Reset reproduction of CMU Computer Science report CMU-CS-84-116. Censor-edited version published in *AI Magazine*, Vol. 5, No. 2, pp. 59-63 (1984), errata, Vol. 5, No. 4, p. 78 (1984). Reprinted July 1994. Current address: MIT Laboratory for Computer Science, Cambridge, Massachusetts.

Expert Systems Without Computers
Or
Theory and Trust in Artificial Intelligence

Jon Doyle

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract: Knowledge engineers qualified to build expert systems are currently in short supply. We propose that production of useful and trustworthy expert systems can be significantly increased by pursuing the idea of *articulate apprenticeship* independent of computer implementations. Making theoretical progress in artificial intelligence should also help.

February 14, l984

I thank Jaime Carbonell, John McDermott, Joseph Schatz, and Derek Sleeman for helpful discussions and comments. This research was supported by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539. The views and conclusions contained in this document are those of the author, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Government of the United States of America.

Expert systems and their proponents have caused a revolution in the way we think about work, skill, and their possibilities for automation. This revolution is very important. We now actively seek out tasks for automation that would never have been considered previously. It seems clear that the work of our society and industry includes many economically important (if often mundane) tasks whose automation may be possible with the new techniques. Indeed, this embarrassment of riches has produced a shortage of knowledge engineers trained in constructing expert systems from the current toolkit of knowledge engineering techniques, languages, and systems, so that many worthwhile possibilities go unattended for lack of trained manpower. This bottleneck may not be inevitable, however. The following attempts to clarify the roles that computers and knowledge engineers play in building expert systems, in order to pin down the bottleneck and the possibilities for overcoming it. Our conclusions are that much progress may be possible with articulate human experts and self-conscious human apprentices before one needs to turn to computers or to knowledge engineers, and that the degree to which this may be done depends in part on the level of theoretical understanding in artificial intelligence. If these conclusions are true, the shortage of knowledge engineers may not be as significant as it seems, and might be ameliorated more quickly and effectively by employing readily available human experts and novices to rough out preliminary knowledge bases than by attempting to educate large numbers of knowledge engineers in the current fashion.

Articulate Apprenticeship: the essence of knowledge engineering

> Experience has also taught us that much of [his] knowledge is private to the expert, not because he is unwilling to share publicly how he performs, but because he is unable. He knows more than he is aware of knowing. (Why else is the Ph.D. or the Internship a guild-like apprenticeship to a presumed "master of the craft"? What the masters really know is not written in the textbooks of the masters.) But we have learned that this private knowledge can be uncovered by the careful, painstaking analysis of a second party, or sometimes by the expert himself, operating in the context of a large number of highly specific performance problems. (Feigenbaum, 1977)

Although many texts on knowledge engineering stress understanding of data-structures, inference procedures, and skills in manipulating them, as the quoted passage suggests, the key idea in the practice of knowledge engineering is the very old one of apprenticeship. Let us recall how the world of master craftsmen, journeymen, and apprentices worked in the guilds of yesteryear. The master cobbler, say, would take an ignorant apprentice and demonstrate the construction of a shoe, perhaps with a few comments about his actions. The apprentice then attempted to duplicate the feat. But being an ignoramus, having been fascinated by the master's gold ring instead of by his awl, and having been thrown into a daydream about his girlfriend by the master's remark about the need for supple hides, he completely botches the intended shoe. The master beats and curses the lout, and demonstrates the other shoe, perhaps making special note of the places where the apprentice made errors. After enough repetitions of these steps, the apprentice becomes a journeyman. At this point he is moderately competent, but more important, has learned something about how to criticize his own work, so that he can improve on his own without requiring the attentions of the master to analyze his errors. If he later gets good enough, he is rewarded with the "assistance" and fees of his own apprentices.

Progress has been made since the twelfth century. The most important new twist on this old idea is that of *articulate apprenticeship*. Instead of relying on largely mute exchanges of performances, we now appreciate the value of masters who try to explain more of what they do, so that the apprentices need not struggle as much trying to perceive what is going on, and apprentices who explain why they did what they did, so that the master can better understand and correct their ignorance and error. In articulate apprenticeship, the master need not actually do anything except order, explain and criticize, since little burden of perception is trusted to the apprentice. Instead of making demonstrations, the master just tells the apprentice rules for doing things (diagnosing diseases, interpreting squiggles on charts, guessing market behaviors) and gives the novice test cases to try out. The novice still botches the task, but explains in detail what he did in terms of the rules he followed. The master examines these explanations, and suggests further rules or changes to old ones to overcome the problem, and again they repeat these steps until the apprentice becomes competent. In some cases, rules can be provided for self-diagnosis. Altogether, these form the basis for how-to, worked-problem, and programmed-instruction books, aids to learning unheard of in the time of the guilds. But now, as then, we have no sure way of making masters from journeymen.

