On the Development of Commercial Expert Systems

Reid G. Smith

Schlumberger-Doll Research
Old Quarry Road
Ridgefield, Connecticut 06877

Abstract

We use our experience with the Dipmeter Advisor system for well-log interpretation as a case study to examine the development of commercial expert systems. We discuss the nature of these systems as we see them in the coming decade, characteristics of the evolution process, development methods, and skills required in the development team. We argue that the tools and ideas of rapid prototyping and successive refinement accelerate the development process. We note that different types of people are required at different stages of expert system development: Those who are primarily knowledgeable in the domain, but who can use the framework to expand the domain knowledge; and those who can actually design and build expert system tools and components We also note that traditional programming skills continue to be required in the development of commercial expert systems Finally, we discuss the problem of technology transfer and compare our experience with some of the traditional wisdom of expert system development.

THE PAST DECADE has seen the development of a number of expert systems, mostly by AI researchers for use in research environments. To date, few have been utilized for industrial applications. As a result, we have little experience with which to characterize either the nature of commercial expert systems or their development process.

The Dipmeter Advisor system is the result of a four year effort by Schlumberger to apply expert systems technology to problems of well-log interpretation. We have observed during this effort that the development of a commercial expert system imposes a substantially different set of constraints and requirements in terms of characteristics and methods of development than those seen in the research environment.

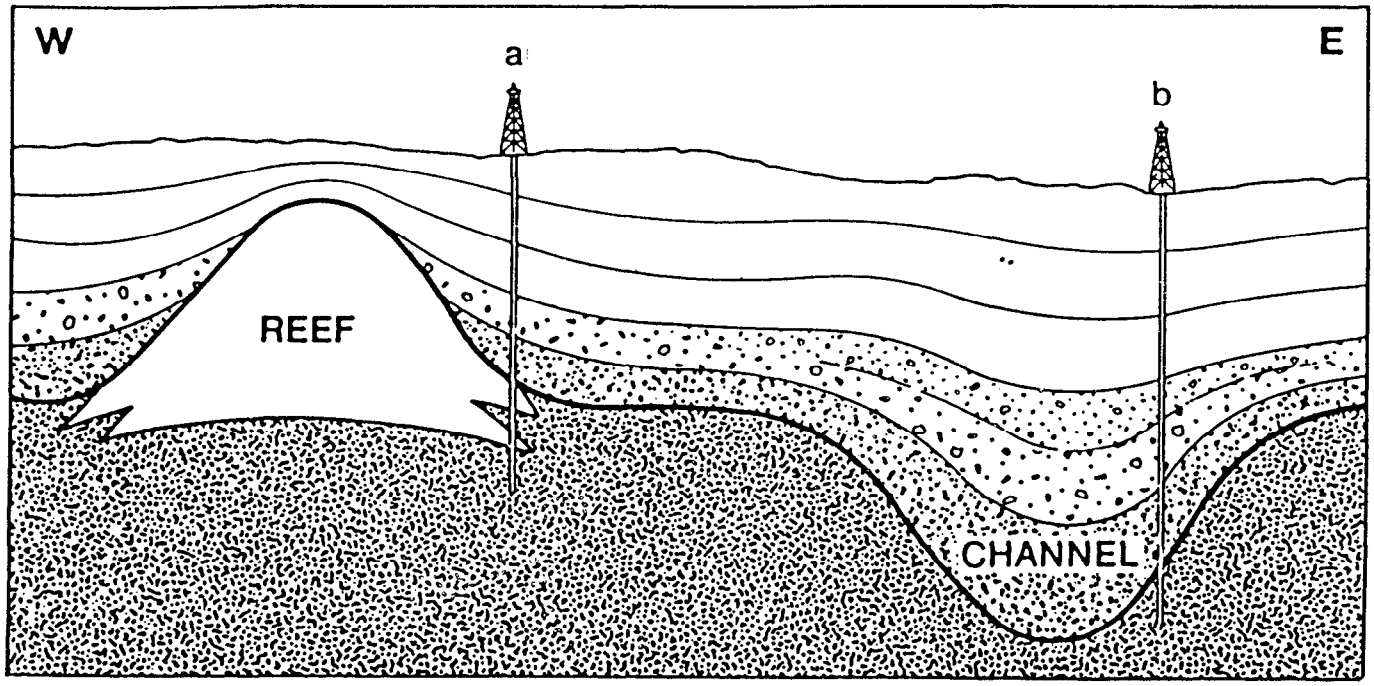
This article is intended as a case study. We briefly describe the dipmeter interpretation problem and the evolution of the Dipmeter Advisor system. During its development a number of ideas have surfaced that we believe to be characteristic of this type of effort, given the current state of the technology. While the data are too sparse for definitive results, these ideas are thought to be important and suggestive as guidelines for subsequent commercial expert system undertakings.

David Barstow, J A. Gilreath, Tom Mitchell, and Peter Will made a number of helpful suggestions for this paper David Gallo and Chip Hendrickson provided the football figures

Example: Dipmeter Interpretation

The Problem

Oil-well logs are made by lowering tools into the borehole and recording measurements made by the tools as they are raised to the surface. The resulting logs are sequences of values indexed by depth. Logging tools measure a variety of petrophysical properties. The dipmeter tool in particular



Dip in Subsurface Formations.

Figure 1.

measures the conductivity of rock in a number of directions around the borehole. Variations in conductivity can be correlated and combined with measurements of the inclination and orientation of the tool to estimate the magnitude and azimuth of the dip or tilt of various formation layers penetrated by the borehole (Figure 1).

Because the dipmeter tool has high resolution in the vertical direction (0.1-0.2 in.), it provides the petroleum geologist with detailed information on relatively fine-structured sedimentary beds. This type of information is invaluable in defining hydrocarbon reservoir structure and designing methods to drain such reservoirs.

Knowledge of the dip variations as a function of depth in the vicinity of the borehole does not in itself identify geologic features. However, when combined with knowledge of local geology and rock properties measured by other logs (*e.g.*, lithology (sand, shale,)), the characteristic dip patterns (signatures) of geologic events in the depositional sequence can be interpreted.

The right channel of Figure 2 is an interval of a dipmeter log. Dip estimates are shown as tadpoles. Dip magnitude increases to the right of the graph, and the down dip direction is indicated by the tail on each tadpole. The vertical axis is depth. Hollow tadpoles indicate lower confidence dip estimates than solid tadpoles. (So, for example, the tadpole at 8360 ft. indicates a formation that is dipping down to the southeast at approximately 24°.) The left channel is a gamma ray log. (It measures natural gamma radiation in the formation—a rudimentary lithology indicator.)

Sequences of tadpoles can be grouped together in patterns. Three of the characteristic dip patterns are described below (Schlumberger, 1981).

