ACE: An Expert System Supporting Analysis and Management Decision Making[1]

Salvatore J. Stolfo
Columbia University
and
Gregg T. Vesonder
Bell Laboratories

CUCS-33-82

October, 1982

[1]Partial support was provided by the Office of Naval Research under contract number N00014-82-K-0256.

Table of Contents

List of Figures

Abstract

ACE, a system for *A*utomated *C*able *E*xpertise, is a *Knowledge-Based Expert System* designed to provide trouble-shooting reports and management analyses for telephone cable maintenance in a timely manner. Many design decisions faced during the construction of *ACE* were guided by recent successes in expert systems technology, most notably R1/XCON, the Digital Equipment Corporation Vax configuration program. The most significant departure from "standard" expert systems architectures is *ACE*'s use of a conventional data base management system as its primary source of information. Its primary sources of knowledge are the expert users of the database system, and primers on maintenance analysis strategies. The coupling of "knowledge-base" and "data-base" demonstrates in a forceful way the manner in which an expert system can significantly enhance the throughput and quality of data processing environments supporting business management. However, further difficult problems must be solved before the expert system approach becomes a standard technique in the data processing industry.

1. Introduction

Over the past decade, the processing power and the capacity of computers has increased dramatically, with equally impressive decreases in cost. This cost-effectiveness, spurred by rapidly increasing economic pressure for higher productivity, has caused conventional data processing systems to be pushed to their limits of operation. Designers and developers of computer systems are now being asked to provide solutions to complex problems, tasks performed mainly by highly trained human experts. This is precisely the focus of *Knowledge Engineering*: the construction of complex *Knowledge-Based Expert Systems*.

In general, knowledge-based expert systems are Artificial Intelligence (AI) problem-solving programs designed to operate in narrow "real-world" domains, performing tasks with the same competence as a skilled human expert. The heart of these systems is a *Knowledge Base*, a large collection of facts, definitions, procedures and heuristic "rules of thumb", acquired directly from a human expert. The *Knowledge Engineer* is an intermediary between the expert and the system who extracts, formalizes, represents, and tests the relevant knowledge within a computer program.

Just as robotics and CAD/CAM technologies offer the potential for higher productivity in the "blue-collar" work force, it appears that AI expert systems will offer the same productivity increase in the "white-collar" work force. As a result, Knowledge Engineering has attracted considerable attention from government and industry for research and development of this emerging technology. Of particular importance to business and government is the use of computer systems to enhance office productivity and management environments beyond the storage and retrieval functions of conventional databases.

But, are AI expert systems mature enough now to impact on current management environments? In this paper, we answer this question with a resounding "yes".

We describe *ACE*, a knowledge-based system for *A*utomated *C*able *E*xpertise, which is designed to provide support for management analysis, automating decision making for telephone cable maintenance.

The development of *ACE*, undertaken by Bell Telephone Laboratories two years ago, demonstrates in a forceful way the manner in which AI techniques can be applied to significant and practical "real-world" problems.

In our opinion, LISP, long the mainstay of AI programming, can no longer be considered too inefficient and cumbersome for "practical" applications. Since software development dominates the cost of computing systems, the flexibility and transparency of LISP significantly reduced the cost of implementing a very large and complex system. The developers of the *ACE* prototype were able to implement, test and verify the performance of the system well ahead of predicted schedules. Furthermore, the current knowledge engineering techniques (that is, the "accepted wisdom", Davis'[1981] well phrased euphemism) can be seen to have wider applicability than medical diagnosis [Shortliffe 1976], genetic engineering [Stefik 1980], elucidation of unknown chemical compounds [Buchanan and Feigenbaum 1978] and geological surveying [Duda et al. 1979].

ACE is not a consultant. It is an automatic analysis system, perusing large volumes of maintenance reports generated by personnel of the telephone system. The data provided by CRAS, a conventional data management and report generating system, contains enough detail to permit specialists to produce analyses of trouble spots in the local phone network. These analyses are subsequently used to predict future work force requirements and budgetary needs of the maintenance centers, and drive plant rehabilitation decisions. *ACE* currently produces timely summaries of its own analyses of CRAS data, permitting the specialists to focus their investigation on specific aspects of the recent repair tasks. Thus, *ACE* demonstrates a successful merger of two complementary and independent technologies: database and knowledge base systems.

