Chapter 13

Knowledge elicitation

Nigel Shadbolt and Mike Burton

Introduction

Expert systems are computer programs which are intended to solve real-world problems, achieving the same level of accuracy as human experts. There are many obstacles in such an endeavour. One of the greatest is the acquisition of the knowledge which human experts use in their problem solving. The issue is so important to the development of knowledge-based systems that it has been described as the 'bottle-neck in Expert Systems construction' (Hayes-Roth *et al.*, 1983).

Despite its central role there is no comprehensive theory of knowledge acquisition available. Many regard the area as an art rather than a science. It is not the purpose of this chapter to investigate the theoretical shortcomings of knowledge acquisition but to deliver practical advice and guidance on performing the process.

Expert systems

In the early days of Artificial Intelligence much effort went into attempts to discover general principles of intelligent behaviour. Newell and Simon's (1963) General Problem Solver exemplifies this approach. They were interested in uncovering a general problem solving strategy which could be used for any human task. In the early 1970s this position came to be challenged. A new slogan came to prominence—'in the knowledge lies the power'. A leading exponent of this view was Edward Feigenbaum of SRI. He observed that experts are experts by virtue of domain specific problem solving strategies together with a great deal of domain specific knowledge. It was the attempt to incorporate these various sorts of domain knowledge which resulted in the class of programs called Expert Systems.

Throughout this chapter we will be assuming that current commercially available expert system software will be the implementation vehicle for the programs. Thus the form in which the knowledge will be implemented is likely to be standard *production rules* with perhaps a *structured object* facility such as *frames*. For a review of the major types of expert system architecture see Jackson (1985) and of different knowledge representation formalisms see Shadbolt (1989).

The problem of acquisition

The people who build expert systems, the knowledge engineers, are typically not people with a deep knowledge of the application domain. However, it is the knowledge engineers who must gather the domain knowledge and then implement it in a form that the machine can use. In the simplest case, the knowledge engineer may be able to gather information from a variety of non-human resources: e.g. text books, technical manuals. However, in most cases one needs actually to consult a practising expert. This may be because there isn't the documentation available, or because real expertise in the problem solving derives from practical experience in the domain, rather than from a reading of standard texts. The task of gathering information generally, from whatever source, is called *knowledge acquisition*. The subtask of gathering information from the expert is called *knowledge elicitation* (KE).

Many problems arise before an elicitation of the detailed domain knowledge is ever conducted. There are possible failures in the understanding of what it is realistic to build. Sometimes the failure occurs when formulating the role of the system. On other occasions there is an inadequate understanding of the task environment. Very often the effort and resources required to build systems are underestimated: this occurs in both the development and maintenance of systems. A particularly nasty situation arises when the knowledge engineer is expected to conjure up knowledge for areas in which no evidence of systematic practice exists at all. Knowledge engineers seem to be expected to provide theories for domains where there is no theory. Providing we can avoid all of these obstacles then we get down to detailed issues of KE.

The problem of elicitation

The question in KE is this: how do we get experts to tell us exactly what they do? The task is enormous, particularly in the context of large expert systems. There are a variety of circumstances which contrive to make the problem even harder. Much of the power of human expertise lies in laid-down experience, gathered over a number of years, and represented as heuristics. Often the expertise has become so routinized that experts no longer know what it is that they do or why.

There are also commercial reasons to try to make KE more effective. We would like to be able to use techniques which will minimize the effort spent in gathering, transcribing and analysing the knowledge. We would like to minimize the time spent with expensive and scarce experts. And, of course, we would like to maximize the yield of usable knowledge.

This chapter will continue by describing, in sufficient detail for the reader to apply them, examples of major KE methods. We will then mention other techniques and where the reader can find out more about them. In later sections we will review aspects of expertise and cognition that are likely to directly affect the KE process. Finally, we describe the construction of programmes of acquisition.

Methods of knowledge elicitation

The structured interview

Almost everyone starts in KE by determining to use an interview. The interview is the most commonly used knowledge elicitation technique and takes many forms, from the completely *unstructured* interview to the formally-planned, *structured* interview. (For a full review of interview techniques see Sinclair in this volume.) The structured interview is a formal version in which the knowledge engineer has planned the whole session. The structured interview has the advantage that it provides structured transcripts that are easier to analyse than unstructured 'chat'. The relatively formal interview which we have specified here constrains the expert–elicitor dialogue to the general principles of the domain. Experts do not work through a particular scenario extracted from the domain by the elicitor; rather the experts generate their own scenarios as the interview progresses. The structure of a typical interview is as follows.

1. Ask the expert to give a brief (10 min) outline of the target task, including the following information:
   (a) an outline of the task, including a description of the possible solutions to the problem;
   (b) a description of the variables which affect the choice of solutions;
   (c) a list of major rules which connect the variables to the solutions.
2. Take each rule elicited in stage 1, ask when it is appropriate and when it is not. The aim is to reveal the scope (generality and specificity) of each existing rule, and hopefully generate some new rules.
3. Repeat stage 2 until it is clear that the expert will not produce any additional information.

It is important in using this technique to be clear and specific about how to perform stage 2. We have found that it is helpful to constrain the elicitor's interventions to a specific set of *probes*, each with a specific function. Here is a list of probes (P) and related functions (F) which will help in stage 2.

P1	Why would you do that?
F1	Converts an assertion into a rule.
P2	How would you do that?
F2	Generates *lower order* rules.

P3 When would you do that? Is <the rule> always the case?
F3 Reveals the generality of the rule and may generate other rules.
P4 What alternatives to <the prescribed action/decision> are there?
F4 Generates more rules.
P5 What if it were not the case that <currently true condition>.
F5 Generates rules for when current condition does not apply.
P6 Can you tell me more about <any subject already mentioned>.
F6 Used to generate further dialogue if expert dries up.

The idea here is that the elicitor engages in a type of slot/filler dialogue. The requirement that the elicitor listens out for relevant concepts and relations imposes a large cognitive load on the elicitor. The provision of fixed linguistic forms within which to ask questions about concepts, relations, attributes and values makes the elicitor's job very much easier. It also provides sharply focused transcripts which facilitate the process of extracting usable knowledge. Of course, there will be instances when none of the above probes are appropriate (such as the case when the elicitor wants the expert to clarify something). However, you should try to keep the interjections necessary in such situations to a minimum. The point of specifying such a fixed set of linguistic probes is to constrain the expert to giving you all, and only, the information you want.