- *Green Pattern*: An interval (zone) of constant dip magnitude and azimuth. This pattern is characteristic of structural dip—caused by large-scale tectonic disturbance that occurs long after deposition and compaction of sediment.
- *Red Pattern*: A zone of increasing dip magnitude with constant azimuth over depth. This pattern is indicative of down dip thickening, which may be associated with distortions near structural features (*e.g.*, faults), differential compaction of sediment over buried topographic features (*e.g.*, reefs), or channel filling.
- *Blue Pattern*: A zone of decreasing dip magnitude with constant azimuth over depth. This pattern is indicative of down dip thinning, which may be associated with distortions near structural features, differential compaction beneath denser overlying deposits (*e.g.*, sand lenses), or sediment transport by water or wind.

From this localized data, a skilled interpreter is often able to make comprehensive deductions about the geological history of deposition, the composition and structure of the beds, and the optimum locations for future wells.

The Dipmeter Advisor System

The 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 (Nii, 1982). The program is written in INTERLISP-D and operates on the Xerox 1100, 1108, or 1132 Scientific Information Processor.[1]

The system consists 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); and a menu-driven graphical user interface that provides smooth scrolling of log data.

Conclusions are stored as instances of one of 65 token types, with approximately 5 features/token, on a blackboard that is partitioned into 15 layers of abstraction (*e.g.*, patterns, lithology, stratigraphic features). There are 90 rules, and the rule language uses approximately 30 predicates and functions. The rules have the empirical association flavor. A sample is shown below.[2]

IF

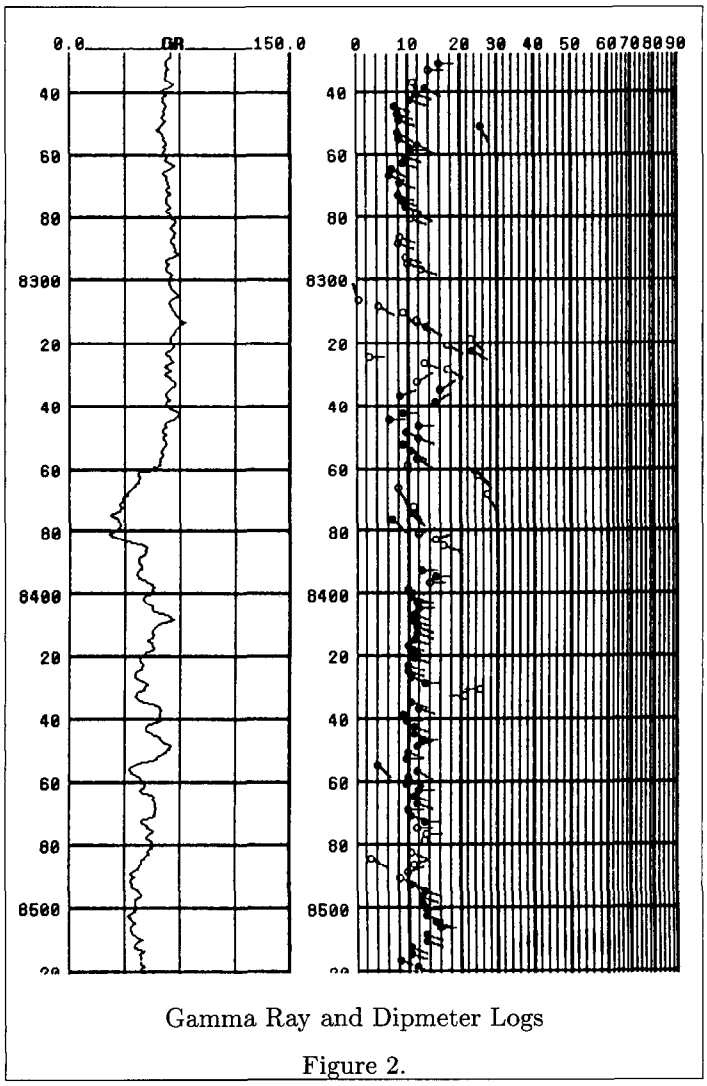
there exists a delta-dominated continental-shelf marine zone, and
there exists a sand zone intersecting the marine zone, and
there exists a blue pattern within the intersection,

THEN

assert a distributary fan zone
top ← top of blue pattern
bottom ← bottom of blue pattern
flow ← azimuth of blue pattern

The system divides the task of dipmeter interpretation into eleven 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.

- **Initial Examination:** The human interpreter can peruse the available data and select logs for display.
- **Validity Check:** The system examines the logs for evidence of tool malfunction or incorrect processing.
- **Green Pattern Detection:** The system identifies zones in which the tadpoles have similar magnitude and azimuth.



Gamma Ray and Dipmeter Logs

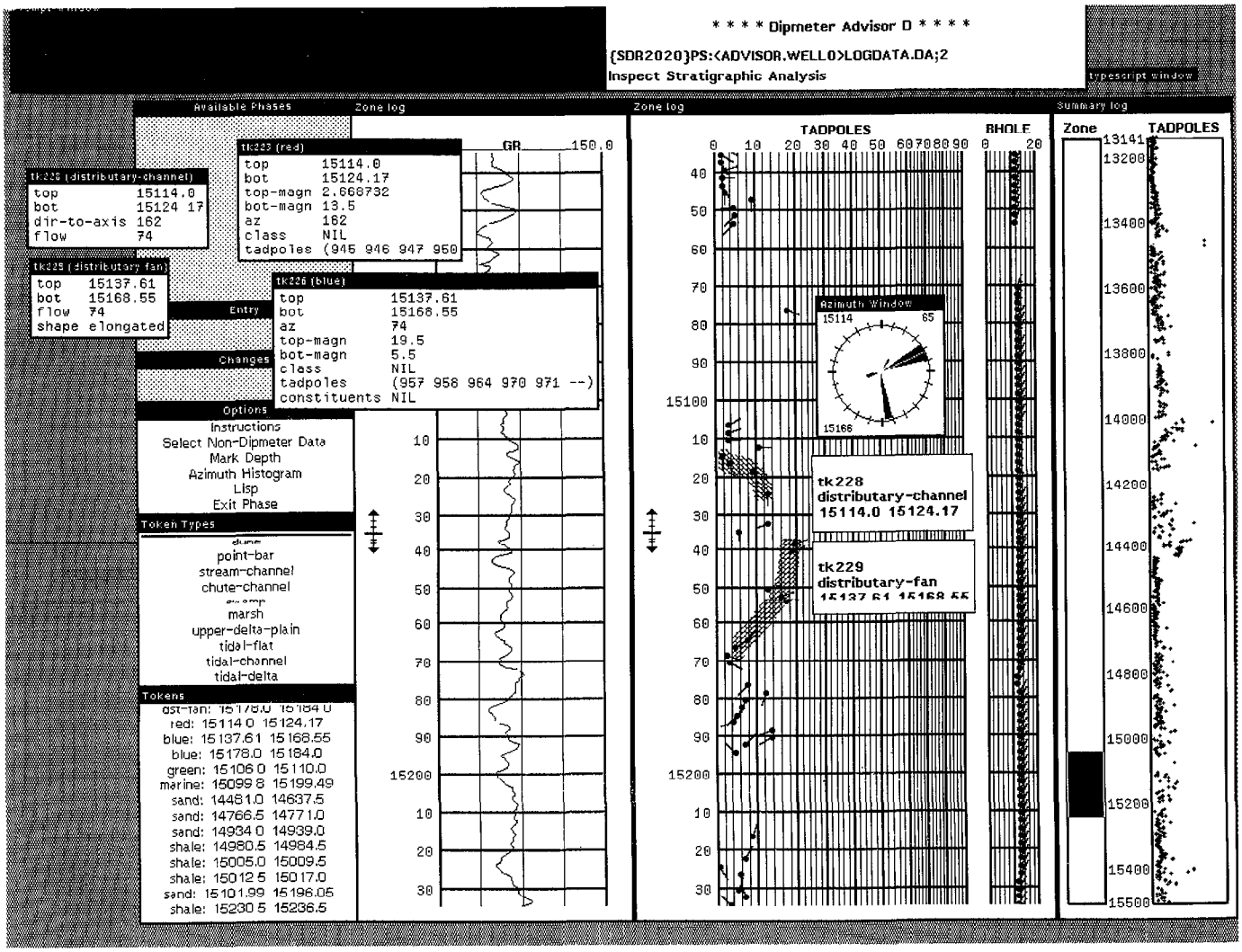
Figure 2.