Although superficially the problem domain of *ACE* appears fundamentally different from that of R1/XCON, the Digital Equipment Corporation Vax computer configuration program reported by McDermott [1981], there are enough common characteristics of both domains suggesting that the organization of R1/XCON is suitable for *ACE*'s application. Consequently, many of the design decisions faced in the development of *ACE* were guided by the methods employed in R1/XCON.

Specifically, the *ACE* inference engine is a *forward-chaining Production System*

executing the *Match* problem-solving paradigm. The declarative nature of the OPS4 production system notation [Forgy and McDermott 1977] effected the rapid development of the *ACE* knowledge base. The team of knowledge engineers, or "knowledge gatherers" (cognitive psychologists), who interviewed several human experts, had little difficulty operationalizing the acquired knowledge directly in OPS4. Subsequent development cycles of test, debug and modify were carried out with relative ease. The end result was a working prototype well ahead of schedule. An advanced version of the system will be present in the telephone operating companies in the near future.

The following sections detail the problem domain and organization of *ACE*, and describe how the development and installation of a prototype of this system in a "live" production environment was completed in a cost-effective manner. The concluding sections indicate several important problems that must be addressed by designers and developers, if their expert systems are to be moved from the prototypical laboratory environment to the live production environment.

2. The Problem Domain

In normal operation, the telephone network supports a telephone line from a residential or business site. This line is called a *cable pair*. A collection of pairs are bundled together to form the cables that hang from telephone poles, or reside underground. A collection of cables form a *wirecenter*. These three levels form the bulk of the local telephone network and the cable maintenance force concentrate their efforts at all three levels. A more detailed description of the organization of the local telephone network can be found in [Bell 1977].

A variety of electrical faults and environmental conditions can cause failure of one or more cables or individual pairs. (Insect infestations in terminal boxes, and gnawing rodents are perennial problems.) A critically important and expensive operation performed by the operating companies is general maintenance and rehabilitation of these lines.

Customer generated maintenance reports provide important information for identifying "trouble spots" within the local network. In a high-density geographic area, the logging

and tracking of failure reports has become an important and expensive data processing operation.

In order to identify trouble spots for repair and rehabilitation, many telephone companies use CRAS to monitor the maintenance of the local network on a daily basis. Highly trained analysts routinely peruse impressive volumes of data and attempt to identify spots for maintenance to prevent further disruption of service to customers.

CRAS provides a set of report generating programs each producing a specialized summary of various aspects of customer and employee generated maintenance reports. An individual record maintained by CRAS consists of numerous fields detailing a repair task reported by a customer or employee. This is referred to as a *trouble* in telelphone company jargon. Many distinct CRAS records representing troubles may refer to the same pair or cable, indicating a potential chronic problem.

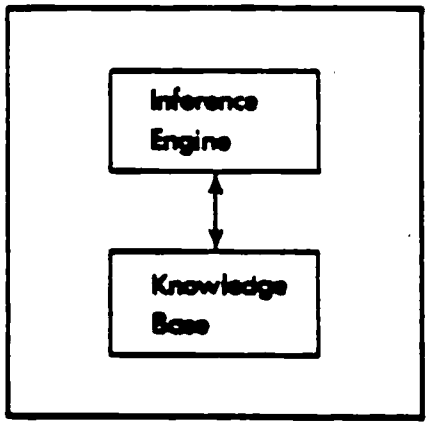
The large volume of detailed information allows the human expert to perform a great many analyses and to make an informed selection of candidates for rehabilatative maintenance. However, the limited number of specialists available, and the size of the database inhibits the timely analysis and reporting of persistent problem areas which require rehabilitation. The backlog delays the assessment of future work force needs, and the selection of prospective areas for maintenance. The approach of installing an expert system as an adjunct to the CRAS database facility, assisting management decision making, was proposed as a solution to the long-term problem of timely and accurate selection of areas for rehabilitation.