The sample of dialogue below is taken from a real interview of this kind. It is the transcript of an interview by a knowledge engineer (KE) with an expert (EX) on VDU fault diagnosis*.

EX I actually checked the port of the computer.
KE Why did you check the port?
EX If it's been lightning recently then it's a good idea to check the port + because lightning tends to damage the ports.
KE Are there any alternatives to that problem?
EX Yes; that ought to be prefaced by saying do that if it was several keys with odd effects + not necessarily all of them, but more than 2.
KE Why does it have to be more than 2?
EX Well if it was only one or two keys doing funny things then the thing to do would be to check the keys themselves + check the contacts of the keys + check that they're closing properly + speed would affect all keys, parity would affect about half the keys.

This is quite a rich piece of dialogue. From this section of the interview alone we can extract the following rules:

```
IF there has been recent lightening
THEN check port for damage
IF there are two or fewer malfunctioning keys
THEN check the key contacts
IF about half the keyboard is malfunctioning
THEN check the parity
```

*In the transcripts we use the symbol + to represent a pause in the dialogue

IF the whole keyboard is malfunctioning
THEN check the speed

Of course these rules may need refining in later elicitation sessions, but the text of the dialogue shows how the use of the specific probes has revealed a well-structured response from the expert†.

In all the interview techniques (and in some of the other generic techniques as well) there exist a number of dangers that have become familiar to knowledge engineers. One problem is that experts will only produce what they can verbalize. If there are non-verbalizable aspects to the domain, the interview will not recover them. This can arise from two causes. It may be that the knowledge was never explicitly represented or articulated in terms of language (consider, for example, pattern recognition expertise). Then there is the situation where the knowledge was originally learnt explicitly in a propositional form but the experts may have *compiled* the knowledge to such an extent that they regard the complex decisions they make as based on hunches or intuitions; in fact, these decisions are based upon large amounts of remembered data and experience, and the continual application of strategies. In this situation they tend to give ***black box*** replies "I don't know how I do that . . . .", "It is obviously the right thing to do . . .".

Another problem arises from the observation that people (and experts in particular) often seek to justify their decisions in any way they can. It is a common experience of the knowledge engineer to get a perfectly valid decision from an expert, and then to be given a spurious justification. For these and other reasons we have to supplement interviews with additional methods of elicitation. Elicitation should always consist of a programme of techniques and methods. This brings us on to consider another technique much favoured by knowledge engineers.

Protocol analysis

Protocol Analysis (PA) (considered in detail by Bainbridge in this book) is a generic term for a number of different ways of performing some form of analysis of the expert(s) actually solving problems in the domain. In all cases the engineer takes a record of what the expert does—preferably by video or audio tape—or at least by written notes. Protocols are then made from these records and the knowledge engineer tries to extract meaningful rules from the protocols.

We can distinguish two general types of PA—*on-line* and *off-line*. In on-line PA the expert is being recorded solving a problem, and concurrently a commentary is made. The nature of this commentary specifies the two subtypes of the *on-line* method. The expert performing the task may be describing what they are doing as problem solving proceeds. This is called

†In fact, a possible ***second-phase*** elicitation technique would be to present these rules back to the expert and ask about their truthfulness, scope and so forth.

self-report. A variant on this is to have another expert provide a running commentary on what the expert performing the task is doing. This is called ***shadowing***.

Off-line PA allows the expert(s) to comment retrospectively on the problem solving session—usually by being shown an audio-visual record of it. This may take the form of retrospective self-report by the expert who actually solved the problem, it could be a critical retrospective report by other experts, or there could be group discussion of the protocol by a number of experts including its originator. In the case in which only a behavioural protocol is obtained then obviously some form of retrospective verbalization of the problem solving episode is required.

Before PA sessions can be held, a number of pre-conditions should be satisfied. The first of these is that the knowledge engineer is sufficiently acquainted with the domain to understand the expert's tasks. Without this the elicitor may completely fail to record or take note of important parts of the expert's behaviour. A second requirement is the careful selection of problems for PA. The sampling of problems is crucial. PA sessions may take a relatively long time and only a few problems consequently can be addressed. Therefore, the selection of problems should be guided by how representative they are. Asking experts to sort problems into some form of order (Chi *et al*., 1981, 1982) may give an insight into the classification of types of problems and help in the selection of suitable problems for PA (see also the next two sections on concept sorts and laddering).

A further condition for effective PA is that the expert(s) should not feel embarrassed about describing their expertise in detail. It is preferable for them to have experience in thinking aloud. Uninhibited thinking aloud has to be learned in the same way as does talking to an audience. One or two short training sessions may be useful, in which a simple task is used as an example. This puts the experts at ease and familiarizes them with the task of talking about their problem solving. In trying to decide when it is appropriate to use PA bear in mind that it is alleged that different KE techniques differentially elicit certain kinds of information. With PA it is claimed that the sorts of knowledge elicited include: the 'when' and 'how' of using specific knowledge, problem solving and reasoning strategies; evaluation procedures and evaluation criteria used by the expert; and procedural knowledge about how tasks and subtasks are decomposed.

When actually conducting a PA the following are a useful set of tips to help enhance its effectiveness:

1. Present problems and data in a realistic way; the way problems and data are presented should be as close as possible to a real situation
2. Transcribe the protocols as soon as possible; the meaning of many expressions is soon lost, particularly if the protocols are not recorded.
3. In almost all cases an audio recording is sufficient, but videorecordings have the advantage of containing additional and disambiguating information.

4. Avoid long self-report sessions; thinking aloud is significantly more tiring for the expert, than is being interviewed, because of the need to perform a double task This is one reason why shadowing is sometimes preferred.

5. In general, the presence of the knowledge engineer is required in a PA session. Although adopting a background role, the knowledge engineer's very presence suggests a listener to the interviewee, and lends meaning to the talking aloud process. Therefore, comments on audibility, or even silence by the knowledge engineer, are quite acceptable.

Protocol analyses share with the unstructured interview the problem that they may deliver unstructured transcripts which are hard to analyse. Moreover, they focus on particular problem cases and so the scope of the knowledge produced may be very restricted. It is difficult to derive general domain principles from a limited number of protocols. These are practical disadvantages of protocol analysis, but there are more subtle problems. Two actions, which look exactly the same to the knowledge engineer, may be the result of two quite different sets of considerations. This is a problem of impoverished interpretation by the knowledge engineer. They simply do not know enough to discriminate the actions. The obverse to this problem can arise in shadowing and the retrospective analyses of protocols by experts. Here the expert(s) may simply wrongly attribute a set of considerations to an action after the event This is analogous to the problems of misattribution in interviewing.