- **Structural Dip Analysis:** The system merges and filters green patterns to determine zones of constant structural dip.
- **Preliminary Structural Analysis:** The system applies a set of rules to identify structural features (*e.g.*, faults).
- **Structural Pattern Detection:** The system examines the dipmeter data for red and blue patterns in the vicinity of structural features [3]
- **Final Structural Analysis:** The system applies a set of rules that combines information from previous phases to refine its conclusions about structural features (*e.g.*, strike of faults).
- **Lithology Analysis:** The system examines the open hole data (*e.g.*, gamma ray) to determine zones of constant lithology (*e.g.*, sand and shale).
- **Depositional Environment Analysis:** The system applies a set of rules that draws conclusions

[1]Early versions of the program are described in (Davis, 1981) and (Gershman, 1982)

[2]This sample is similar to the actual interpretation rule, but has been simplified somewhat for presentation

[3]The algorithms used by the system to detect dip patterns are beyond the scope of this paper.



Stratigraphic analysis interaction

Figure 3.

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.

- **Stratigraphic Pattern Detection:** The system examines the dipmeter data for red, blue, and green patterns in zones of known depositional environment.
- **Stratigraphic Analysis:** The system applies a set of rules that use 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.[4]

Figure 3 shows a sample Xerox 1100 screen following the stratigraphic analysis phase. On the extreme right the system displays a summary log of dip magnitude for the entire well. The black box indicates the region of the well that is expanded in the second window from the right. This window shows the dipmeter data together with the deviation of the borehole itself. The next window displays two other logs: GR (gamma ray) and ILD (a resistivity log). (Each of these windows can be smoothly scrolled by moving the mouse into its speed bar, one of which is visible on the left side of the dipmeter window. A more radical movement can be achieved by moving the mouse into the black elevator box visible on the right side of the dipmeter window. The size of the interval viewable in the dipmeter and other log windows is also under mouse button control.)

The system summarizes relevant conclusions in the (scrolling) windows in the lower left hand part of the screen. The

[4]The rules obtained to date are due to J. A. Gilreath of Schlumberger Offshore Services, New Orleans, LA The feature detectors and signal-processing algorithms were written independently by project members

user (a dipmeter interpreter) has selected a number of conclusions to be examined in greater detail and shown as annotations on the dipmeter log. Also shown is the dip azimuth trend before and after structural dip removal.[5]

Building Commercial Expert Systems

Embedded Systems

Domain practitioners are typically much more interested in the utility and performance of a system that is to help them solve their problems than in the particular methods used to construct it. Furthermore it is unlikely that traditional AI methods alone will solve real problems. They are likely to be augmented by techniques from signal processing and pattern recognition, to name but two possibilities. This implies that the computer scientist involved in commercial expert system development must be prepared to solve problems that involve a variety of disciplines and techniques.

It is our view that the expert system kernel is likely to be a (perhaps even relatively small) component embedded in a larger system. The particular suite of problems common to signal understanding problems may, of course, bias our outlook, but we believe that it is difficult to avoid the conclusion that acceptance and real use of expert systems depend on far more than a knowledge base and inference engine.[6]

Indeed our experience has been that these traditional parts of an expert system are not the predominant parts of the overall system either in terms of the amount of code or the resources required for system development. It is instructive in this regard to examine the relative amounts of code devoted to various functions in the Dipmeter Advisor system:

Inference Engine:	8%
Knowledge Base:	22%
Feature Detection:	13%
User Interface:	42%
Support Environment:	15%

This breakdown cannot be used, of course, as a direct measure of programming effort or as an indicator of where the system gets its power. However, the human interface figure especially is familiar to designers of expert systems like MYCIN (Shortliffe, 1976), (VanMelle, 1981) and PROSPECTOR/KAS (Reboh, 1981). It demonstrates the importance of a good programming language, and indicates that traditional programming skills continue to be required for the development of commercial expert systems.

[5]The scrolling graphics code was written by Paul Barth and Tony Passera. Extensions to the INTERLISP-D menu package were written by Eric Schoen

[6]Gaschnig has made a similar observation in the context of the PROSPECTOR system (Gaschnig, 1982).

System Evolution

Based on our experience, we hypothesize an oscillating focus of attention in commercial expert system development projects. Initially, the focus is a demonstration of feasibility; acquiring the knowledge for a constrained problem and finding the appropriate set of expert system tools with which to encode and apply the knowledge. This phase could be relatively short. It is followed by a phase of expansion of the domain knowledge—during which the expert system tools remain relatively constant. A point will likely come at which the intial tools do not provide sufficient power to allow continued expansion of the system's expertise. At that point, the focus will move away from domain problems and toward selection—more likely development—of new expert system tools. Once a new set of more powerful tools has been constructed, then the focus will again return to the domain problems at hand.

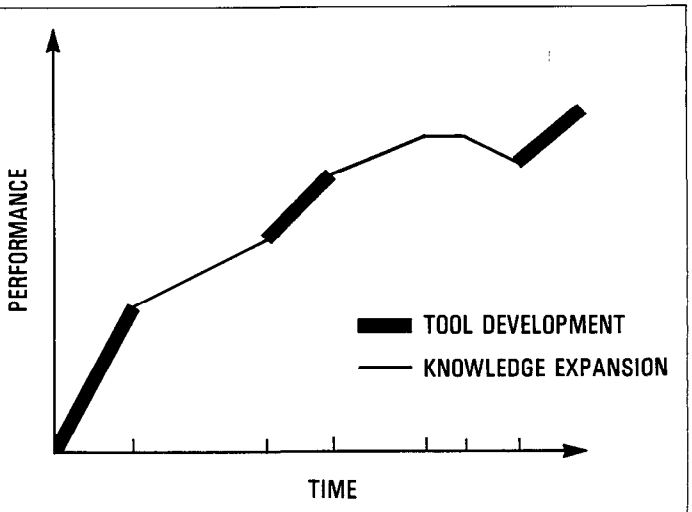
Naturally any particular system may not pass through very many of these oscillations. The focus in the R1 project, for example, didn't appear to oscillate at all (McDermott, 1981). We believe this is due to the nature of the task. There was little of the uncertainty about the nature of the problem that is evident in the the signal understanding or diagnosis tasks. Consequently the initial tools were in fact sufficiently powerful to handle the problem.