3. The System Organization

Knowledge-based expert systems have been constructed, typically, from two loosely coupled modules, collectively forming the *problem-solving engine* (see figure 3-1). The *knowledge base* contains all of the relevant domain-specific information permitting the program to behave as a specialized, intelligent problem-solver. Much of the research in AI has concentrated on effective methods for representing and operationalizing this knowledge. The representations that have been proposed have taken a variety of forms including purely declarative-based logical formalisms, "highly-stylized" rules or

productions, and structured generalization hierarchies commonly referred to as semantic nets and frames. Within *ACE*, the knowledge base is implemented in rule form, to be detailed shortly.

Figure 3-1: Organization of the Problem-Solving Engine of a "typical" Expert System.



The *inference engine* is that component of the system which *controls* the deductive process. The earliest AI problem-solvers implemented an iterative branching technique searching a large space of problem states. In contrast, the state-of-the-art expert systems separate the control strategy from an inflexible program, and deposit it in the knowledge base along with the rest of the domain-specific knowledge. Thus, the problem-solving strategy becomes domain-dependent, and is subject to the same methods of acquisition and deductive manipulation as are facts and assertions.

ACE has been designed with the same prescription. However, a third component has been added: a data base (see figure 3-2).

Over the past decade or so, database technology has progressed from "dull witted" systems providing facilities for the efficient manipulation of large data files to current state-of-the-art systems which offer abstract data models to facilitate "semantically-based" retrieval functions. However, few, if any, offer any ability to deduce new data

from old; at best, aggregate data functions can be applied but deductions from raw data are not possible.

On the other hand, knowledge based expert systems have been implemented as consultants to specialists in very narrow domains, embodying limited amounts of knowledge. These systems are designed as problem-solvers: they deduce new information in highly complex domains, but do not, in general, operate with a massive amount of data.

ACE is an attempt to merge the two technologies, which may synergistically benefit numerous applications. Kellog [1982] reports on the Knowledge Management I (KM-I) system, whose organization is very similar in scope to that represented by *ACE*. However, KM-I is designed as an intelligent front-end, interfacing the database and user with an English-like language query facility. The knowledge base within KM-I essentially interprets "high-level" queries, and responds in comparable form. Thus, the database is much easier to use, and the access methods are closer to the language of the application area.

In contrast, in *ACE* the expert system is the user! Tailored to solve a single problem or class of problems, the knowledge base component of *ACE* automates the tasks the database has been designed to support. Thus, the expert system not only answers the questions, but poses them too.

In the following sections we detail the organization, implementation and operation of *ACE* in a live production environment.

3.1. Details of the Implementation

In *ACE* the knowledge base and inference engine are implemented entirely in Franz LISP [Foderaro 1979] and the OPS4 Production System language, running on a DEC VAX-11/780. Supporting routines provided for *ACE*, including CRAS interfacing and electronic mail facilities (to send *ACE* output to selected user mailboxes) are implemented in resident UNIX[2] software. Communications with a host CRAS system is

[2]UNIX is a registered trademark of Bell Telephone Laboratories.

supported by a network program, UUCP, with UNIX managing all levels of system operation. (C is the root of all systems.) The gross system level organization is depicted in figure 3-2.

The reader will note that a human user is not necessary to run *ACE*. *ACE* is stand-alone, awakened each night by a UNIX timer facility to perform its function at the close of a day's transactions. The only record an *ACE* user sees of its operation (besides system logs) is a message specifying line failures and trouble spots it discovered during its nightly run (see figure 3-3).

Upon completing its analysis of the day's events, *ACE* also performs analyses of the history of trouble reports in the plant in view of the events of the previous day. At the close of its operation, *ACE* updates all of its information proposing "partial hypotheses" with supporting evidence, about possible future events that seem worthy of further exploration as new data becomes available. Thus *ACE* is temporally-based, and data-driven. Its state of knowledge is dependent on past events, and predictions of future events, and the verification or refutation of those predictions.

3.2. Details of the Problem-Solving Engine

Within *ACE*, the corpus of knowledge about wirecenters, CRAS data and commands, and anlysis strategies is embodied by an OPS4 Production System program.

3.2.1. Production Systems

In general, a *Production System* is defined by a set of rules, or *productions*, which form the *Production Memory*(PM), together with a database of assertions, called the *Working Memory*(WM). Each production consists of a conjunction of *pattern elements*, called the *left-hand side* (LHS) of the rule, along with a set of actions called the *right-hand side* (RHS). The RHS specifies information that is to be added to (asserted) or removed from WM when the LHS successfully matches against the contents of WM. The

Figure 3-2: The organization of *ACE* and supporting systems.