A particular problem with self-report, apart from its being tiring, is the possibility that verbalization may interfere with performance. The classic demonstration of this is for a driver to attend to all the actions involved in driving a car. If one consciously monitors such variables as engine revs, current gear, speed, visibility, steering wheel position and so forth, the driving invariably gets worse. Such skill is shown to its best effect when performed automatically. This is also the case with certain types of expertise. By asking the expert to verbalize. one is in some sense destroying the point of doing protocol analysis—to access procedural, real-world knowledge.

Having pointed to these disadvantages, it is also worth remembering that context is sometimes important for memory—and hence for problem-solving. For most non-verbalizable knowledge, and even for some verbalizable knowledge, it may be essential to observe the expert performing the task, for it may be that this is the only situation in which the expert is actually able to 'communicate' knowledge.

Finally, when performing PA it is useful to have a set of conventions for the actual interpretation and analysis of the resultant data. Kuipers and Kassirer (1983) and Belkin *et al*. (1987) provide detailed guidelines for such analysis.

The two classes of generic technique discussed so far are *natural* and intuitively easy to understand. Experts are used to expressing their knowledge in these sorts of ways. The techniques that follow are what we might term *contrived*, and permit the expression of knowledge in ways that are likely to be unfamiliar to the expert.

Concept sorting

Concept sorting is a technique that is useful when we wish to uncover the different ways an expert sees relationships between a fixed set of concepts. In the version we will present an expert is presented with a number of cards on each of which a concept word is printed. The cards are shuffled and the expert is asked to sort the cards into either a fixed number of piles or else to sort them into any number of piles the expert finds appropriate. This process is repeated many times. One attempts to get multiple views of the structural organization of knowledge by asking the expert to do the same task over and over again, each time creating at least one pile that differs in some way from previous sorts. The expert should also provide a name or category label for each pile on each different sort.

Performing a card sort requires the elicitor to be not entirely naive about the domain. Cards have to be made with the appropriate labels before the session. However, no great familiarity is required as the expert provides all the substantial knowledge in the process of the sort. We now provide an example from our VDU domain to show the detailed mechanics of a sort.

The concepts printed on the cards were faults and corrective actions drawn from a structured interview with the expert. He had outlined 20 faults and outcomes:

(A) damaged VDU port	(B) faulty key contacts
(C) incorrect parity setting	(D) program is busy
(E) damaged tube	(F) terminal not switched on
(G) mains fuse blown	(H) fuse in VDU blown
(I) loose connection or break in line	(J) press control-c
(K) press control-q	(L) press control-z
(M) VDU power supply fault	(N) software fault
(O) video section fault on VDU	(P) incorrect terminal speed setting
(Q) terminal requires reset	(R) VDU transmission fault
(S) transmit line fault	(T) receive line fault

The expert was shown possible ways of sorting cards in a *toy* domain as part of the briefing session, and then asked to sort the real elements in the same way.

The dimensions/piles (P) which the expert used for the various sorts (S) were as follows:

S1	P1 hardware fault	P2 software fault
S2	P1 easy to clear	P2 difficult to clear

S3	P1	no component damage	P2	component damage usually as symptom of another outcome
	P3	component damage as an outcome		
S4	P1	communication fault	P2	not a communication fault
S5	P1	no output	P2	no response
	P3	garbled output	P4	combination of no response and garbled output

The following table shows the pile number of each sort for each fault or corrective action (card label); almost all the fault concepts are distinguishable from one another—even with these few sorts.

	S1	S2	S3	S4	S5
A	1	2	3	1	4
B	1	1	3	2	3
C	2	1	1	1	3
D	2	1	1	2	1
E	1	2	3	2	1
F	2	1	1	2	1
G	1	1	2	2	1
H	1	1	2	2	1
I	1	2	3	1	4
J	2	1	1	2	2
K	2	1	1	2	2
L	2	1	1	2	2
M	1	2	3	2	1
N	1	2	1	1	2
O	1	2	3	2	1
P	2	1	1	1	3
Q	2	1	1	1	4
R	1	2	1	1	4
S	2	2	3	2	4
T	1	2	1	1	4

Using this information we can attempt to extract decision rules directly. An example of a rule extracted from the sortings is:

IF	it is a hardware fault (sort 1/pile 1)
AND	it is easy to clear (sort 2/pile 1)
AND	it is component damage (sort 3/pile 3)
AND	it is NOT a communication fault (sort 4/pile 2)
AND	there is garbled output (sort 5/pile 3)
THEN	there are faulty key contacts (outcome B)

As can be seen from the example such sorts produce long and cumbersome rules. In fact many of the clauses may be redundant—once you have established that the fault is a hardware one, there is no need to check whether it is component damage. However, the utility of this technique does not reside solely in the production of decision rules. We can use it, as we have said, to explore the general interrelationships between concepts in the domain. We are trying to make explicit the implicit structure that experts impose on their expertise. Variants of the simple sort are different forms of *hierarchical* sort. One such version is to ask the expert to proceed by producing first two piles, on the second sort three, then four and so on. Finally we ask if any two piles have anything in common. If so you have isolated a higher order concept that can be used as a basis for future elicitation.

The advantages of concept sorting can be characterized as follows. It is fast to apply and easy to analyse. It forces into an explicit format the constructs which underlie an expert's understanding. In fact it is often instructive to the expert; a sort can lead the expert to see structure in his view of the domain which he himself has not consciously articulated before. Finally, in domains where the concepts are perceptual in nature (i.e. x-rays, layouts and pictures of various kinds) then the cards can be used as a means of presenting these images in an attempt to elicit names for the categories and relationships that might link them. There are, of course, features to be wary of with this sort of technique. Experts can often confound dimensions by not consistently applying the same semantic distinctions throughout an elicitation session. Alternatively, they may over-simplify the categorization of elements, missing out important caveats. One thing that we have found (Schweikert *et al.*, 1987) is that an expert's own opinion of the worth of a technique is no guide to its real value. In methods such as sorting we have a situation in which the expert is trying to demonstrate expertise in a non-natural or contrived manner. He or she might be quite used to chatting about their field of expertise, but sorting is different and experts are suspicious of it. Experts may in fact feel they are performing badly with such methods. However, on analysis one finds that the yield of knowledge is as good and sometimes better than for non-contrived techniques.