In the MYCIN project we seem to be observing the beginnings of an oscillation. The initial system was constructed. Then the rule base was expanded, leaving the initial expert system tools intact. More recently a new design, NEOMYCIN, has appeared—a new set of tools (Clancey, 1981).

Along with the oscillating focus, we hypothesize a rough performance versus time curve. For this discussion performance is taken to include factors such as computation time, accuracy of solutions, and breadth of coverage. We expect this curve to show periods of high positive slope corresponding to implementation of new expert system tools, followed by periods of lower slope corresponding to expansion of domain knowledge, followed by periods of level or even decreasing slope corresponding to reaching (or surpassing) the amount of domain knowledge and generality that can be supported by the tools.

Figure 4 shows the type of performance improvement that we hypothesize, together with the relative emphasis. The emboldened portions of the graph indicate periods of new expert system tool development. The remaining portions correspond to periods of expansion and refinement of domain knowledge. (During the startup period, of course, the two activities proceed concurrently.)

It is currently the case that the precise set of tools required to solve a given problem cannot be accurately predicted *a priori*. Periods of domain knowledge expansion using relatively stable tools are required to expose problem areas and focus tool selection and development. As experience with expert systems grows, for any given problem,



Expert System Incremental Development

Figure 4.

development will effectively begin further along the curve. Developers will start with a better understanding of the set of tools that will eventually be required to solve the problem.

At any given point in time, then, an expert system may require improvement in terms of domain knowledge and expert system tools. Just as the focus of the project will vary, depending on which of the two types of improvement is most pressing, the type of person required to improve the system will also vary.

Improvements in the first area can be made to a large extent by people primarily knowledgeable in the domain, but not necessarily knowledgeable in the design of expert systems. For example, at one stage of its development the Dipmeter Advisor system was familiar with a relatively small number of different lithologies. The performance of the system could be improved in this area without redesign. Similarly, the coverage of the rules could be extended to handle more environments, or specialized to handle local anomalies.[7]

Improvements in expert system tools cannot be made without redesign. This type of effort requires a person who can build such systems, as opposed to one who can use the framework to expand capabilities. For example, the Dipmeter Advisor system uses rule order to help circumvent potential multiple interpretations for the same interval in the well, or simply draws multiple conclusions for the same zone. The human interpreter must select the correct interpretation. The system also has a very local view of consistency in the vertical sequence. This is attributable to the fact that it is reasoning from sets of empirical rules and has no model of the underlying geological processes that lead to the rules. Improvements in these areas cannot be made without redesign.

[7]We have already noted, however, the likelihood that traditional programming skills will continue to be required.

System Development

We have attempted a critical review of the development side of the Dipmeter Advisor system. Although we are as yet unable to abstract a development methodology, several observations stand out. Almost every major issue and decision in the evolution of the Dipmeter Advisor system addressed one or more of the following:

- Demonstration of Feasibility
- Demonstration of Utility and Performance
- Evaluation of Utility and Performance.

Demonstration of Feasibility: The problem of dipmeter interpretation was initially selected as a vehicle for investigating the applicability of expert system techniques to well-log interpretation. Until feasibility could be demonstrated, other questions were secondary.

As a first step, a substantial effort was expended on acquisition of dipmeter interpretation knowledge. This effort was carried out over a 12 to 18 month period using standard techniques (protocols, videotape, discussion, representative examples, and so on). A single expert was studied in detail, again adhering to standard practice.

The implementation of a prototype system followed data acquisition and was carried out in approximately four months (completed in December 1980). The rule base and inference engine were written in INTERLISP (245 Kbytes of source code) and ran on a DEC 2020. The user interface was graphical, written in FORTRAN (450 Kbytes of source code), and ran on a RAMTEK 9400 connected to a VAX 11/780. The VAX and 2020 were linked via a CHAOSnet. The rule base was made up of aproximately 30 rules. There were also several feature detectors and signal processing algorithms.

Demonstration of Utility and Performance: The prototype system demonstrated to the expert that significant analyses were possible. To determine commercial viability, other issues must be addressed. Does the system solve enough of the problem to be interesting and useful? Can the system perform with the efficiency and interactivity necessary in a field environment without overutilizing available computing resources?

Two examples demonstrate the problem. The initial prototype had no means of actually detecting the red and blue patterns and the lithology zones that are required to perform an unaided interpretation. It did not solve enough of the problem to be useful. This lack resulted in implementation of algorithms for simple detection of tadpole patterns and lithologic zones.

Second, the detection of green patterns and determination of structural dip took approximately 18 minutes in the first test well. This duration was unacceptable for actual use—later effort reduced the time to under 2 minutes.

Evaluation of Utility and Performance: Field evaluation was the next hurdle for the Dipmeter Advisor system. Several questions had to be addressed. First, was the rule base sufficiently complete to solve correctly a wide variety of problems in the geological environments for which it was de-

Several questions had to be addressed. First, was the rule base sufficiently complete to solve correctly a wide variety of problems in the geological environments for which it was developed? Second, what changes and effort would be required when working in other geological environments? And third, did the rule base sufficiently capture the thinking of enough dipmeter interpreters to be useful?

To date, this has been the most difficult area. People in the engineering and field groups had to address the above questions. To accomplish this, the prototype system had to be capable of operating in their existing environment—possibly upgraded with modest investment.

One of the difficulties with the initial prototype was the unusual architecture of linked computers, which was not a standard company configuration. In an effort to facilitate testing, the system was reimplemented in FRANZLISP (except for the graphical interface), totally on the VAX 11/780. Unfortunately this change did not solve the problem. The VAX/RAMTEK configuration, as a shared resource in a generally overloaded situation, required an excessively long time to complete a case. Under worst conditions, it took several hours. (In an unloaded VAX environment, it could be completed in one-half hour or less.)

At this point, new technology came to the rescue, and the system was re-implemented on the Xerox 1100, which has both a dedicated processor and sophisticated graphics. In this implementation the graphical interface code was integrated into the remainder of the system. The result was approximately 612 Kbytes of INTERLISP-D source code. This implementation was robust enough and fast enough to allow transfer to a Schlumberger Interpretation Engineering group for testing in a non-research environment.

We can summarize this section as follows: A commercial expert system is ultimately constructed to solve a real problem (as opposed to being constructed, say, to determine the limits of a problem-solving architecture). As a result, the developers should avoid a demonstration mentality. Careful thought at all stages of development about the eventual disposition of the system may prevent the necessity for multiple re-implementations.

The Development Team

Development of a commercial expert system requires people with a variety of skills. The following set is typical. We will expand on it in the remainder of this section.

- Domain Expertise
- Knowledge Engineering
- Expert System Tool Design
- Programming Support

First, it goes without saying that committment of one or more articulate domain experts is crucial to the success of any expert system development.