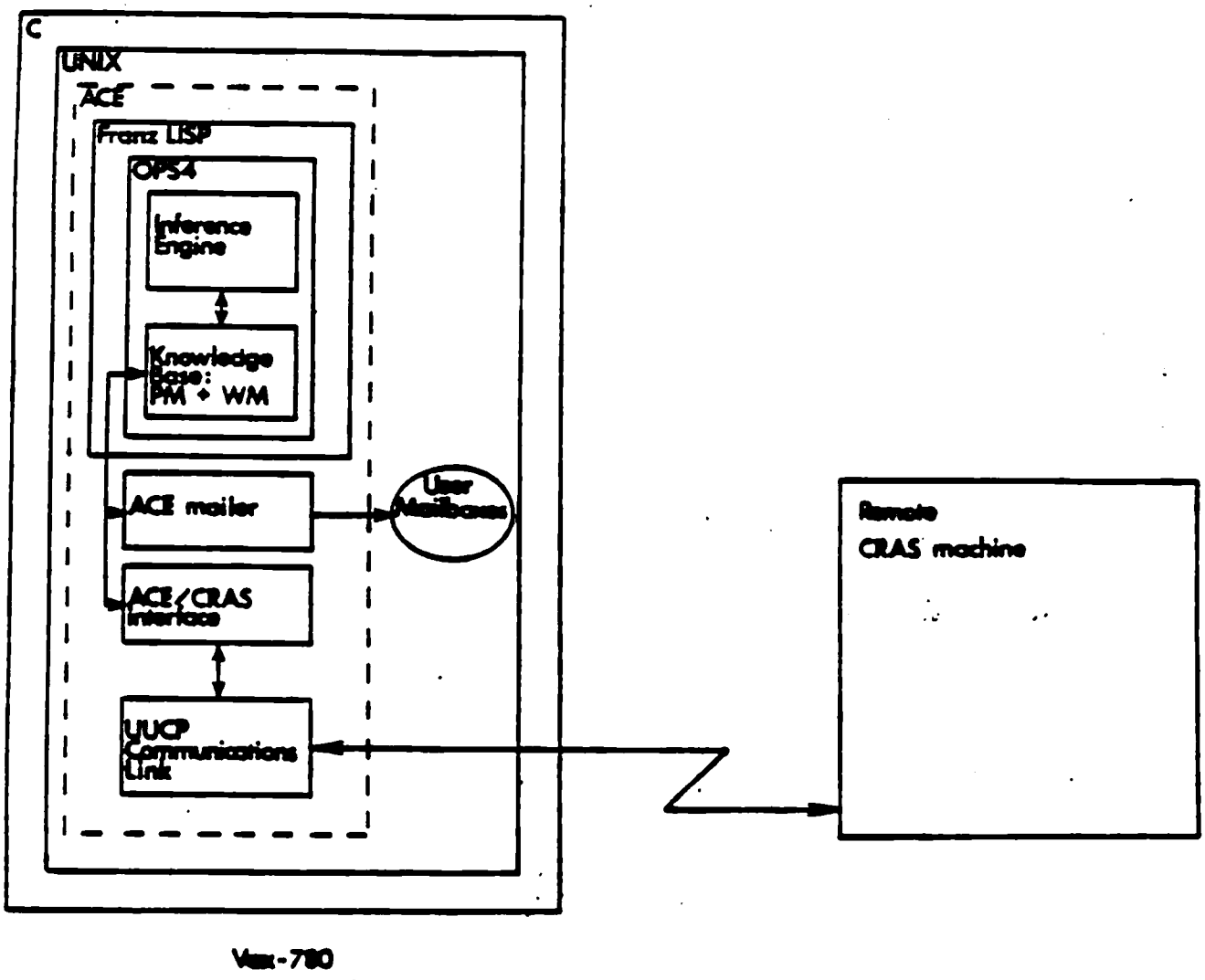


Figure 3-3: A sample *ACE* message produced by real data.

```
Ace 1.0 output message follows:

**********ACE REPEAT MESSAGE**********
In cable 47 wirecenter 999550 1 cable
troubles were reported yesterday.  There
were 15 cable troubles in the cable in
the past thirty days.  4 pairs were in
the complement range from 600 to 700.  75
percent of the addresses were williams.

The most common disposition code was 0436.
This indicates the presence of a problem at one
or more crossboxes in this cable.  In all
likelihood it is one crossbox.  This crossbox
should be cleaned, rehabilitated and properly
closed up, to prevent further troubles, or it
should be replaced if it has deteriorated to a
point where that is necessary .

The detail data for cable 47 is:

pair    disp    ccode   ctime   ctta    troub-address
827*    436     0       1021045 50034   17 williams
848*    436     0       1290945 50175   17 williams
855*    400     0       1141045 53021   17 williams
859*    436     0       1301430 50184   4 watson
838     436     0       1111930 50128   b box williams st
879     431     0       1221430 50161   austin @ kenwood
891     431     0       1071100 50113   p2 9 kenwood
1006    400     0       1121330 53010   17 williams st
1031    436     0       1141000 65002   87 pearl
1101    401     0       1241600 25036   pr18 upland
1509    431     0       1080930 50116   35 calender
1607    431     0       2011445 50193   calender 1st fr puta
1607    431     0       1072150 50110   131 huron ave
1695    433     0       1061500 25003   p@ 35 calendar
2513    401     0       1181100 25021   p 19 pleasant st


End of ACE transmission.
```

RHS can also specify operations to be performed at the UNIX command (shell) level.

In operation, the PS repeatedly executes the following cycle of operations:

1. *Match*: For each rule, determine whether the LHS matches the current environment of WM. All matching instances of the rules are collected in the *conflict set of rules*.

2. *Select*: Choose exactly one of the matching rules according to some predefined criterion.

3. *Act*: Add to or delete from WM all assertions specified in the RHS of the selected rule or perform some operation in the UNIX shell.