Laddered grids

Once again this is a fairly contrived technique, and it will be necessary to explain it fully to the expert before starting. The expert and the knowledge engineer construct a graphical representation of the domain in terms of the relations between domain elements. The result is a qualitative, two-dimensional graph where nodes are connected by labelled arcs. No extra elicitation method is used here, but expert and elicitor construct the graph together by negotiation.

In using the technique the elicitor enters the conceptual map at some point and then attempts to move around it with the expert. A formal specification of how we use the technique is shown below together with an example of its use.

1. Start the expert off with a seed item.
2. Move around the domain map using the following prompts:
   1. To move DOWN the expert's domain knowledge:
      How can you tell it is <ITEM>?
      Can you give examples of <ITEM>?
   2. To move ACROSS the expert's domain knowledge:
      What alternative examples of <CLASS> are there to <ITEM>?
   3. To move UP the expert's domain knowledge:
      What have <SAME LEVEL ITEMS> got in common?
      What are <SAME LEVEL ITEMS> examples of?
      What is the key difference between <ITEM 1> and <ITEM 2>?

The elicitor may move around the domain map of knowledge in any order which seems convenient. As the elicitation session progresses, the elicitor keeps track of the elicited knowledge by drawing up a network on a large piece of paper. This representation allows the elicitor to make decisions (or ask questions) about what constitutes higher or lower order elements in the domain. In order to give the reader the flavour of the technique, there follows an extract from a laddered grid elicitation session. Once again, the knowledge domain is VDU fault diagnosis.

KE What examples of a 'no response' problem are there?
EX It's a software problem, unless a small set of keys is affected.
KE Can you give me an example of what you do if there's a small set of keys affected?
EX Yes. You check the key contacts.
KE Can you give me examples of actions for software problems?
EX Yes You intervene or don't intervene. You don't intervene if there's a user error where they misinterpret a slow program as this fault.
KE Can you give me an example of when you intervene?
EX When there's an editor problem. Then you use control keys.
KE Can you give me an example of using control keys?
EX Press control-c, control-q and control-z.
KE What is the difference between control-c and control-q?
EX Control-q gets you out of control-s and control-c gets you out of the program. Control-q should be used before control-c.
KE What is the difference between control-c and control-z?
EX Control-c gets you out of the program and should be used before control-z. Control-z quits the system.

From this portion of a laddered grid interview the elicitor drew up a hierarchical representation of the domain as shown in Figure 13.1.

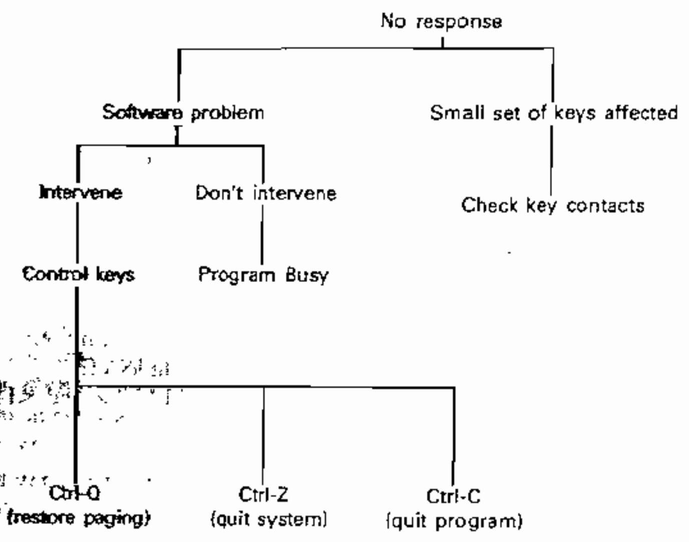

Figure 13.1. Laddering in the VDU Domain.

This hierarchy gives rise to the following set of rules which could be included in the knowledge base of an expert system for VDU fault finding.

```
IF   there is no response
AND  it is a software problem
AND  intervention is needed
THEN press control-q

IF   control-q doesn't work
THEN press control-c

IF   control-c doesn't work
THEN press control-z
```

We have found that this form of knowledge elicitation is very powerful for structured domains. As with other contrived techniques we have found that whilst an expert may think this technique is revealing little of interest, subsequent analysis provides good quality rules.

The limited information task

A technique which does not provide a spatial representation of the domain, but rather a set of hints or suggestions which may prove useful in expert system construction is a technique called the *limited information task* (Hoffman,

1987) or *20 questions* (Grover, 1983). The expert is provided with little or no information about a particular problem to be solved, but must then ask the elicitor for specific information which will be required to solve the problem. The information which is requested, along with the order in which it is requested, provides the knowledge engineer with an insight into the expert's problem solving strategy. One difficulty with this method is that the knowledge engineer needs a good understanding of the domain in order to make sense of the experts' questions, and to provide meaningful responses. The elicitor should have forearmed themselves with a problem from the domain together with a *crib sheet* of appropriate responses to the questions.

In a version of the limited information task which we use, we tell the expert that the elicitor has a scenario in mind and the expert must determine what it is. The scenario might represent a problem, a solution or a problem context. The expert is told that they may ask the elicitor for more information, though what the elicitor gives back is terse and does not go much beyond what was asked for in the question. The expert may be asked to explain why each of the questions was asked.

An example of the kind of interaction produced by this technique is shown below. Here the problem domain is in the construction of lighting systems for the inspection of industrial products and processes.

EX Is this in the manufacturing industry?
KE Yes.
EX So we've ruled out things like fruit, vegetables, cows?
KE Yes.
EX Is it the metal industry?
KE The material is wood.
EX So we could be dealing with a large object here like a chair or table.
KE The object is large.
EX It's likely to be a 3-D object, you've got to pick it up and turn it over.
KE That's right.
EX So what I need now are the dimensions of this object in terms of the cube that will enclose it.
KE It would have similar dimensions to the table top.
EX Do I inspect one surface or all the surfaces?
KE All of them.
EX Is the inspector looking for one or many faults?
KE One particular fault.
EX Can you describe it for me?
KE It's pencil marks about half an inch long.
EX What colour is the wood?
KE Dark unfinished wood.
EX We've got a contrast problem here. At this point I'd go and look at the job + to see if the graphite pencil marks reflect light + sometimes

it does, but it depends on the wood + if it does you can select the light to increase the contrast between the fault and the background.

