed in the late 1970's at Stanford Research Institute (SRI) by a team that included Peter Hart, Richard Duda, Rene Reboh, K. Konolige, P. Barrett, and M. Einandi. The development of PROSPECTOR was funded by the U.S. Geological Survey and by the National Science Foundation.

PROSPECTOR is designed to provide consultation to geologists in the early stages of investigating a site for ore-grade deposits. Data are primarily surface geological observations and are assumed to be uncertain and incomplete. The program alerts users to possible interpretations and identifies additional observations that would be valuable to reach a more definite conclusion.

Once the user has volunteered initial data, PROSPECTOR inserts the data into its models and decides which model best explains the given data. Further confirmation of that model then becomes the primary goal of the system, and the system asks the user questions to establish the model that will best explain the data. If subsequent data cause the probabilities to shift, of course, the system changes priorities and seeks to confirm whichever model seems most likely in light of the additional data. See Figure 6 for a sample of the question and answer sequence from a prospector session.



*Figure 6: Question and Answer from Prospector.*

In 1980, as a test, PROSPECTOR was given geological, geophysical, and geo-chemical information supplied by a group that had terminated exploration of a site at Mt. Tolman in Washington in 1978. PROSPECTOR analyzed that data and suggested that a previously unexplored portion of the site probably contained an ore-grade porphyry molybdenum deposit. Subsequent exploratory drilling has confirmed the deposit and, thus, PROSPECTOR has become the first knowledge-based system to achieve a major commercial success. The weakest part of PROSPECTOR's performance was its failure to recognize the full extent of the deposit it identified.

PROSPECTOR's five models represent only a fraction of the knowledge that would be required of a comprehensive consultant system for exploratory geology. SRI continues to develop and study PROSPECTOR, but there are no plans to market the system. The principal scientists who developed PROSPECTOR and KAS, the expert system building tool derived from PROSPECTOR, have left SRI to form a private company (Syntelligence).

PROSPECTOR has never become an operational system. Its innovations and successes, however, have inspired a large number of knowledge engineers, and there are a number of commercial systems under development that rely on one or more of the features first developed and tested during the PROSPECTOR project.

Schlumberger's DIPMETER ADVISOR system attempts to emulate human expert performance in dipmeter interpretation. It utilizes dipmeter patterns together with local geological knowledge and measurements from other logs. It is characteristic of the class of programs that deal with what has come to be known as signal to symbol transformation.

The system is made up of four central components:

- a number of production rules partitioned into several distinct sets according to function (e.g. structural rules vs stratigraphic rules)

- an inference engine that applies rules in a forward-chained manner, resolving conflicts by rule order

- a set of feature detection algorithms that examines both dipmeter and open hole data (e.g. to detect tadpole patterns and identify lithological zones)

- a menu-driven graphical user interface that provides smooth scrolling of log data.

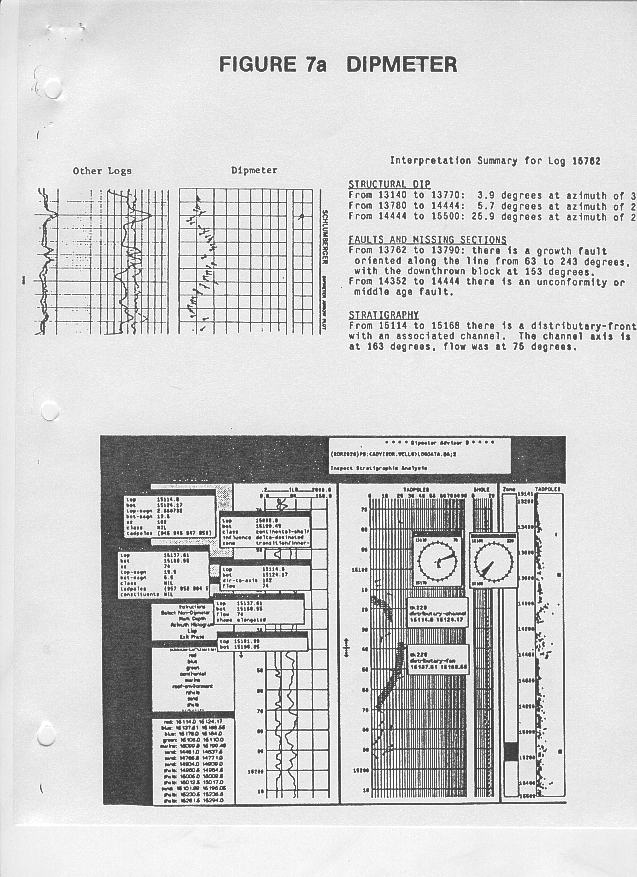
There are 90 rules and the rule language uses approximately 30 predicates and functions. A sample is shown below, similar to an actual interpretation rule, but simplified somewhat for presentation:

IF there exists a delta dominated, continental shelf marine zone AND there exists a sand zone intersecting the marine zone AND there exists a blue pattern within the intersection

THEN assert a distributary fan zone

WITH top = top of blue pattern WITH bottom = bottom blue pattern WITH flow = azimuth of blue pattern

The system divides the task of dipmeter interpretation into 11 successive phases as shown below. After the system completes its analysis for a phase, it engages the human interpreter in an interactive dialogue. He can examine, delete, or modify conclusions reached by the system. He can also add his own conclusions. In addition, he can revert to earlier phases of the analysis to refer to the conclusions, or to rerun the computation.



*Figure 7: Dipmeter.*

1. Initial Examination: The human interpreter can peruse the available data and select logs for display.

2. Validity Check: The system examines the logs for evidence of tool malfunction or incorrect processing.

3. Green Pattern Detection: The system identifies zones in which the tadpoles have similar magnitude and azimuth.

4. Structural Dip Analysis: The system merges and filters green patterns to determine zones of constant structural dip.

\*5. Preliminary Structural Analysis: The system applies a set of rules to identify structural features (e.g. faults).

6. Structural Pattern Detection: The system examines the dipmeter data for red and blue patterns in the vicinity of structural features. The algorithms used by the system to detect dip patterns are beyond the scope of this paper. It is worth noting, however, that textbook definitions do not provide sufficient specification. The problem is complicated by local dip variations and occasional gaps in the data.

\*7. Final Structural Analysis: The system applies a set of rules that combines information from previous phases to refine its conclusions about structural features (eg. strike of faults).

8. Lithology Analysis: The system examines the open hole data (e.g. gamma ray) to determine zones of constant lithology (e.g., sand and shale).

\*9. Depositional Environment Analysis: The system applies a set of rules that draws conclusions about the depositional environment. For example, if told by the human interpreter that the depositional environment is marine, the system attempts to infer the water depth at the time of deposition.

10. Stratigraphic Pattern Detection: The system examines the dipmeter data for red, blue, and green patterns in zones of known depositional environment.

\*11. Stratigraphic Analysis: The system applies a set of rules that uses information from previous phases to draw conclusions about stratigraphic features (e.g. channels, fans, bars).

For the phases shown above, "\*" indicates that the phase uses production rules written on the basis of interactions with an expert interpreter. The remaining phases do not use rules. A sample screen is shown in Figure 7.

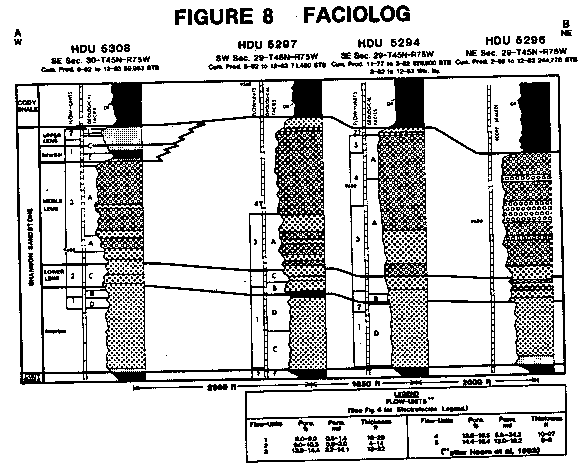
During the creation of these components, Schlumberger has developed a number of proprietary tools for constructing expert systems. These include STROBE for definition of data representation, rule definition and rule integrity checking; IMPULSE for data entry to STROBE; XPLAIN for justifying and explaining rules and deductions; CRYSTAL for interactive display of data, graphics, window management on the screen, as well as task definition; and a relational data base manager. These functions are described quite well in the AI literature and serve as models of the best that is being done in the field. These are listed in the Bibliography to this Chapter. The tools are written in Interlisp-D on Xerox equipment, or Commonlisp and C on DEC VAX equipment. Some processing is done by a host computer which communicates with the Xerox workstation.

Schlumberger also has an extensive research activity in conventional open hole analysis of logs using expert systems. However, the technical literature on the artificial intelligence aspects of the subject is sparse. It is assumed that the tools mentioned above are being used.