In OPS4, data elements in WM have the form of arbitrary LISP list structures. Both the LHS and RHS of OPS4 productions are conjunctions of pattern elements composed of constants, embedded sublists of pattern elements, and ***existentially quantified*** pattern variables. User defined LHS predicates are provided, which can test for a variety of conditions of WM, as are user defined RHS actions, which may perform functions beyond modification of WM.

An English language equivalent of an *ACE* production rule is presented in Figure 3-4.

Figure 3-4: An Example *ACE* production.

IF a range of pairs within a cable have generated
a large number of customer reports
ANDIF a majority of the work on those pairs was
done in the terminal block

THEN look for a common address for those repairs.

During the selection phase of PS execution, OPS4 provides *conflict resolution strategies* based on the *recency* of matched data in WM, as well as syntactic discrimination. Rules matching data elements that were more recently inserted in WM are preferred, with ties decided in favor of rules that are more specific (i.e., have more constants) than others.

The mode of operation of OPS4 has been referred to in the literature on Production Systems as *forward-chaining, data-driven* execution. As data is deposited in or removed from WM, instances of matching rules are inserted in or deleted from the conflict set of rules. The action of the RHS of the selected rule forms a new conflict set on the next cycle of execution. Thus, the initial contents of WM, and subsequently, the data and flow of data in WM drives the sequence of productions selected for execution.

In contrast, *backward-chaining goal-directed* execution, typified by MYCIN-like systems [Shortliffe 1976, Davis 1976], operate by "unwinding" the rules backwards beginning with a goal to be achieved or satisfied. When a goal (or some desired state of WM) is asserted, all rules whose RHS mention the goal in question are executed in reverse. The constituent elements of the *LHS* of each rule are proposed as subgoals to achieve. Thus, each LHS contributes a conjunctive set of subgoals (AND goals) while the entire set of relevant rules collectively contribute a disjunctive set of subgoals (OR goals). The resulting AND/OR goal tree exhaustively generated in this fashion is terminated with primitive goals achieved by the presence or absence of specific conditions of the initial WM.

3.2.2. Characterizing the Problem Domain

The forward-chaining, data-driven approach typified by R1/XCON was chosen for *ACE* since cable analysis is primarily a bottom-up, data-driven task.