EX: I'd be doing this in three phases: first a general lighting, then specific for surface lighting, and then some directional light [expert then gives technical specifications for these types of light].

This interview gives us an interesting insight into the natural line of enquiry of an expert in this domain. Often expert systems gather the right data but the order in which they are gathered and used can be remote from how an expert works. This can decrease the acceptability of the system if other experts are to use it, and it also has consequences for the intelligibility of any explanations the system offers in terms of a retrace of its steps to a solution.

It will be seen that we can once again extract decision rules directly from the dialogue: e.g.

```
IF   fault colour is black
AND  object colour is dark
THEN contrast is a problem
```

The drawbacks to this technique are that the elicitor needs to have constructed plausible scenarios and to be able to cope with questions asked. The experts themselves are often uncomfortable with this technique; this may well have to do with the fact that, as with other contrived techniques, it is not a natural means of manifesting expertise. Whilst a few scenarios may reveal some of the general rules in a domain the elicitation is very case specific. In order to get the range of knowledge for a sweep of situations many scenarios would need to be constructed and used.

Automatic elicitation

As KE is acknowledged to be a time consuming and difficult process, the idea of automated elicitation is particularly attractive. A number of programs have been developed towards this goal, and we will briefly consider some of them in this section.

There are two main types of automated elicitation: (1) those systems which are implementations of standard KE techniques; and (2) those systems which use machine learning techniques to induce rules from sets of worked examples and observed data. In addition to these categories, there are also systems which use knowledge about the structure of a particular domain in order to drive the elicitation. However, these are large-scale systems dedicated to specific projects, and are not generally available. Readers interested in this category of system are referred to Marcus *et al.* (1985) for a review.

Of those systems which implement standard techniques, the most successful are based on the *repertory grid*. This technique has its roots in the psychology of personality (Kelly, 1955) and is designed to reveal a conceptual map of a domain, in a similar fashion to the card sort as discussed above (see Shaw and Gaines, 1987a, for a full discussion; also, see Sinclair in this book). Briefly, subjects are presented with a range of domain elements and asked to choose three, such that two are similar, and different from the third. So, for example, people might be presented with a set of animals, and choose elephant and hippo as the two similar elements, and sparrow as the third. The subject is then asked for their reason for differentiating these elements, and this dimension is known as a construct. In our example 'size' would be a suitable construct. The remaining domain elements are then rated on this construct.

This process continues with different triads of elements until the expert can think of no further discriminating constructs. The result is a matrix of similarity ratings, relating elements and constructs. This is analysed using a statistical technique called *cluster analysis*. In KE, as in clinical psychology, the technique can reveal clusters of concepts and elements which the expert might not have articulated in an interview. However, the disadvantage of the technique is that it is very time-consuming to administer by hand, and the cluster analysis is complex to perform and interpret. This naturally suggests that an implemented version would be appropriate.

There are several repertory grid programs on the market. However, the best known programs are ETS (Boose, 1985), its successor, AQUINAS (Boose and Bradshaw, 1987), and KITTEN (Shaw and Gaines, 1987b). Although these are largely research tools rather than commercial products, they provide a good focus for discussion of KE using an automated repertory grid. With each of these systems the expert interacts directly with a computer. The programs are run in such a way that the repertory grid is built-up interactively, and the expert is shown the resultant knowledge. Experts have the opportunity to refine this knowledge during the elicitation process. The output of these systems is a set of machine-executable rules. In AQUINAS, these rules can take the form necessary for a wide range of expert system shells. These systems bypass the human elicitor altogether and have been used with considerable success in small-scale domains where solutions can be comfortably enumerated (see Boose, 1986 for a list of successful applications of ETS). However, the rep grid is not well-suited to more complex domains involving construction or planning.

We now turn to automated elicitation of the second kind. Machine induction is the process whereby a program is presented with many solved problems (or other appropriate large data sets) from the domain, and uses statistical regularities to infer underlying rules. An introductory overview to this technique can be found in Hart (1986). The best known algorithm for this is ID3 (Quinlan, 1979, 1983), and versions are now available in some small-scale shells. To conceptualize the process, imagine presenting sets of motor car symptoms alongside decisions about the fault diagnosis. It may be that a mechanic cannot readily articulate a rule relating a set of symptoms to a particular fault, but if they occur reliably together with the fault, the program induces a rule relating them. So, the next time this configuration of faults is encountered, the appropriate diagnosis is made.

Although this technique sounds attractive, it has several associated problems. Rules may be induced which do not have any basis in the domain. This can arise according to quite arbitrary decisions such as the order in which the learning examples are input. Furthermore, there is the problem of cause and effect—the fact that a particular set of faults co-occurs with a diagnosis on the learning set does not necessarily imply a causal relation. Research continues on more sophisticated machine learning techniques for KE (e.g. Michalski *et al.*, 1986; Mitchell, 1982). Clearly though, induction programs do not provide a complete solution to automatic elicitation.

Finally, in this section, we should mention that there are now several large-scale knowledge acquisition environments under construction. These typically provide a number of automated KE techniques, knowedge base editors, automated transcript analysis and various other support software for the knowledge engineer. These systems are currently at the research stage, and as yet are not generally available. However, the interested reader is referred to Anjewierden (1987), Motta *et al.* (1987) and Diederich *et al.* (1987) for accounts of KADS Power Tools, KEATS and KRITON, respectively. These systems indicate the shape and form of the next generation of knowledge engineering tools.

A taxonomy of KE techniques

We have attempted to sample some of the major approaches to elicitation and where appropriate give a detailed description of techniques that are likely to be of use. There are many variants on the methods we have described. Below we have provided a taxonomy of methods with which we are familiar together with a primary reference for each one.