The relevance of this story to the case of expert systems should be clear. Here we try to force or seduce a human expert into articulating his rules of thumb. (For the purposes of this note, we will say "rule" to mean facts, procedures, etc. as well.) We get the most ignorant apprentice possible (a computer) to interpret these rules on a corpus of cases. Then we have the expert suggest changes to the rules based on explanations of the behavior on cases. Iterative improvement is the path to perfection here as well. If the task is suitable, eventually we wind up with a computer-based journeyman of routine competence, but with no power for self-improvement or adaptation to related tasks. Unlike the human apprentice, we can mass produce the computer and its program, so we are often happy to trade the final degrees of quality and self-perfectability for unlimited quantities of useful skills.

Our claim is that articulate apprenticeship is the essence of expert systems, and that all other issues—in particular, the computer and knowledge engineer bottlenecks—are secondary, concerned with implementation of the journeyman rather than his design and construction, concerned with computer systems that realize and facilitate apprenticeship rather than with articulation and refinement of the expertise proper. Feigenbaum states the basis of this view as "We must hypothesize from our experience to date that the problem solving power exhibited in an agent's performance is primarily a consequence of the specialist's knowledge employed by the agent, and only very secondarily related to the generality and power of the inference method employed" (Feigenbaum, 1977). Davis simply says "In the knowledge is the power" (Davis, 1983). Only experiment will tell, but if this view is even partially true, it suggests the possibility that many would-be users of expert systems may be able to rough out, possibly even perfect, their expert systems in the absence of both computers and knowledge engineers. For example, the attendees at the 1980 Workshop on Expert Systems (the writers of *Building Expert Systems*) were greeted by two "mystery" experts who had, by themselves, thoroughly documented their expertise. As a result, each of the knowledge engineering teams present found the programming task fairly clear and straightforward, yielding working prototype systems in just a couple days of competitive hard work. These experts may have been unusual in their motivation and effort, but I doubt they were very unusual in their ability to self-consciously explain their knowledge. If sufficiently well motivated and interested, similar accomplishments may be possible for many more experts, and there is little to lose by trying to do so, since even knowledge engineers are useless with unmotivated and uninterested experts. On the other hand, as we explain below, the difficulty of training knowledge engineers may be due to the limitations of current techniques for representing expertise, and their is little hope for improving this situation without making substantial theoretical progress in artificial intelligence. Thus it may be more fruitful to separate training in articulate apprenticeship from training in current computer systems, for the former will be useful today and tomorrow, while the latter continually become obsolete.

Benefits and Burdens of Using Computers

To judge the feasibility of building expert systems without computers, we briefly examine the role of computers and artificial intelligence in the expert system development process. Contrary to dogma, the use of computers is not an unmitigated boon, even if it may be on the whole worthwhile. Instead, the use of computers and current artificial intelligence techniques in building expert systems has some clear advantages, some clear disadvantages, and some aspects which may be viewed in either light. We examine these in turn, but do not assign comparative weights or importances.

The principal advantages of using computers in building expert systems are that they are far better bookkeepers and dunces than untrained humans. If the task requires lots of knowledge, then human interpreters of the rules bog down, either overlooking relevant rules or taking forever to find them. Slowness and clumsiness are the norm for novices. Related to this, human interpreters may be too charitable to the rules, unconsciously using common sense to fill in gaps or to correct obvious blunders rather than consciously objecting to the ambiguities. For preliminary or smallish knowledge bases, human interpreters may be quick enough and unimaginative enough to catch many of the flaws in the rules; but not as fast or as uncharitable as computer-based systems. But, just as programmers are trained to read programs literally, it might be possible to train interpreters to be similarly strict, or to use ordinary programmers as interpreters. I realize this sounds like training people to do mental arithmetic after the pocket calculator has been invented, but until the questions raised below are solved, it may be a temporary necessity. (Perhaps we could train children to interpret rule systems by replacing PAC-man with "MYC-man" (MY-san?), and let them chase conclusions rather than ghosts.)