Using the taxonomy of problem domains described in [Stefik et al. 1982], we classify the telephone cable maintenance problem in the following way:

- The large volumes of data are temporally-based. The analyses produced by *ACE* are dependent on the frequencies of failures occurring over time. Thus, the analyses performed by *ACE* are data-driven.

- The data are reliable for the task. There are few errors, if any, and no noisy data. The CRAS data base provides enormous detail containing most of the relevant information about the cable plant.

- The knowledge of the domain is reliable. When *ACE* discovers a persistent problem, no new information concerning past events will force it to retract from its position.

- The search of the CRAS database is exhaustive, but the knowledge is reliable enough to provide significant prunning of the space of possible conclusions.

- Lastly, the main focus of the problem-solver is the quality of the inferences it makes. The design of *ACE* was driven by the necessity of producing the same analyses as its human counterparts.

These characteristics render the *Match* [Newell 1969] problem-solving strategy the ideal paradigm for *ACE*. As in R1/XCON, *ACE* requires no backtracking search of a large problem space, rather its primary task is divided into a fixed sequence of subproblems:

- the data are filtered without loss of important detail,

- requests are generated for more appropriate data,

- aggregate database functions are applied,

- alarming conditions are noted and diagnosed (if possible),

- and finally its findings with supporting data and recommendations are mailed to the appropriate users.

3.2.3. The structure of the Knowledge Base

Although no structure is provided (or imposed) by OPS4 PM, the set of approximately 100 productions and approximately 50 related LHS and RHS functions in *ACE*'s knowledge base can be loosely organized into subsets of related rules which collectively perform the analysis.

A set of productions performs short term analysis by examining the flow of trouble reports on a daily basis. If troubles are reported for a cable that has no previous history of troubles then information is retained that indicates that this cable may soon require attention.

When new failures are reported for a cable with a history of persistent problems, *ACE* requests further detailed reports from CRAS, along with a list of standardized procedures used to repair the type of troubles reported. This information is used to deduce:

1. whether the repair task done on that cable suggests that preventive maintenance may reduce future troubles,

2. if preventive maintenance is required then what type is likely to be appropriate,

3. and, if possible, where the rehabilitation should be done.

Thus, *ACE* not only identifies trouble spots, but suggests how to repair them too.

Each of these aspects of short-term analysis requires substantial deductive power. For example, locating the physical place where rehabilitation should be performed is a very difficult problem. This is magnified by the fact that the employee reporting the trouble is permitted to enter the site of the failure in a free textual format. Subsequently, the CRAS record of the failure would contain an entry that not only may be inaccurate but also subject to typographical error and capricious abbreviation style.

For instance, "WASH 5" might refer to the same location as "WASHINGTON AND FIFTH". "CALENDER STREET" might refer to the same location as "CALANDAR". Thus, *ACE* also contains productions and associated LISP functions encoding heuristics to estimate whether two different addresses refer to the same general location.

The main sources of knowledge for the short term analysis are

1. textbook knowledge obtained from primers on telephone cable analysis,

2. expert advice from the developers of CRAS,

3. expert advice from theoreticians of cable analysis both in Bell Telephone Laboratories and in the local operating companies,

4. local analysts from the operating companies and users of CRAS who perform the actual analyses.

Another portion of *ACE* PM contains a set of productions which know how to communicate with CRAS. Based on requests for more data generated by other analyses, these productions assemble the appropriate CRAS commands and parameters and then monitor the resulting data stream retrieved from CRAS. The actual transmission of requests and data is handled by a UNIX communications program accessible from within OPS4.

Finally, a set of rules assemble the appropriate message about the day's events recognized by the system, and call on the UNIX mail facilities to deliver them to the appropriate users. *ACE* knows the target of each message based on the relative importance of the message to the user. This information is contained in WM and is user modifiable.

4. In defense of OPS4 and LISP

The lack of structure of OPS4-like representations is viewed by some researchers in AI as an impediment to knowledge organization and acquisition. We strongly disagree with this assessment. The very lack of structure permitted the developers to experiment with many different ways of describing and representing the same piece of knowledge. Eventually, this lead to a representation that was felt to be a natural fit to what was expressed by the human experts. Frame-based [Minsky 1975] and backward-chaining PS approaches were investigated very early but lead often to confusing and conflicting representations. The domain knowledge was continually manipulated to fit the representation!