The term knowledge engineer is normally used to mean computer scientist intermediary—the link between expert and machine. We have divided this role into two parts in order to emphasize that two different kinds of task are involved. The development team normally requires at least one member to interact with the domain experts and encode domain knowledge. We reemphasize that the interaction and encoding activities do not necessarily require someone who can construct expert systems, but rather someone who can become knowledgeable in the domain and who can use an existing expert system framework to extend the capabilities of the evolving system.

The team also requires someone with a detailed understanding of the design and implementation of expert systems—someone who can construct the underlying framework in which to encode domain knowledge. Unfortunately, such people are currently scarce and in demand. Among the ways around this bottleneck are use of off-the-shelf development tools, and training of existing staff in expert system design techniques. Companies presently exist to perform training. Development tools are somewhat more problematic, but they too have started to appear. We will return to this point later in the article.

Finally, the development team requires traditional programming support for integration into pre-existing systems, for graphical interfaces, and so on. In this catchall category we include expertise in related areas (*e.g.*, statistical algorithms and signal processing algorithms) as dictated by the application domain.

Naturally, some of the skill categories shown may be co-located in the same persons. (This has traditionally been the case for knowledge engineering/expert system tool design.)

In later stages, experiment designers, software engineers, and other domain practitioners (not necessarily experts) are required to test and debug the knowledge and framework in more stringent and wide-ranging tests and to produce the actual commercial product. Once again, it is possible that these tasks will fall to the original team members. We would argue, however, that the downstream engineering of the original code is not the best utilization of people with expert system design skills. These people are too few in number today to be underutilized.

Rapid Prototyping and Successive Refinement

In the beginning of a commercial expert system development project, it is important to demonstrate the feasibility of the system. Rapid prototyping seems to be an appropriate strategy—especially given the usual vagueness of the understanding of what can be accomplished.

The main concern in such an approach is a flexible and powerful development environment. Traditionally, such an environment is not even closely related to the commercial computational environment. This lack leads to the problems noted above. With the advent of inexpensive personal workstations, however, there is real hope that the situation may be changing (as has been our experience with the Dipmeter Advisor system).

Significant questions still remain. One of the problems of rapid prototyping is that it provides a good start toward system development, but does not offer clear guidance on how to produce a well-engineered commercial product (see, for example Sheil, 1983). Traditionally this fault is viewed as a problem in technology transfer.

The Dipmeter Advisor system has been developed using rapid prototyping techniques and has evolved as a series of prototypes through successive refinement. Our experience with the process suggests technology transfer through not a single release from research to engineering but rather through successive releases, corresponding to successive prototypes. This type of transfer is appropriate for an expert system in which domain knowledge expansion and refinement can be expected to continue for some time, but for which the system framework has demonstrated that it is sufficiently powerful to warrant engineering effort.

Such an approach to technology transfer naturally imposes restrictions. The designers must somehow convey to their engineering organizations a more accurate perception of the expected lifetimes of the prototypes. Furthermore, the designers are forced to pay even more attention to user interfaces than our earlier figures would suggest. If the systems are going to be changing rapidly then they must have especially convenient and easy-to-learn interfaces.

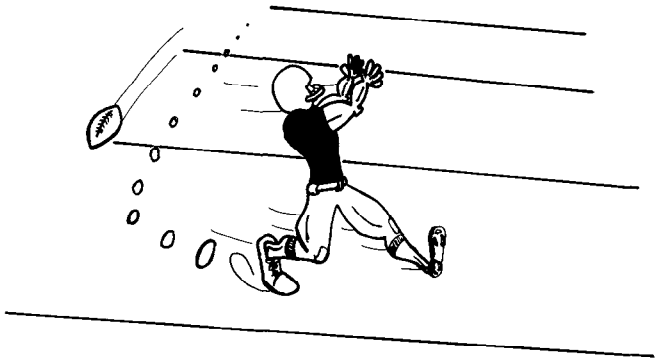
Expert Systems Technology Transfer

Construction of expert systems requires skills that are possessed by a very small number of individuals. Furthermore, the rapid prototyping development methodology makes traditional technology transfer more difficult—the systems are in a constant state of flux. As a result it is fair to say that for the foreseeable future, greater than normal responsibility will lie with the research and advanced engineering organizations to ensure successful transfer.

Based on our experience with the Dipmeter Advisor system, we can suggest some actions to ease the problem. The suggestions refer to a number of phases of expert system development—from problem choice to transfer to engineering.

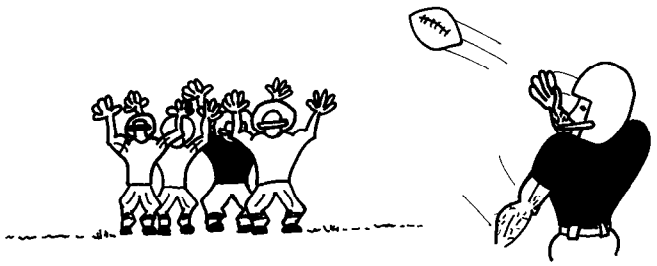
Technology transfer can be viewed as a (forward!) pass from research to engineering. In order to ensure a successful completion, both passer and receiver must have the same pass pattern in mind. From the point of view of the passer, if no open receivers are open, then a pass is ill-advised. Similarly, the passer must be sensitive to the constraints under which the receiver operates. Throwing the ball in the general area and hoping that a receiver will appear to make the catch is also ill-advised. From the point of view of the receiver, once the ball has been caught, it is his responsibility to move on down the field.

From our football analogy, as represented in Figures 5, 6, and 7, we can take away a number of useful suggestions. First and foremost, for a research organization, constructing demonstrations or prototypes and simply throwing them to engineering isn't enough. The engineering staff need to be aware of the design desiderata, the false starts, the simplifications and approximations made in the interests of expediency, the interactions between components, and so on—a host of insider information. Furthermore, once in the engineering organization, there must be committed and capable receivers to carry the project toward commercial



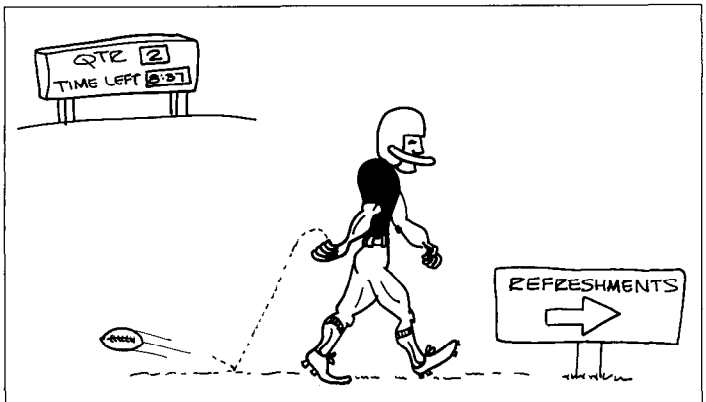
Research and Engineering do not have the Same Pattern in Mind

Figure 5.



Research does not appreciate the Constraints under which Engineering operates

Figure 6.



Engineering is not fully committed to a Successful Transfer.

Figure 7.

deployment.