FACIOLOG is one of the open hole analysis products based on this technology. It is used to generate a rock facies description from the electrical log measurements. It works well where rock sample descriptions are available to aid calibration.

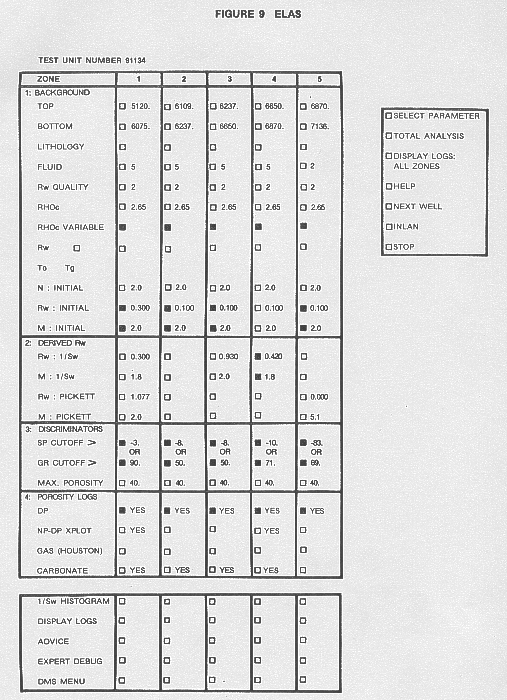
The computation begins with environmental corrections to log data , followed by N-dimensional crossplots of the available curves. The program selects the principal components by considering the length of each axis of the multidimensional crossplot. Once the principal components have been identified, the local modes are projected onto two dimensional crossplots. Local modes represent intervals which have similar log characteristics. These may be large in number and are then re-clustered manually into a lesser number of terminal modes representing geologically significant rock types.

Lithofacies is predicted from log response by pattern matching (or backward chaining) through a database containing the values of the principal components and individual log responses for a large number of possible facies. Each of these values is actually a volume in N-dimensional space. Points which do not fall within any of the volumes are undefined. Points that fall within more than one volume are handled by a probability function which finds the best solution, which is also controlled by a vertical consistency check. This database is created by calibrating to core descriptions, and can be updated to contain local information. Numerous examples can be found in the references, and a typical display shown in Figure 8.



*Figure 8: FACIOLOG Display.*

Another reference describes an experimental lithology identification expert system using curve shape recognition. It is not clear whether this work is related to FACIOLOG in any way.

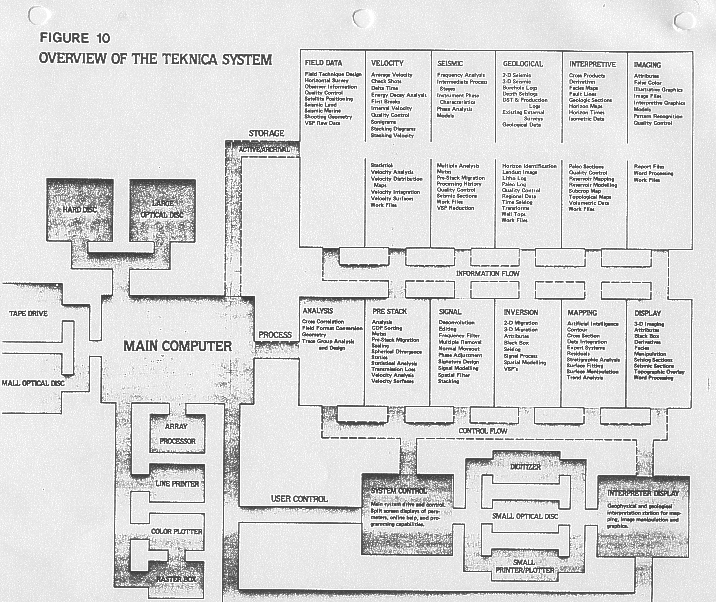


*Figure 9: ELAS display.*

ELAS is an expert system front end for Amoco's interactive log analysis package, which runs on an IBM mainframe-terminal configuration. The front end was written with the EXPERT tool, and is used to prompt a user through the log analysis steps of the interactive program. Both EXPERT and INLAN, Amoco's interactive log analysis package are written in Fortran.

This form of expert system is often called a surface level model. The surface level model is of the production rule type, whereas the deep model is of purely mathematical description, expressed as a set of equations. The latter are implemented as complex software tools, such as reservoir simulators or log analysis packages.