The declarative nature of the OPS4 production system language was suitable for recording what was being discovered about cable maintenance. Thus, the "openess" and modifiability of PSs [Rychener 1976] and declarative representations, in general, substantially accelerated the acquisition task. Only after several person-months of effort, when the knowledge-base reached a stable level of competence, did it become clear how to organize and forge the knowledge into a network of frames, or some other structured representation formalism. It is also clear now how to represent many of the analysis strategies using a backward chaining or frame-based formalism. However, the program is robust in its present form and continues to be refined without resorting to a major effort to change representational formalisms.

The interactive nature of LISP programming and debugging environments enhanced programmer productivity. The flexibility of LISP was well appreciated; especially when such earlier unforeseen tasks such as address matching were undertaken. It is believed that the system was implemented in much less time when contrasted with more traditional approaches to building large scale systems. In addition such a system would

not be as easily conceived using current data processing tools and methodologies.

By way of summary, the programming environment provided by OPS4 (and LISP) had a positive impact on the organization of the knowledge-base. The openess, modifiability and extensibility of the PS formalism greatly enhanced the ease with which old knowledge was updated and new knowledge was assimilated. *ACE* was implemented relatively quickly and displayed an admirable level of competence on its initial tests in a live environment.

5. Testing *ACE*

ACE has been field tested continually since the spring of 1982 by a long-distance connection to a remote CRAS system. The local analysts report that they are very satisfied with *ACE*'s analyses. Although *ACE* had not discovered any problems unknown to the human users, it did discover them quicker and missed neither the obvious nor the subtle. The praise of the analysts was outwayed only by their enthusiasm for permanent installation of the software.

The execution performance characteristics of the system are also very encouraging. Each nightly run on the cable maintenance records for a large metropolitan area consisting of about 400,000 lines has averaged only about one hour of Vax-11/780 CPU time (which can confidently be improved). On average, 5000 production invocations are sufficient for a complete run. New analysis tasks are under development to expand the scope of *ACE*'s grasp of cable maintenance.

6. Important Issues for Expert Systems

ACE is a representative of a new technology, generally unfamiliar to end-users of data processing systems and mainstream professional programmers. Our experiences in developing and maintaining the system have illuminated several important problems that must be addressed by developers and designers of expert systems if their commercial efforts are to be successful.

6.1. The Scope of Expert Systems

Expert systems are not solely applicable to tasks requiring years of study and field experience. There are numerous opportunities for developing expert systems in more mundane areas of expertise that do not require tools at the "cutting edge" of our technology. Tools that have been developed in the last decade (such as OPS4) provide ample opportunity to create large scale systems that will be profitable for many industries. We believe such opportunities will increase the amount of research on expert systems and related areas of AI and encourage the use of more advanced tools in the business data processing environment.

6.2. User Interaction with the Knowledge Base

Davis [1976] has taught us how to maintain, debug and extend a large knowledge base, but are these techniques adequate and should they be available in a live production environment? A distinction needs to be made between the expert contributing to the knowledge base and the eventual end-user. Can casual users be entrusted with the task of modifying and debugging the very heart of an expert system, its knowledge base? If given this ability, will incomplete and incompetent knowledge grow and fester over several months of modification?

From the point of view of the supplier, the real value of its marketed system, subject to proprietary constraints, will be the knowledge and the representation of that knowledge in the system. Should a casual user (or worse, a competitor) be privy to the innermost secrets of an expert system? How could this be prevented if the user were permitted to modify and debug the knowledge base?

Consider the problem *ACE* faces in locating the physical site of a failure. Each installation of *ACE* cannot be established simply by setting switches and parameters, it must be taught some of the characteristics of the local cable environment. Therefore, the knowledge base must be opened to the user community to some degree. However, ACE, as well as any commercial expert system, must not only be able to acquire new knowledge and explain itself effectively, but also it must know how much to say (or not say) at the appropriate level of detail: it must be able to protect itself.