- Non-contrived
  - Interviews
    - Structured
      - Focused interviews (Hart, 1986)
      - Forward scenario simulation (Grover, 1983)
    - Teach back (Johnson and Johnson, 1987)
  - Unstructured (Weis and Kulikowski, 1984)
- Protocol analysis
  - Verbal
    - On-line (Johnson *et al.*, 1987)
    - Off-line (Elstein *et al.*, 1978)
    - Shadowing (Clarke, 1987)

Behavioural (Ericsson and Simon, 1984)
Contrived
Conceptual mapping
Sorting and rating (Gammack, 1987a,b)
Repertory grid (Shaw and Gaines, 1987a)
Pathfinder (Schvaneveldt *et al.*, 1985)
Goal decomposition
Laddered grid (Hinkle, 1965)
Limited-information task (Grover, 1983; Hoffman, 1987)
Automatic
Rule induction (Shapiro, 1987)

Having discussed the principle methods of elicitation we should spend a little time reflecting on the nature of two other major components of the KE enterprise, namely the experts and the expertise they possess.

On experts

Experts of course come in all shapes and sizes. Ignoring the nature of your expert is another potential pitfall in KE. A coarse guide to a typology of experts might make the issues clearer. Let us take three categories we shall refer to as *academics, practitioners* and *samurai* (in practice experts may embody elements of all three types). Each of these types of expert differ along a number of dimensions. These include: the outcome of their expert deliberations, the problem solving environment they work in, the state of the knowledge they possess (both its internal structure and its external manifestation), their status and responsibilities, their source of information, and the nature of their training.

How are we to distinguish these different types of expert? The academic type regards their domain as having a logically organized structure. Generalizations over the laws and behaviour of the domain are important to them; theoretical understanding is prized. Part of the function of such experts may be to explicate, clarify and teach others, thus they talk a lot about their domains. They may feel an obligation to present a consistent story both for pedagogic and professional reasons. Their knowledge is likely to be well structured and accessible. These experts may suppose that the outcome of their deliberations should be the correct solution of a problem. They believe that the problem can be solved by the appropriate application of theory. They may, however, be remote from everyday problem-solving.

The practitioner class on the other hand are engaged in constant day to day problem-solving in the domain. For them specific problems and events are the reality. Their practice may often be implicit and what they desire as an outcome is a decision that works within the constraints and resource limitations in which they are working. It may be that the generalized theory of the academic is poorly articulated in the practitioner. For the practitioner heuristics may dominate and theory be thin on the ground.

The samurai is a pure performance expert—the only reality is the performance of action to secure an optimal performance. Practice is often the only training and responses are often automatic.

One can see this sort of division in any complex domain. Consider for example medical domains where we have professors of the subject, busy housemen working the wards, and medical ancillary staff performing many important but repetitive clinical activities.

The knowledge engineer must be alert to these differences because the various types of expert will perform very differently in KE situations. The academic will be concerned to demonstrate mastery of the theory. They will devote much effort to characterizing the scope and limitations of the domain theory. Practitioners, on the other hand, are driven by the cases they are solving from day to day. They have often *compiled* or *routinized* any declarative descriptions of the theory that supposedly underlies their problem-solving. The performance samurai will more often than not turn any KE interaction into a concrete performance of the task—simply exhibiting their skill. But there is more to say about the nature of experts and this is rooted in general principles of human information processing. Psychology has demonstrated the limitations, biases and prejudices that pervade all human decision making—expert or novice. To illustrate this, consider the following facts, all potentially crucial to the enterprise of KE.

It has been shown repeatedly that the context in which one encodes information is the best one for recall. It is possible then, that experts may not have access to the same information when in a KE interview, as they do when actually performing the task. So there are good psychological reasons to use techniques which involve observing the expert actually solving problems in the normal setting. In short, protocol analysis techniques may be necessary, but will not be sufficient for effective knowledge elicitation.

Consider now the issue of biases in human cognition. One well-known problem is that humans are poor at manipulating uncertain or probabilistic evidence. This may be important in KE for those domains which require a representation of uncertainty. Consider the rule:

IF the engine will not turn over
AND the lights do not come on
THEN the battery is flat with probability *X*

This seems like a reasonable rule, but what is the value of *X*, should it be 0·9, 0·95, 0·79? The value which is finally decided upon will have important consequences for the working of the system, but it is very difficult to decide upon it in the first place. Medical diagnosis is a domain full of such probabilistic rules, but even expert physicians cannot accurately assess the probability values. In fact there are a number of documented biases in human cognition which lie at the heart of this problem (see for example Kahneman *et al.*, 1982). People are known to undervalue prior probabilities, to use the ends and middle of the probability scale rather than the full range, and to *anchor* their responses around an initial guess. Cleaves (1987) lists a number of cognitive biases likely to be found in knowledge elicitation, and makes suggestions about how to avoid them. However, many knowledge engineers prefer to avoid the use of uncertainty wherever possible.

Cognitive bias is not limited to the manipulation of probability. A series of experiments has shown that systematic patterns of error occur across a number of apparently simple logical operations. For example, *Modus Tollens* states that if 'A implies B' is true, and 'not B' is true, then 'not A' must be true. However people, whether expert in a domain or not, make errors on this rule. This is in part due to an inability to reason with contrapositive statements. Also in part it depends on what A and B actually represent. In other words, they are affected by the content. This means that one cannot rely on the veracity of experts' (or indeed anyone's) reasoning.

All this evidence suggests that human reasoning, memory and knowledge representation is rather more subtle than might initially be thought.

On expertise

Clearly the expertise embodied by experts is not of a homogeneous type. In constructing expert systems it is likely that very different types of knowledge will be uncovered which will have very different roles in the system. There are a number of analyses available of the *epistemology* of expertise. Our analysis is based to a large extent on that of Breuker (1987).

First, we can distinguish what is called *domain level* knowledge. This term is being used in the narrow sense of knowledge that *describes* the concepts and elements in the domain and relations between them. This sort of knowledge is sometimes called *declarative*, it describes what is known *about* things in the domain. The propositions below can be seen as domain level knowledge in this sense:

damaged VDU is a hardware fault
transmit line fault is a software fault

There is also knowledge and expertise which has to do with what we might call the *inference level*. This is knowledge about how the components of expertise are to be organized and used in the overall system. This is quite a high level description of expert behaviour and may often be implicit in expert practice. The following is a description of knowledge about part of a systematic diagnosis system at the inference level.

> To perform systematic diagnosis we will have knowledge about a complaint, and knowledge about observables from the patient or object. We select some aspect of the complaint and using a model of how the system should be performing normally we look to see if a particular parameter of the system is within normal bounds.