ELAS is currently being used in a research environment for formalizing and integrating knowledge from different experts of Amoco's different regions of exploration and production. Additional efforts are underway to make available this form of analysis to Amoco's practicing well-log analysts in the field. A fair amount of information on this system can be found in the references, and a sample of the master panel for controlling the system is shown in Figure 9.



*Figure 10: Overview of Teknica System.*

MUDMAN is a program developed by NL Baroid Corp. to assist mud engineers in the field. The inputs to MUDMAN include the specifications of the type of mud needed in a particular well and the chemical and physical properties of the mud that is actually present. MUDMAN compares the specifications to the actual properties, provides an analysis of drilling problems, and recommends corrective treatments. It is written in OPS5 on DEC computers.

MUDMAN was specifically designed for sale to Baroid's customers, which are oil companies. Baroid has described MUDMAN as the first expert system sold as a commercial product to the oil industry.

A reference to a Chinese system for well log interpretation called WELIES, based on a tool they built for the purpose (MES), is too brief to determine what the system actually does. It relies heavily on published methods such as PROSPECTOR and DIPMETER ADVISOR. It is written in Fortran on a Perkin-Elmer machine.

TEKNICA Resource Development Ltd have proposed to subscribers, and are actively engaged in, development of a comprehensive seismic data processing system using artificial intelligence techniques. It will enable a user to access many diverse exploration functions using a single human interface package. Typical functions include seismic data analysis, pre-stack and signal processing, seismic inversion, mapping, and display. Well log analysis, especially analysis with geophysical emphasis will be included. See illustration in Figure 10.

The system is written in Interlisp-D on Xerox workstations, using optical discs for data storage. The workstation is connected to an IBM PC/AT with 370 emulator so that all existing Teknica software, written in Fortran, can be controlled from the AI workstation. Although no expert system is not imbedded in the package, this will be forthcoming.

LOG/MATE ESP ASSISTANT - A Prototype Expert Log Analysis System  
Over the past several year, the author has been involved in design and development of an expert system for log analysis call LOG/MATE ESP ASSISTANT. It is based on the premise that an expert system could help a user run an existing program (LOG/MATE ESP) and help choose parameters for the algorithmic solutions.

LOG/MATE ESP is based on the algorithmic solutions and computer program design criteria described in The Log Analysts Handbook, written by E.R.Crain, and published by Pennwell Books.. It is a highly interactive, fourth generation language system developed, written, maintained, and used by experts in anticipation of this expansion into an expert system. It is designed for scientific applications and is not restricted to log analysis.

It already has some features of artificial intelligence, such as English language input and output, and the use of user-controlled or contributed algorithms and graphics. It is command and data driven. It has an elaborate relational data base, and an algorithm processor that isolates mathematical definitions from the operating code. A dictionary system keeps track of parameter names, log curve names, output curve names and other variables.

It has easily understood graphics and printer output, which can be modified or designed under user control, by using simple table or menu entries. Plot and report descriptions are also isolated from operating code to prevent users from crashing the system. The data base is not yet isolated in this fashion, but will be in a this new release.

The first phase of our development effort will result in a system which will interact with a data base containing local geological and petrophysical data, derived from actual log analyses, as well as from initial textbook data. This data represents the parameters needed by the analyst to solve the standard log analysis algorithms used in his area, sorted by formation name and locality.

This database will learn from an expert's use of the system, and could be called a teachable database. It will not learn everything, but only those things we wish it to learn. The learning function will be provided by an application program which will update the historical data base upon user command. This update facility will add parameter values used successfully by the analyst since the last update, provide a mapping facility for data evaluation, and an editing feature to remove or correct inconsistent data.

It would thus be possible for experts to share local knowledge among many users, and to provide less experienced users with a good starting point for their analyses. It also serves as the perfect memory for both advanced and novice users.

Systems sold locally could contain a considerable amount of data since it would be readily available from our own files. Those sold internationally would likely be delivered with an empty database, except for universally accepted rock and fluid properties. These would be updated by the software as analyses are run, preferably by knowledgeable analysts.

An integral part of this enhancement will be a parameter picking feature, so that parameter values can be extracted from the historical data base, as well as from depth plots and crossplots of current data, for use in analyzing the current well. This feature will be utilized by the next phase of the program development.