6.3. Traditional Issues in Mature Operations Support Systems

Most of the research and development of expert systems has focussed on the creation of the software. The maintenance of the implementation code also poses serious problems that have been neglected by most expert systems projects. Most suppliers of software provide maintenance agreements backed by a large complement of system programmers or software engineers performing that maintenance. Currently, most software engineers are proficient in "conventional" languages and structured design techniques, few though are skilled in LISP programming, Artificial Intellegence techniques and Knowledge Engineering. Consequently, a costly training effort is demanded to have adequately skilled maintenance personnel for expert systems.

An exception to this rule is the DEC R1/XCON maintenance team. Digital Equipment Corporation has established a group of people responsible for maintaining and extending the expert system originally developed by McDermott [1981]. This group is in close contact with McDermott and Forgy of Carnegie-Mellon University and serve as a model for future forays into applications of expert systems.

And what if expert systems advance to the point where they may include self-modifying knowledge-bases? Operating continually in an environment which may provide a mass of detail with intricate subtlety, will the original designers be able to maintain an ever changing knowledge base? Will the system be able to explain its new knowledge effectively? Indeed, will they be able to recognize the old familiar knowledge base at all? When considering *ACE*'s problem, there are many ways it could improve its performance by learning more about the local cable environment. (For instance, Ace could learn about persistent problems in a particular area and differentiate between these problems and novel one-of-a-kind failures.) Thus, as the technology progresses, the problems facing the development and maintenance team will be even more complex.

Many of these considerations led to our decision of designing *ACE* as a stand-alone "batch" system. Direct user interaction would have required sophisticated front-ends. For example, a natural language component could provide English-like input and explanation facilities similar to KM-I. Each component would have contributed significantly to development costs.

Rather, we concentrated on producing a practical and useful system performing a

necessary task as quickly as possible. The expertise embodied by *ACE* has immediate need and is cost-effective for its intended customers.

However, as noted, eventual user interaction will be demanded. As the number of delivered systems grows, the maintenance of each knowledge-base cannot be performed by an already overworked development and maintenance team. Facilities to provide a limited form of knowledge base maintenance, and acquisition by local users are included in the future plans for the development of *ACE*.

Laying the foundations for solving these problems in general is the focus of our current research efforts. We expect to report on the experiences of permanently installing *ACE* in a live environment, and some of the attempts to solve the maintenance problem in the near future.

References

Bell Telephone Laboratories 1977. *Engineering and Operations in the Bell System.*

Buchanan, B. G. and Feigenbaum, E. A. 1978. DENDRAL and Meta-DENDRAL: Their applications dimension. *Journal of Artificial Intelligence*, 11:5-24.

Davis, R. 1976. *Applications of meta-level knowledge to the construction, maintenance and use of large knowledge bases*. Rep. No. STAN-CS-76-552, Computer Science Dept., Stanford University.

Davis, R. 1982. Expert systems: where are we and where do we go from here, *IJCAI 7*, (Invited lecture).

Duda, R., Gashnig, J. and Hart, P.E. 1979. Model design in the PROSPECTOR consultant system for mineral exploration. In D. Michie (Ed.), *Expert systems in the micro-electronic age*, Edinburgh University Press, 153-167.

Foderaro, J.K. 1979. *The FRANZ LISP manual*. Computer Science Dept., University of California, Berkeley.

Forgy, C. and McDermott, J. 1977. OPS, a domain-independent production system language. *IJCAI* 5, 933-939.

Kellog C. 1982. Knowledge management: a practical amalgam of knowledge and data base technology, *AAAI-82*, 306-309.

McDermott, D.V. 1981. R1: the formative years. *AI Magazine* 2:21-29.

Minsky, M. 1975. A framework for representing knowledge. In P. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill, 211-277.

Newell, A. 1969. Heuristic programming: ill-structured problems. In J. S. Aronofsky (Ed.), *Progress in Operations Research*, John Wiley and Sons, 361-414.

Rychener, M. D. 1976. *Production Systems as a programming language for artificial intelligence applications*. Computer Science Dept., Carnegie-Mellon University.

Shortliffe, E. H. 1976. *Computer-based medical consultations: MYCIN*. New York: American Elsevier.

Stefik, M. 1980. *Planning with constraints*. Rep. No. STAN-CS-80-784, Computer Science Dept., Stanford University.

Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F. and Sacerdoti, E. 1982. *The organization of expert systems: a prescriptive tutorial*. Xerox Corporation Report, Palo Alto Research Center.