This need argues for early, on-site involvement by engineering (and perhaps field) personnel in the research laboratory. Such involvement is not atypical for development projects, of course. It is especially important for expert systems technology, however, given its relative immaturity and the lack of trained specialists.

These people can also help to ensure that the completed system is well-integrated into the overall complex of systems in use by the organization as a whole. Furthermore, they are probably in a much better position than the researchers actually to effect this integration.

We have made the point several times that expert system development is an incremental process. Even so, it is worth reemphasizing. Without domain expertise there is no system! Over the course of a commercial expert system development effort, it is necessary to maintain the commitment of at least one domain expert—personal commitment and corporate commitment. The importance of the latter should not be overlooked. Experts make money for their employers and time devoted to a development project may well have a short term negative impact on their normal productivity. Hence, management needs to support expert involvement.

One way to ensure this commitment is to work on problems that the experts actually want solved!

We have found it useful in easing our interactions with both engineering and field personnel to deal in what we might call value-added systems. By this we mean that the ultimate user gets a number of advantages from using our new systems—one of which is symbolic inference. This advantage is evident in the Dipmeter Advisor system. Even if the interpreter never uses the inference machinery, he still derives some benefit from the system—namely, a powerful interactive log interpretation environment. In addition, he is always in control of interactions with the system—he can interactively control the system's inference procedure. This option has the effect of giving him an environment in which he can explore the ramifications of his own hypotheses about the local geology in addition to acquiring access to some of the expertise of other senior interpreters.

We have also found it necessary to construct our systems in such a way that they do not have a negative impact on the standard field computing environment. As pointed out previously, personal workstations have offered real relief in this area. They have, not however been without cost—they have necessitated a relatively large investment in networking software.

For additional thoughts on the problem of moving advanced computer science technology into real world environments see (Newell, 1983). Newell makes a number of salient observations on the basis of his experience with the installation of the ZOG system on the USS CARL VINSON. One of the considerations for which he argues is flexibility. The functionality expected of a system often changes over time. It may therefore be difficult to predict what its eventual use will be. As a result, the developers of real systems are advised to avoid rigidity in their designs.

Some Observations on the Traditional Wisdom

For the remainder of this section we consider a number of maxims of expert system development in the light of our experience in the commercial environment. [See (Barstow, 1981); (Buchanan, 1982); or (Davis, 1982) for good summaries of the traditional wisdom of expert systems development.]

A common maxim of expert system development is that we should throw away the code for the Mark-I version of the system as soon as it demonstrates feasibility and get started on Mark-II. In the commercial environment, there is great reluctance to throw away code. As a result, a likelier scenario involves a series of progressive releases of the system to the expert and possibly to the engineering organization for development and use. The fact is that even though the knowledge engineer knows all too well the limitations of Mark-I, and even has ideas on how to overcome them, Mark-I may still provide some useful service. We do not yet know how to manage this type of progressive and evolutionary technology transfer.[8]

It is well accepted that expert system development is an incremental process. Usually we understand this fact to mean that the performance of the system improves incrementally. There is, however, another kind of change that may occur—namely, our experts are themselves moving targets, partially as a result of the perspective gained through experience in expert system development! This has been apparent during the Dipmeter Advisor project. The existence of tools for testing the ramifications of geological hypotheses led our expert dipmeter interpreter to try a number of approaches to stratigraphic analysis. The program was a test bed for his evolving ideas.

It is traditional wisdom that the task should be very carefully defined before the system is designed. Our experience has been that this process is quite difficult. In consonance with our comments on the rapid prototyping development strategy, it is not clear that task definition can be done in a rigorous fashion. We suggest a contingent definition—one that is clear for a time, but can be easily changed. We should note that the evolving performance of the system itself at least partially fuels changes in the task definition.

It is generally accepted that construction of the Mark-I system should be commenced as soon as one example of the intended behavior is understood. We did not obey this maxim. We now believe that we spent too much time in knowledge acquisition before actually starting to build a

[8]This is a good illustration of a conflict that can arise as a result of somewhat different goals of research and of development in expert systems. The former is concerned with continued exposition and machine implementation of human expert reasoning methods, while the latter is concerned with construction of products that utilize already understood and implemented methods

system. This activity had the effect of slowing our rate of progress. We could not move forward in formalizing the knowledge that had been gained, because we could not demonstrate in concrete terms our understanding of it.

This problem can be exacerbated by the seductive simplicity of an intuitively appealing formalism like production rules. We want a domain expert to communicate as much as possible about what he is doing in solving a problem—independent of whether or not it appears to him to fit naturally into the formalism. Difficulties can arise when the expert attempts to map his explanations directly into the formalism—perhaps at the expense of accuracy. If insufficient testing is done throughout the process of knowledge acquisition, then a misunderstanding may develop about exactly what the system can and cannot do with rules so generated.

Some of the development team also deemed themselves to have acquired more expertise than was warranted. This is a natural tendency. It was partially due to infrequent interactions with the expert. More responsibility fell on the shoulders of the system developers to organize the domain knowledge than appears prudent. This infrequency also led to a problem of validation—how to be sure that we were on the right track. On a related note, we can testify to the necessity of an adequate set of generic examples with which to test the system as it evolves.

One piece of traditional wisdom might be questioned. It is common to deal with a single expert during the development of an expert system. The perceived danger is that it is difficult enough to capture the perspective of a single expert, let alone those of a number of experts. In the particular context of dipmeter interpretation, however, it might have been useful to involve a number of experts with differing backgrounds from the outset. For example, while the rules for a first approach are most appropriately phrased by a dipmeter interpreter, the necessary geological vocabulary and structure are most appropriately obtained from a geologist. In future systems, we will attempt to synthesize these overlapping points of view.

In a similar vein, we have noted a difficulty that can arise when a single expert is used and when he provides all examples with which to test the system. When working with familiar examples, our expert does indeed appear to apply forward-chained empirical rules—kinds of compiled inferences. Recently, however, we have participated in experiments with a number of interpreters (and examples) from around the world. During these experiments we noted that all the experts exhibited a different mode of reasoning when faced with completely unfamiliar examples. They appeared to reason from underlying geological and geometric models—supplementing the rules. In some sense, this behavior is of course to be expected. However, actual evidence of a change in reasoning style by the single expert that we dealt with for the rule base development was elusive. It was complicated by the fact that he has extremely broad experience. Hence, finding a completely unfamiliar example was quite difficult. We believe that dealing with multiple experts would have provided concrete evidence of this phenomenon much sooner in the life of the project.

With regard to acceptance of the expert systems approach, our experience has been somewhat different from that of the R1 designers in that there was general relatively rapid acceptance of the ideas within our organization. From early in the project, concerns revolved almost totally around performance and utility in the problem domain.

We have seen a substantial increase in the size of the rule base (approximately tripled) and the functionality required of the system before we could consider field evaluation. McDermott has described a similar experience with R1. The size of its rule base tripled during the development phase (McDermott, 1981).