The second parallel phase will result in a prototype expert system that will act as an analyst's assistant. It would allow less experienced log analysts to perform detailed and successful analyses without the help of an expert. This phase involves extracting analysis rules and methodology from an expert in log analysis. Log analysis rules are of three distinct kinds:

1. algorithm usage rules   
2. parameter selection rules   
3. iterative or re-analysis rules

These rules, or heuristics, will be coded into a rule base which can be used to guide analysts to the correct procedure for a particular problem. Many of the rules for all three types have already been codified by the author his textbook, again in anticipation of this project. They can be generic or location specific rules, but this fact must be identified within the rule. Unstated rules will be elicited by interaction between the expert (the author), a knowledge engineer, and the prototype inference engine operating on a computer specially acquired for the task.

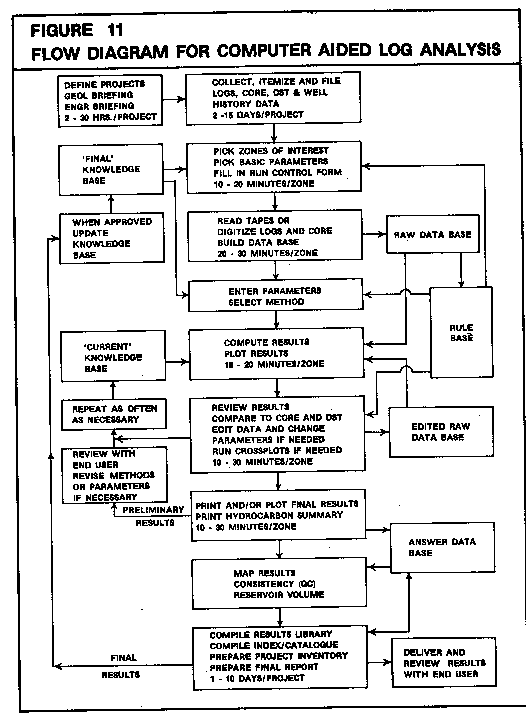
Usage rules are based on the availability of log data and constraints concerning hole condition, borehole and formation fluid type, rock type, and tool or algorithm resolution. They are intended to provide the best initial set of algorithms to use.

Parameter picking rules are also fairly well defined and tie directly to the historical database of phase one, as well as to existing LOG/MATE ESP features such as depth plot and crossplot interactions. These rules are described in various chapters of the textbook, and again are intended to produce the best initial or default values for any job.

Iterative rules are based on result analysis and numerous heuristics about algorithm usage, parameter selection, and data editing. This is where the real expertise of the experienced log analyst lies. These are the most difficult rules to codify, and we may not be successful in this area. Some rules are defined in the book, but most will have to be discovered by actual analysis trial runs.

The way in which these rules interact with the log analysis function is shown in Figure 11.

The system will have to be flexible enough to allow experienced users to add or change rules in all three categories, because many rules vary between analysts and between localities. Therefore, some investigation of appropriate rule managing tools, or inference engines, such as Rulemaster, KEE, KnowledgeCraft, and hardcoded LISP will be undertaken. It is likely that the inference engine required to manage the three kinds of rules will be relatively simple and could be coded specifically in C or LISP for this application, after learning what is needed. It will probably be similar in operational details to our existing algorithm processor.



*Figure 11: Flow Diagram for computer aided log analysis.*

The syntax and protocol for entering, parsing, and invoking rules requires a fairly sophisticated editor. It will be more complex than the inference engine and may have to be written and run in LISP. We will attempt to eliminate this possibility as it adds considerably to the hardware and software cost of the delivery system, and also reduces portability.

Conflict, completeness, and consistency issues are still to be resolved, as no available tools cover these problems adequately. We will have to trust the initial expert and subsequent users to behave rationally, or to be smart enough to find their errors and correct them. This is similar in many ways to debugging problems in conventional programming.

After the different rule sets have been tested in this manner, the rule base will be merged with the LOG/MATE ESP log analysis package. Testing the integrated prototype system on a potential delivery vehicle will follow. The hardware and software to be used for this phase will be a UNIX/C environment on medium priced engineering workstations, possibly with a LISP environment inserted between UNIX and C. The hardware will be similar to the DEC micro VAX, HP series 300, and possibly the IBM PC/AT or IBM PC/RT. Higher performance may be possible on Sun, Apollo, or Symbolics machines if suitable UNIX/LISP/C environments are available for them.