Another advantage of computer-based systems is that they may be debugged at all hours by different experts scattered around the world, either by remote connection, or through the ease of replication and reproduction. This can be particularly important when the experts cannot be relieved of their usual responsibilities for extended periods.

Two features of computer-based construction of expert systems are often thought to be advantages, but on examination these advantages seem dubious. The first is that an implemented prototype might be polished into a production tool with little effort. This may sometimes be possible, but often it seems more sensible to use the prototype as a guide for constructing a specially crafted production version, where virtues like speed, size, and robustness take precedence over virtues aimed solely at facilitating apprenticeship. We expect different things of journeymen and apprentices. Perhaps some day we will have compilers that condense expert systems into microprocessors, but until then, the need to take this step manually means that having the implemented prototype may not speed the implementation of the production version.

Second, the use of formal knowledge representation languages for expressing information instead of natural languages and jargon is often thought to offer hygenic benefits, especially in accentuating the uncharitability of the articulation and interpretation processes. This would be a definite advantage if current systems of representation were better. But as things stand, lack of knowledge about what good representation systems should look like suggests that some large portion of the pain of choosing and using existing languages may be gratuitous, that many of the bothersome details of expression have little relation to the content to be encoded. Put another way, if the power really is in the knowledge, then the knowledge ought to be separate from the bewildering considerations involved in choosing a system architecture. The inventors of current representation techniques usually praise their languages for how well they are suited to expressing expertise, but then turn around and stress how arcane an art is *true* knowledge engineering. I cannot help but think, looking at these languages, that perhaps they have some of the praise and blame misplaced. Instead of being unqualified advantages, current formal languages are mixed blessings.

The dubious virtue of using current knowledge representation languages is just a symptom of one of the more serious disadvantages of using computers at this time for expert system development. The large problem here is that the frameworks currently supplied by artificial intelligence for representation, reasoning, and decision-making are simply inadequate and ill-understood. One result of this is that concrete frameworks like EMYCIN, OPS5, PROLOG, etc. must be worked around rather than worked with. People put up with the onerous chore of making these systems work in spite of themselves simply because little else is available for immediate use. While AI has some better ideas, they have not yet been embodied in systems as practicable as EMYCIN *et alia*. And as long as expert system development is tied to concurrent computer implementation, one must take what implemented systems one can get.

Finally, the most talked-about disadvantage of developing expert systems on computers is also a consequence of this lack of adequate theories of representation, reasoning, and decision-making. The irregularities and peculiarities of current techniques offer almost insuperable barriers to understanding by the uninitiated, thus creating a virtual priesthood of knowledge engineers privy to the inner mysteries. Since the techniques available are ill-understood, merely being taught them helps no more than being taught the words in a foreign language without being taught the grammar, meaning, or culture. It forces the would-be knowledge engineer to endure an apprenticeship every bit as inarticulate as that of the twelfth-century cobbler, and this is the source of scarcity of knowledge engineers. One could, of course, articulate the expertise of master knowledge engineers, but if most of this expertise is concerned with rituals and mindless tricks developed to circumvent the infelicities of current knowledge engineering systems, there might not be much point to it, especially since the details of these systems are in a constant state of flux themselves. Make things simple enough conceptually, and how-to books and community college courses will solve the training problem.