The traditional wisdom notes the importance of early construction of a flexible user interface. For the Dipmeter Advisor system the interface is graphical. It has proved invaluable in testing and user acceptance. Furthermore, as has been noted elsewhere (Buchanan, 1982), expert systems that are actually used by people trying to solve problems in their own domains of interest (as opposed to being used by researchers as vehicles for experimentation with AI techniques) must pay particular attention to human interface issues. For the Dipmeter Advisor system, it was only after we constructed a personal workstation implementation that was flexible, robust, and fast that it became possible to consider seriously testing by the Schlumberger engineering organization.

One final observation worth noting relates to the impact of an expert system on the domain experts. As has been found in other applications of expert systems (Feigenbaum, 1980) the existence of an expert system is helping to identify the real knowledge used in the field—the kind of knowledge that is rarely found in textbooks. A program that captures some of it at least gives a concrete basis for comparing the methods of different experts. As Gaschnig has noted (Gaschnig, 1982) it can also help a group to reach some form of consensus. The Dipmeter Advisor system has stimulated an examination of current dipmeter interpretation methods that promises to improve quality.

System Development Tools

We have discussed the utility of flexible programming environments and personal workstations in the development of expert systems. Powerful tools for creation, modification, and maintenance of knowledge bases and related code are also especially helpful during the design and testing of expert systems. In a sense, they are augmentations to the programming environment that further assist in the rapid prototyping approach. Such tools may also help in reducing the necessity for a tool designer on every expert system development team.

In this section we describe a small set of development tools that we are finding helpful in our current expert system

efforts. The tools are STROBE, a structured object programming system, and IMPULSE, a display oriented knowledge base editor. They are described in (Smith, 1983a),(Smith, 1983b), and (Schoen, 1983).

In recent years, interest has been considerable in structured object representation for the design of expert systems. Within this framework, a programmer can encapsulate packets of knowledge and link them together via a variety of relationships to form knowledge bases. Inheritance of properties through generalization hierarchies is standard.

Programming with structured objects offers a number of advantages. From a conceptual point of view, it is helpful to organize computations around programming objects whose internal structure explicitly reflects that of objects in the real world and whose communication with each other is via messages. It is also helpful to organize data structures according to taxonomic hierarchies and to distinguish between general classes of entity and specific individual entities. This style of programming encourages thought about the structure and interrelations between various packets of knowledge in a system.

From a programming point of view, it is helpful to encapsulate procedure definitions and data definitions. This process leads to modular code and helps prevent inappropriate application of procedures. The behavior of an object in stereotyped situations is defined within the object itself in a set of procedures specific to that object as opposed to being defined in a general procedure buried in an amorphous system. Inheritance of procedure definitions also enhances modularity and storage efficiency. It has the added benefit of simplifying the sharing of procedures.

From an expert system point of view, structured objects help to capture what we might call automatic inferences—the kind of inferences that would otherwise be made by explicit rule application (Nilsson, 1980). For when a system built with structured objects discovers that some object belongs to a known class of objects, then it immediately acquires access (through inheritance) to the body of information already known about the class.

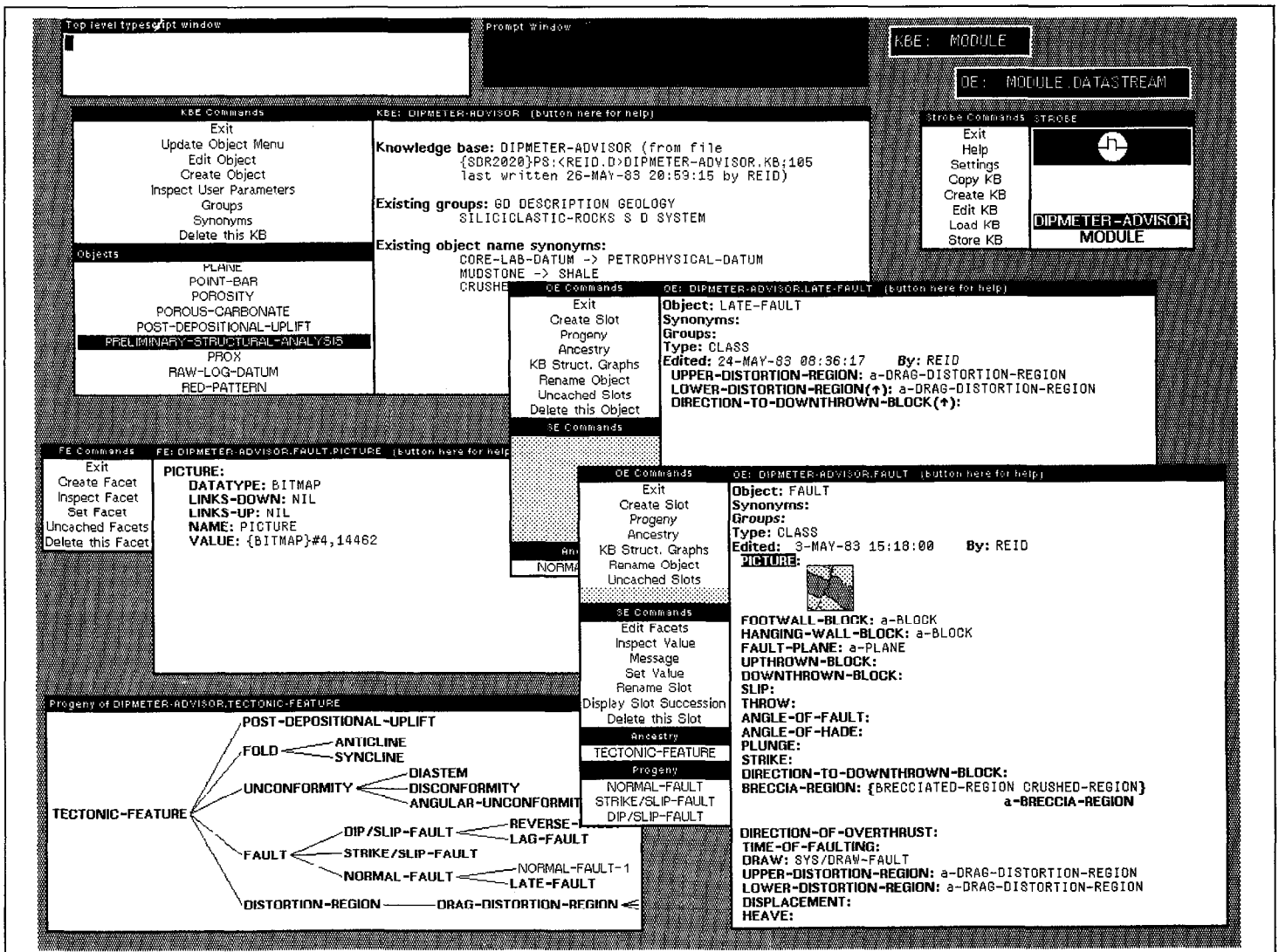


Figure 8. Sample IMPULSE Editor Screen

STROBE is a system that provides object-oriented programming support tools for INTERLISP. A STROBE knowledge base is made up of a number of interrelated objects. The characteristics of an object and its links to other objects are encoded as a number of slots. The slots themselves have structure—facets—that can be used for annotation. STROBE implements multiple resident knowledge bases, tangled generalization hierarchies, flexible inheritance of properties, procedural attachment, message-passing, and procedure invocation in conjunction with several types of data access and alteration. It offers a primitive foundation with which more complex structured object representation schemes can be constructed.