*Figure 12.*

**Limitations of Expert Systems**One of Schlumberger's papers on the Dipmeter Advisor was very candid about their feelings toward progress in expert system development. Their conclusions are as follows:

1. Don't throw away Mark-1 version; progressive releases of software are more practical

2. Expert system development is an incremental process

3. Experts are themselves moving targets

4. Careful definition is impossible beforehand, suggest a contingent definition instead

5. Too much time was spent in knowledge acquisition compared to testing of knowledge against real data

6. Knowledge engineers are not domain experts but often think they are after brief exposure

7. Need more than one expert to overcome bias and gaps in knowledge

8. Need multiple disciplines because subjects are too broad or inter-related

9. Need varied real examples to validate results

10. Experts don't use same rules when new areas are worked

11. Rules give false sense of security to experts and analysts

12. Rule base size triples during development and testing

13. Need excellent human interface for testing and user acceptance

A number of these conclusions contradict directly the cherished tenets of the AI community, such as the use of multiple experts, as expressed in technical papers and textbooks. It seems that there is much to be learned, by both sides, from the expert system development process. Our own experience tends to confirm the points listed above. We have two additional suggestions:

1. Keep it simple and don't try to achieve too much

2. Do your own literature search, read it, and get started right away using simple tools to solve simple problems

An expert system might find that these two rules are the same.

There are six prerequisites before consideration of expert system development for a particular task:

1. There must be a high payoff relative to the effort needed to create the system,

2. The problem can only be solved with the help of an expert's knowledge,

3. An expert is available who is willing to formalize his knowledge,

4. The problem may have more than one rational acceptable answer,

5. The problem, solution, and input data descriptions change rapidly over time or space,

6. The problem is never fully defined.

If at least five of these items are present, it is probably worth investigating an expert system solution to a problem. Otherwise more conventional programming will suffice.

**Conclusions**Log analysis is a highly complex skill and the problem solving techniques used by analysts are poorly understood. The approaches used must take into account many different kinds of knowledge, including physics, chemistry, geology, petrophysics, electronics, drilling practice, and computer science. Artificial intelligence, as represented by the expert system, can provide the tools necessary to allow a computer program to reason about these subjects, given the usage and iterative rules of an expert analyst. Such a system, tunable by each user to his or her own local area, is currently under development by the author and his colleagues.

**Bibliography**1. The Artificial Intelligence Experience Susan J. Scown Digital Equipment Inc., 1985

2. Artificial Intelligence in Business Paul Harmon and David King John Wiley and Sons New York, 1985

3. Understanding Artificial Intelligence H.C.Mishkoff Texas Instruments Learning Center, 1985

4. Handbook of Artificial Intelligence (3 volumes) Avron Barr, Edward Feigenbaum et al W. Kaufman Publishing, Los Altos, 1981

5. Artificial Intelligence David L. Waltz Scientific American, Oct. 1982

6. Programming in Prolog W.F.Clocksin, C.S.Mellish Springer Verlag, 1982

7. LISP P.H.Winston Berthod-Klaus-Paul-Horn, 1983

8. Principles of Artificial Intelligence N.J.Nillson Tioga Press. 1982

9. Programming in OPS5 Lee Brownstone et al Addison Wesley, 1985

10. Building Expert Systems for Controlling Complex Programs Sholom Weiss et al Rutgers University/Amoco, 1983

11. An Approach to Expert Control of Interactive Software Systems Sholom Weiss et al Proc IEEE, 1985

12. The Dipmeter Advisor: Interpretation of Geologic Signals Randall Davis et al Proc IJCAI, 1981

13. The Dipmeter Advisor System Reid Smith and James Baker Proc IJCAI, 1983

14. Applying Artificial Intelligence to the Interpretation of Well Logs P.L.Baker and S.W.Smoliar Proc IEEE, 1984

15. Problem Solving in The Domain of Quantitative Well Log Analysis Stephan W. Smoliar Proc IEEE, 1985

16. Expert Systems and Deep Knowledge Stephan W. Smoliar 10th World Computer Conference,1985

17. An Object Oriented Approach to Planning Stephan W. Smoliar Schlumberger, 1985

18. Strobe: Support for Structured Object Knowledge Representation Reid G. Smith Proc IJCAI, 1983

19. Impulse: A Display Oriented Editor for Strobe Eric S. Schoen, Reid G. Smith Proc Nat Conf AI, 1983

20. A Modular Tool Kit for Knowledge Management Gilles M. Lafue, Reid G. Smith Proc IJCAI, 1984

21. Declarative Task Descriptions as a User Interface Structuring Mechanism R.G. Smith, G.M.E. Lafue, S.C. Vestal Computer Magazine, Sept, 1984