Another type of expert knowledge is the *task level*. This is sometimes called *procedural* knowledge. This is knowledge to do with how goals and sub-goals, tasks and subtasks should be performed. Thus in a classification task there may exist a number of tasks to perform in a particular order so as to utilize the domain level knowledge appropriately. This type of knowledge is present in the following extract.

First of all perform a general inspection of the object. Next examine the sample with a hand lens. Next use a prepared thin-section and examine that under a cross-polarizing microscope.

Finally, there is a level of expert knowledge referred to as *strategic knowledge*. This is information that monitors and controls the overall problem solving. This can have to do with the way resources are used; what to do if the proposed solution fails or is found to be inappropriate in some way; what to do when faced with incomplete or insufficient data. Such information is contained in the following extract from an interview.

If I had time I would always check the video driver board. If its a Zenith machine I'd always check that because they are notorious for going wrong . . .

Any field of expertise is likely to contain these various sorts of knowledge to greater or lesser extents. At any particular knowledge level the information may be explicit or implicit in an expert's behaviour. Thus in some domains the experts may have no real notion of the strategic knowledge they are following whilst in others this knowledge is very much in the forefront of their deliberations. Also, of course, the requirements on a system about how far it needs to implement these various levels will vary. It is almost universally acknowledged that significant reasoning about problem domains requires more than just modelling simple relationships between concepts in the domains—it requires causal models of how objects influence and affect one another, models of the processes in which objects participate. This is a hard problem, and often the limitations of first generation expert systems mean that sophisticated domain models cannot be supported.

This brings us to a final important feature of KE. Often the process of acquisition yields much more knowledge than can be implemented (Young, 1987). In this case the knowledge engineer ought to think about laying down the knowledge in a format that will allow it to be used when more powerful implementation systems arrive. Putting knowledge into *deep freeze* in this way requires using an expressive and unambiguous intermediate representation of the knowledge to be stored. A number of candidates exist for this; Young and Gammack (1987) provide a brief review of the alternatives.

Methodologies and programmes of KE

Are there any guidelines as to how techniques should be assembled to form a programme of acquisition? In other words when should we use the various techniques? The choice may depend on the characteristics of the domain, of the expert, and of the required system. Furthermore, it is clear that some techniques are going to be more costly in terms of time with the expert, or of the effort required for subsequent analysis of transcripts.

There are a number of articles and books available on 'how to do knowledge elicitation'. These often contain advice of the most general kind, and emphasize the pragmatic considerations of expert system development. General reviews can be found in Welbank (1983), Hoffman (1987), Kidd (1987) and Hart (1986). While these reviews are based on experience of the general kind, there have also been a number of attempts to make formal recommendations. Gammack and Young (1985) offer a mapping of knowledge elicitation techniques onto domain type. Their analysis requires that domain knowledge be separated into different categories, and provides suggestions about the particular techniques which are most likely to be effective in each category. Although the analysis alerts one to the fact that there are different types of knowledge with any domain, it does not provide engineers with guidelines about how to identify each type.

The most thorough attempt to integrate KE procedure is provided by KADS—Knowledge Acquisition and Domain Structuring (see Breuker, 1987 for an overview). KADS embodies seven principles for the elicitation of knowledge and construction of a system. These principles are stated below (following Breuker, 1987):

1. The knowledge and expertise should be analysed before the design and implementation starts.
2. The analysis should be model-driven as early as possible.
3. The content of the model should be expressed at the epistemological level.
4. The analysis should include the functionality of the prospective system.
5. The analysis should proceed in an incremental way.
6. New data should be elicited only when previously collected data have been analysed.
7. Collected data and interpretations should be documented.

Principle 1 is quite straightforward and requires no further explanation. Principle 2 requires that one should bring to bear a model of how the knowledge is structured early on in the process, and use it to interpret subsequent data. Principle 3 means that one should use an appropriate intermediate level representation device, and not try to force knowledge into a particular formalism before one knows how it may best be implemented. Principle 4 is a reminder that a complete analysis includes an understanding of how the system is to work, e.g. who will use it, and in what situation.

One cannot gain a full understanding of the problem simply by trying to map out an expert's knowledge without regard to how it will be used. Principle 5 emphasizes the fact that there is a wide variety of related topics within a domain. This means that construction of a model should be 'breadth-first', embodying all aspects at once, rather than attempting fully to represent one sub-part after another. Principles 6 and 7 are once again straightforward. Like many of the best recommendations, the utility of these statements is most apparent when they are not adhered to.

A basic insight from KADS is that knowledge acquisition should be viewed under the metaphor of model building, rather than the mining of information (Breuker, 1987). The principles above emphasize this point—in short, KE is not an independent part of constructing an expert system, and viewing it as such will lead to inefficient practice. Whether or not the adoption of such an approach makes for more efficient and effective KE is a moot point as these claims have not been formally evaluated. In so far as they impose a discipline on the KE process they are likely to be useful.

A rather less disciplined approach and yet one that is almost always associated with expert systems is *rapid prototyping* (Hayes-Roth *et al.*, 1983). Indeed some see the approach as specifying a separate elicitation technique. The idea is that it is easier for experts to criticize a working system, than it is to specify the system in the first place. Initially, a prototype is built, without much regard to its weaknesses, and the expert makes suggestions about its performance. These suggestions are incorporated into the system by programmers, and at the next session there should be fewer errors. This cycle continues until the expert is satisfied with the behaviour of the system (see Christie and Gardiner in this volume).

Finally, there has been more formal evaluation of KE techniques (cf. for example, Cooke and McDonald, 1987; Schweickert *et al.*, 1987; Burton *et al.*, 1987a, 1988). This type of research is still in its infancy. Where there are concrete and robust findings from the work we have tried to subsume them when describing the pros and cons of the techniques themselves.

Conclusions

The problem of knowledge elicitation is a subtle and complex one. We have described some of the techniques that are used in this enterprise. But we have also sought to provide an indication of the difficulties inherent in doing this kind of work. At present knowledge elicitation is itself a form of expertise—experienced knowledge engineers come to recognize the subtleties of expert thinking, and can develop skills for capturing these. However, it is clear that a formal methodology is absent.

As expert systems technology becomes more readily available, more people will have to face the issue of knowledge elicitation. In order to provide support for this enterprise research is under way in areas as disparate as software engineering, artificial intelligence and psychology. Experience to date has identified interesting and difficult problems which form the research agenda for these groups. In the mean time, prospective knowledge engineers will have to rely on accumulated experience, and general readings such as this one.