IMPULSE provides a convenient user interface to STROBE. A user still expresses his knowledge in terms of STROBE constructs, but interaction with the evolving knowledge bases and objects is via pointing and direct visual manipulation. IMPULSE enables concurrent editing in multiple contexts (*e.g.*, having several object editor windows simultaneously active) and graphical displays of inter-object relationships. Figure 8 shows an IMPULSE screen during an editing session in which parts of the tectonic feature knowledge of the Dipmeter Advisor system are being updated.

In addition to their utility to the builder/maintainer of knowledge bases, tools like STROBE/IMPULSE can assist in the transfer of expertise from domain expert through computer scientist intermediary to machine. One of the most useful roles played by the intermediary is to help provide a logical organization for the knowledge of the domain expert. This assistance is typically provided via many interactions. For each interaction, the intermediary gathers some understanding of a portion of the expert's knowledge, encodes it in a program, discusses the encoding and the results of its application with the expert, and refines the encoded knowledge. Discussion and refinement is facilitated when the knowledge is encoded in domain-specific terms and when it is presented in forms familiar to the domain expert. Our experience with IMPULSE is that its ability to simultaneously display different views of a knowledge base and its characteristic immediate feedback have enhanced interactions with our domain experts.

The tools we have discussed are characteristic of the sort that an expert system development team can be expected to use (and perhaps produce). Our focus has been tools for encoding structural relationships between packets of knowledge. They are typically combined with other tools that provide facilities for encoding and invoking heuristic rules. Tools of this sort are described in the literature [*e.g.*, OPS (Forgy, 1981), LOOPS (Bobrow, 1983), (Gorlin, 1981), and EMYCIN (VanMelle, 1981)]

Conclusions

The current Dipmeter Advisor system has provided substantial demonstration of the feasibility of using expert system techniques in commercial well-log interpretation. Additional analysis and evaluation of the system will certainly further define the the strengths and weaknesses of its approach. The experience gained to date has also helped to suggest characteristics of commercial expert system development as well as properties of a development methodology.

We have found that incremental development of expert systems within a rapid prototyping framework is a viable approach. It has also been important to bear in mind from the beginning of a commercial expert system development effort that the system will eventually be used by people who are more sensitive to utility and performance than to the novel techiques that it may embody.

Early engineering and field involvement are especially important in expert system technology transfer, given its relative immaturity and the scarcity of trained specialists. These people are also in a good position to ease the problem of integration of expert system components into more traditional software systems. The notion of value-added systems—that include symbolic inference as part of an overall package—is useful in ensuring field acceptance.

A variety of skills are required for expert system development. These skills domain expertise, expert system tool design, knowledge engineering, and traditional programming support. Not all of these skills are required throughout a development project, as it oscillates between domain knowledge expansion and redesign of the underlying framework. Furthermore, high-level development tools such as structured object representation languages and standard rule interpreters can reduce the need for tool design.

Finally, we have noted that the traditional wisdom of expert system development offers sound advice. Problems to be wary of are related to the seductive nature of a simple formalism and to the extended use of a single expert.

References

Barstow, D.R and B. G. Buchanan (1981) Maxims for Knowledge Engineering SDR AI Memo No 10, Schlumberger-Doll Research.

Bobrow, D.G. and M. J. Stefik (1983) The LOOPS Manual. Unpublished Memorandum, Xerox Palo Alto Research Center.

Buchanan, B G. (1982) New Research On Expert Systems In J. E. Hayes, D. Michie, and Y-H Pao (Eds.), *Machine Intelligence 10* New York: Wiley & Sons, 269-299.

Clancey, W J. and R. Letsinger (1981) NEOMYCIN: Reconfiguring a Rule-Based Expert System for Application to Teaching. *Proceedings of IJCAI-81*, 829-836

Davis, R., H. Austin, I. Carlbom, B. Frawley, P. Pruchnik, R. Sneiderman, J A. Gilreath. (1981) The Dipmeter Advisor: Interpretation of Geologic Signals. *Proceedings of IJCAI-81*, 846-849

Davis, R. (1982) Expert Systems: Where Are We? And Where Do We Go From Here? AI *Magazine,* Vol. 3, No. 2, 3-22.

Feigenbaum, E. A. (1980) Knowledge Engineering: The Applied Side Of Artificial Intelligence. Tech Rep. STAN-CS-80-812 (HPP-80-21), Dept. of Computer Science, Stanford University.

Forgy, C. L. (1981) OPS5 User's Manual. CMU-CS-81-135, Dept. of Computer Science, Carnegie-Mellon University.

Gaschnig, J. (1982) Application of the PROSPECTOR System to Geological Exploration Problems. In J E. Hayes, D Michie, and Y-H Pao (Eds.), *Machine Intelligence 10.* New York: Wiley & Sons, 301-323

Gershman, A (1982) Building A Geological Expert System For Dipmeter Interpretation *Proceedings of the European Conference on Artificial Intelligence*, 139-140.

Gorlin, D ,F Hayes-Roth, S. Rosenschein, and H. Sowizral (1981) The Language Reference Manual N-1657-ARPA, The Rand Corporation, Santa Monica, CA.

McDermott, J (1981) R1: The Formative Years AI *Magazine*, Vol 2, No. 2, 21-29

Newell, A (1983) An On-Going Case Study In Technological Innovation In L. S Sproull and P. D Larkey (Eds.), *Advances In Information Processing In Organizations* Greenwich CT: JAI Press.

Nii, H.P., E. A. Feigenbaum, J. J. Anton, and A. J. Rockmore (1982) Signal-to-Symbol Transformation: HASP/SIAP Case Study. AI *Magazine*, Vol. 3, No 2, 23-35.

Nilsson, N. J. (1980) *Principles of Artificial Intelligence.* Palo Alto: Tioga Publishing Co

Reboh, R. (1981) Knowledge Engineering Techniques and Tools in the PROSPECTOR Environment. Technical Note 243, SRI International, Menlo Park, CA.

Schoen, E and R. G. Smith (1983) IMPULSE, A Display-Oriented Editor for STROBE. *Proceedings of AAAI-83*, 356-358

Schlumberger (1981) Dipmeter Interpretation: Volume I—Fundamentals. 1981.

Sheil, B (1983) Power Tools For Programmers. *Datamation*, 131-144.

Shortliffe, E H (1976) Computer-Based Medical Consulations: MYCIN. New York: American-Elsevier

Smith, R. (1983a) Structured Object Programming in STROBE Research Note SYS-84-08, Schlumberger-Doll Research, (revised March, 1984.)

Smith, R. (1983b) STROBE: Support For Structured Object Knowledge Representation *Proceedings of IJCAI-83*, 855-858

VanMelle, W., A C Scott, J. S. Bennett, and M. Peairs (1981) The EMYCIN Manual Tech Rep. STAN-CS-81-885 (HPP-81-16), Dept. of Computer Science, Stanford University.