22. Implementation of an Integrity Manager With A Knowledge Representation System R.G. Smith, G.M.E. Lafue Expert Database Systems, 1985

23. Representation and Use of Explicit Justifications for Knowledge Base Refinement R.G. Smith, H.A. Winston IJCAI, 1985

24. The Design of the Dipmeter Advisor System R.G. Smith, R.L. Young Proc ACM, 1984

25. On the Design of Commercial Expert Systems Reid G. Smith AI Magazine, Fall 1984

26. The Use of Qualitative and Qualitative Simulations Reid G. Simmons AI Lab Report, MIT, 1984

27. Oil well Data Interpretation Using Expert System and Pattern Recognition Techniques Alain Bonnet, Claude Dehan IJCAI, 1984

28. Model Expert System (for log analysis) Guan Jiwen, Xu Yung, Chang Minche, Zhao Jizhi IJCAI, 1984

29. Development of the Prospector Consultation System for Mineral Exploration R.O. Duda, et al SRI Report, 1978

30. Model Design in the Prospector Consultant System for Mineral Exploration R.Duda, J.Gasnig, P.Hart Expert Systems and AI Applications, 1979

31. Automatic Determination of Lithology From Well Logs P.C.Delfiner, O.Peyret, O.Serra SPE,1984

32. FACIOLOG - Automatic Electrofacies Determination M.Wolff, J.Pelissier-Combescure SPWLA,1982

**Updated Bibliography - AI Tools**

1. Artificial Intelligence D.L. Waltz Scientific American 1982

2. Artificial Intelligence from A to Z W. Zintz Unix World August 1986

3. Fifth Generation: Banzai or Pie-In-the-Sky? E.A. Weiss Abacus Vol 1 No 1 1984

4. Directions for AI in the '80's P.E. Hart Fairchild Report No 612 1984

5. Artificial Intelligence Becomes a Real Tool J. DelCour GeoByte Spring 1986

6. Expert Systems H.P. Newquist III Computerworld 1986

7. Overview of Expert Systems NASA W.B. Gevarter NBSIR 82-2505 1982

8. Artificial Intelligence: Application Perspectives W.D. McBee Sperry Computer Corp 1983

9. How do expert systems differ from conventional programs? D.A. Waterman Expert Systems Vol 3 No 1 January 1986

10. A Look at Knowldege-Based Information Systems S.P. Ennis GeoByte Winter 1985

11. Expert Systems Research R.O. Duda Fairchild Report No 632 October 1982

12. Expert Systems: working systems and the research literature B.G. Buchanan Expert Systems Vol 3 no 1 January 1986 (huge bibliography)

13. Principles of Rule-based Expert Systems B.G. Buchanan Fairchild Report 636 August 1982

14. Selection of an Appropriate Domain for an Expert System D.S. Prerau The Ai Magazine Summer 1985

15. Acquisition of Procedural Knowledge From Domain Experts P. Friedland Stanford 1985

16. Gradualness Facilitates Knowledge Refinement R. Rada IEEE 1985

17. What to Expect from Expert Systems Z.P. Zeman Canadian Data Systems April 1986

18. Artificial Intelligence Programming Languages Dr. J. Glasgow Unix/World August 1986

19. Unix-Based Tools Help Develop Expert Systems T.J. Landesberg Unix/World August 1986

20. Unix and AI, A Beautiful Marriage R.H. Anderson and R.B. Greenberg Unix/World August 1986

21. An Evaluation of Commercial Expert System Building Tools R.S. Wall Texas Instruments Report 85-30 November 1985

22. Do-It-Yourself Expert Systems J. Gold Computer Decisions January 1986

23. An Affordable Adviser (EXSYS) F.J. Derfler Jr. PC Magazine April 1985

24. Expert-Ease Makes Its Own Rules F.J. Derfler Jr. PC Magazine April 1985

25. A Knowledge Engineering Facility for Building Scientific Expert Systems (RuleMaster) C. E. Riese 1986

26. Expert Systems in Engineering: Some Issues (RuleMaster) C.E. Riese IEEE 1986

27. An Introduction to Expert Systems and RuleMaster C.E. Riese IEEE 1986

28. RuleMaster: A Second-Generation Knowledge-Engineering Facility D. Michie Radian Corp. 1986

29. An Expert System to Aid in Severe Thunderstorm Forecasting S.M. Zubrick Radian Corp. 1986

30. RuleMaster Software Description Radian Corp. 1986

31. Knowledge Craft Software Description Carnegie Group 1985

**Updated Bibliography - Oil Field Applications**1. A Primer on Expert Systems for Well Log Analysis E.R. Crain CWLS Journal 1985