Theory and Trust in Artificial Intelligence

Unfortunately, our lack of adequate theoretical understandings of artificial intelligence techniques and the resulting annoying impediments to expert system construction are not just disadvantages of using current computer-based systems for knowledge engineering. If mere dogwork was the only obstacle, we would raise dogs. But the more serious consequence of unintelligible knowledge engineering tools and systems comes out once we start putting expert systems into use. We have no clear theory of the reasoning and decision-making techniques used in current systems. This means that we cannot easily or reliably predict the system's behavior from knowledge of the information it possesses. Knowledge engineers are currently perceived as indespensible partly because they are often the only people who can understand the systems they have implemented well enough to be able to change them effectively. For example, because of the brittleness and irregularities of the inferential and procedural techniques employed, an expert system may work perfectly on one case yet fail (unexpectedly to everyone but the knowledge engineer) on related cases. In particular, current systems may fail to yield useful tentative conclusions when only partial information is available; they may fail horribly on complete but slightly different cases; and they often cannot solve simpler or more qualitative versions of the same problem. The difficulty is partly one of common sense, and partly one of simplicity. If we told all the expert information to a human, then, aside from bookkeeping errors, we can predict with some accuracy his behavior by putting ourselves in his shoes, by using our reasoning and decision-making powers as a model for his. If computer-based systems used reasoning and decision-making techniques either of sufficient similarity to common sense, or of sufficient simplicity and regularity to be comprehensible even if they diverge from common sense, we could understand the powers and limitations of expert systems as well, extrapolating from rules to conclusions by reflection or simulation. But given the complexity of the behavior of current inference and decision-making techniques, many guesses about behavior are likely to be wrong. This is why artificial intelligence places such a premium on implementing and testing ideas even if they seem intuitively sound, and why knowledge engineers are the only people who understand their systems (if even they do). But if we place trust in an expert system because its information appears sound and reasonable, and because it has succeeded on a few dozen test cases, we are derelict in responsibility and prudence, for the uncommon sense of current systems offers small warrant for success on any other cases. Until we understand them better, either because they match our intuitions more closely, or because we can formally analyze their behavior, we must treat these programs like any other, where testing may only show the presence of bugs, not their absence.

Even without systems that obey comprehensible theories of reasoning and decision-making, we may feel safe in extrapolating success on thousands or hundreds of thousands of cases to acceptable performance in all but the most unlikely events. But when applications include tasks like power plant control, missile detection, personal credit screening, and medical diagnosis, the likelihood of serious errors and uncertainty of information do not need to be increased by brittle procedures for reasoning and decision-making. (In some cases, such as power plant control, the sheer complexity of the task being monitored means that the best human "experts" may themselves be rather incompetent, so that simply encoding their "expertise" in an automatic system may be folly. In these cases, deeper analysis of the system might improve on the best human performance. One can interpret the Steamer project favorably in this way.) Thus developing better theories for computational reasoning and decision-making should help make the world a safer place as well as ease the construction of expert systems. (See also (Doyle, 1983), (McCarthy, 1983), and (Nilsson, 1983).)

Conclusion

Expert systems address real needs. We should build more of them to get the experience and the benefits, but in many ways computers are inessential to getting these benefits, in theory if not in practice. The principal accomplishment to date of the computer-based experts has been one of broadening our imagination of what might be done soon, rather than actually doing substantial tasks. Though in the long run the advantages of using computers should outweigh their disadvantages (those disadvantages not remedied by theoretical progress in artificial intelligence), it would be very interesting to see how far the techniques of articulate apprenticeship can be pushed without the use of computers or certified knowledge engineers. It may be cheaper at present to rough out preliminary versions of knowledge bases using only human apprentices before bringing in specialized machines and scarce knowledge engineers to complete the mechanization. Even if not, there should be many things learned through experimenting with such an approach, things useful in teaching people how to learn and teach more effectively. These, of course, are skills of immense importance to our society, independent of expert systems. Along with progress on articulate apprenticeship, we desperately need progress on the theories of common sense reasoning and decision-making, in order to make machines which can successfully employ the knowledge gained through articulate apprenticeship. And finally, while we are so ignorant about learning and discovery, it might be wise to start paying articulate experts to tell their secrets and then switch to something new, so that they can then tell and switch again.

References

Davis, R., 1982. Expert Systems: Where are we? And where do we go from here?, *AI Magazine* V. 3, No. 2, 3-22.

Doyle, J., 1983. What is Rational Psychology? Toward a modern mental philosophy, *AI Magazine*, V. 4, No. 3, 50-53.

Feigenbaum, E. A., 1977. The art of artificial intelligence: I. Themes and case studies of knowledge engineering, *Fifth International Joint Conference on Artificial Intelligence*, 1014-1029.

Feigenbaum, E. A., and McCorduck, P., 1983. *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*, Reading: Addison-Wesley.

McCarthy, J., 1983. President's Quarterly Message, *AI Magazine* V. 4, No. 4, 5.

Nilsson, N. J., 1983. Artificial intelligence prepares for 2001, *AI Magazine* V. 4, No. 4, 7-14.

Waterman, D., Hayes-Roth, R., and Lenat, D., 1983. *Building Expert Systems*, Reading: Addison-Wesley.