2. Artificial Intelligence - Not Really Black Magic J. Davis Small Business Quarterly Spring 1986

3. Building Expert Systems for Controlling Complex Programs S. Weiss, J. Patchett Amoco IJCAI 1983

4. An Approach to Expert Control of Interactive Software Systems C.V. Apte IEEE 1985

5. An Interactive Computer System for Well Log Analysis S.T. Martner SPWLA 1975

6. Expert Systems: A User's Perspective of Some Current Tools S.P. Ennis IJCAI 1982

7. The Dipmeter Advisor: Interpretation of Geologic Signals R. Davis IJCAI 1981

8. The Dipmeter Advisor System R.G. Smith Schlumberger 1983

9. Dipmeter Advisor J.D. Baker Schlumberger 1983

10. Declarative Task Description as a User-Interface Structuring Mechanism R.G. Smith Schlumberger IEEE 1984

11. Problem Solving in the Domain of Quantitative Well Log Interpretation S.W. Smoliar Schlumberger IEEE 1985

12. An Object-Oriented Approach to Planning S.W. Smoliar Schlumberger 1985

13. Expert Systems and "Deep" Knowledge S. W. Smoliar USC 1986

14. A Modular Tool Kit for Knowledge Management G.M. Lafue Schlumberger 1986

15. Applying Artificial Intelligence to the Interpretation of Petroleum Well Logs P.L. Baker, S.W. Smoliar Schlumberger IEEE 1984

16. STROBE: Support for Structured Object Knowledge Representation R.G. Smith Schlumberger 1984

17. Impulse: A Display Oriented Editor for STROBE E. Schoen Schlumberger 1983

18. Representation and Use of Explicit Justifications for Knowledge Base Refinement R.G. Smith Schlumberger 1986

19. A Perspective on Automatic Programming D. Barstow Schlumberger 1983

20. On the Development of Commercial Expert Systems R.G. Smith Schlumberger AI Magazine Spring 1984

21. The Role of Expert Systems in Producing Log Interpretation Software G.M. Zivy Schlumberger 1989

22. The Use of Qualitative and Quantitative Simulations R.G. Simmons MIT 1984

23. An Example Expert System for the Interpretation of Depositional  Environments K.B. Krystinik USGS Report 85-30 1985

24. Sample Dialog: TWELL Radian Corp. 1985

25. Comparison of an Expert System to Human Experts in Well-Log Analysis and Interpretation E.E. Einstein, K.W. Edwards SPE Formation Evaluation March 1990

26. Log/Mate Assistant: The Integration of an Expert System in a Log Analysis Domain L. Pepperdine CWLS 1987

27. An Artificial Intelligence Approach to Well Log Correlation R.A. Startzman SPWLA 1986

28. Semi-Automated Well Matching Using Gene-Typing Algorithms and a Numerical Lithology Derived From Wire-Line Logs C.M. Griffiths SPWLA 1988

29. Recognition of a Hidden Mineral Deposit by an Artificial Intelligence Program (PROSPECTOR) Science Vol 217 September 1982

30. Model Expert System (for log analysis) Guan Jiwen, Xu Yung, Chang Minche, Zhao Jizhi IJCAI, 1984

31. Model Design in the Prospector  Consultant System for Mineral Exploration R. Duda Expert Systems and AI Applications USGS 1980

32. Artificial Intelligence and the Oil Industry D. Solano ELF Aquitaine 1985

33. A Smart Computer Helps Solve Drilling Problems C.L.Stark  NL Baroid O&GJ 1985

34. Building an Expert System Helps Classify Sedimentary Basins and Assess Petroleum Resources B.M. Miller GeoByte Spring 1986

35. Assistant Expert in Formation Evaluation A.H. Whittaker SPWLA 1986

36. Logix: A Knowledge-Based System for Petrophysical Evaluation L.J.B. Hoffman SPWLA 1988

37. Well Log Data Interpretation Using Artificial Intelligence Technique X. Wu SPWLA 1988

38. Using an Expert System to Identify the Well Test Interpretation Model A.U. Al-Kaabi SPE 1988

39. Use of Artificial Intelligence for Model Identification and Parameter Estimation in Well Test Interpretation O.F. Allain SPE 1988