References

Anjewierden, A. (1987). Knowledge acquisition tools. *AI Communications*, **0**, 29–39.

Belkin, N.J., Brooks, H.M. and Daniels, P.J. (1987). Knowledge elicitation using discourse analysis. *International Journal of Man–Machine Studies*, **27**, 127–144.

Boose, J.H. (1985). A knowledge acquisition program for expert systems based on personal construct psychology. *International Journal of Man–Machine Studies*, **23**, 495–525.

Boose, J.H. (1986). *Expertise Transfer for Expert System Design* (New York: Elsevier).

Boose, J.H. and Bradshaw, J.M. (1987). Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems. *International Journal of Man–Machine Studies*, **26**, 3–28.

Breuker, J. (Ed.) (1987). *Model-Drive Knowledge Acquisition Interpretation Models*. Deliverable task AI, Esprit Project 1098 (University of Amsterdam).

Burton, A.M., Shadbolt, N.R., Hedgecock, A.P. and Rugg, G. (1987). A formal evaluation of knowledge elicitation techniques for expert systems: domain 1. In *Research and Development in Expert Systems IV*, edited by D.S. Moralee (Cambridge: Cambridge University Press).

Burton, A.M., Shadbolt, N.R., Rugg, G. and Hedgecock, A.P. (1988). Knowledge elicitation techniques in classification domains *ECAI-88: Proceedings of the 8th European Conference on Artificial Intelligence*, pp. 85–90.

Chi, M.T.H., Feltovitch, P.J. and Glaser, R. (1981). Categorisation and representation, physics problems by experts and novices. *Cognitive Science*, **5**, 121–152.

Chi, M.T.H., Glaser, R. and Rees, E. (1982). Expertise in problem solving. In *Advances in the Psychology of Human Intelligence: 1*, edited by R.J. Sternberg (Hillsdale, NJ: Erlbaum).

Cooke, N.M. and McDonald, J.E. (1987). The application of psychological scaling techniques to knowledge elicitation for knowledge-based systems. *International Journal of Man–Machine Studies*, **26**, 533–550.

Clarke, B. (1987). Knowledge acquisition for real time knowledge based systems. *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge Based Systems*, Reading University, UK.

Cleaves, D.A. (1987). Cognitive biases and corrective techniques: proposals for improving elicitation procedures for knowledge based systems. *International Journal of Man–Machine Studies*, **27**, 155–166.

Diederich, J., Ruhmann, I. and May, M. (1987). KRITON: a knowledge-acquisition tool for expert systems. *International Journal of Man–Machine Studies*, **26**, 29–40.

Elstein, A.S., Shulman, L.S. and Sprafka, S.A. (1978). *Medical Problem Solving: An Analysis of Clinical Reasoning* (Cambridge, MA: Harvard University Press). Cited in Kuipers and Kassirer (1983).

Ericsson, K.A. and Simon, H.A. (1984). *Protocol Analysis: Verbal Reports as Data* (Cambridge, MA: MIT Press).

Gammack, J.G. (1987a). Formalising implicit domain structure. *Proceedings of the SERC Workshop on Knowledge Acquisition for Engineering Applications.* SERC Report RAL-87-055.

Gammack, J.G. (1987b). Different techniques, and different aspects of declarative knowledge. In *Knowledge Acquisition for Expert Systems: A Practical Handbook*, edited by A.L. Kidd (New York: Plenum Press).

Gammack, J.G. and Young R.M. (1985). Psychological techniques for eliciting expert knowledge. In *Research and Development in Expert Systems*, edited by M. Bramer (Cambridge: Cambridge University Press).

Grover, M.D. (1983). A pragmatic knowledge acquisition methodology. *IJCAI-83: Proceedings 8th International Joint Conference on Artificial Intelligence.*

Hart, A. (1986). *Knowledge Acquisition for Expert Systems* (London: Kogan Page).

Hayes-Roth, F., Waterman, D.A. and Lenat, D.B. (1983). *Building Expert Systems* (Reading, MA: Addison-Wesley).

Hinkle, D.N. (1965). The change of personal constructs from the viewpoint of a theory of implications. Unpublished Ph.D. Thesis, University of Ohio.

Hoffman, R.R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology. *AI Magazine*, **8**, 53–66.

Jackson, P. (1985). *An Introduction to Expert Systems* (New York: Addison-Wesley).

Johnson, L. and Johnson, N. (1987). Knowledge elicitation involving teachback interviewing. In *Knowledge Elicitation for Expert Systems: A Practical Handbook*, edited by A.L. Kidd. (New York: Plenum Press).

Johnson, P.E., Zualkeman, I. and Garber, S. (1987). Specification of expertise. *International Journal of Man–Machine Studies*, **26**, 161–181.

Kahneman, D., Slovic, P. and Tversky, A. (Eds) (1982). *Judgement under Uncertainty: Heuristics and Biases* (New York: Cambridge University Press).

Kelly, G.A. (1955). *The Psychology of Personal Constructs* (New York: Norton).

Kidd, A.L. (Ed.) (1987). *Knowledge Acquisition for Expert Systems: A Practical Handbook* (New York: Plenum Press).

Kuipers B. and Kassirer, J.P. (1983). How to discover a knowledge representation for causal reasoning by studying an expert physician. *IJCAI-83: Proceedings of the 8th International Conference on Artificial Intelligence.*

Marcus, S., McDermott, J. and Wang, T. (1985). Knowledge acquisition for constructive systems. *IJCAI-85: Proceedings of the Ninth International Joint Conference on Artificial Intelligence.*

Michalski, R. (1983). A theory and methodology of inductive learning. In *Machine Learning: An Artificial Intelligence Approach*, edited by R. Michalski, J. Carbonell, and T. Mitchell. (Palo Alto: Tioga).

Michalski, R., Carbonell, J. and Mitchell, T. (Eds) (1986). *Machine Learning: An AI approach*, Volume 2 (Los Altos, CA: Morgan Kaufman).

Mitchell, T. (1982). Generalization as search. *Artificial Intelligence*, **18**, 203–226.

Motta, E., Eisenstadt, M., West, M., Pitman, K. and Evertsz, R. (1987). *KEATS: The Knowledge Engineer's Assistant*. HCRL Tech Report 20 (Milton Keynes: Open University).

Newell, A. and Simon, H.A. (1963). GPS, a program that simulates human thought. In *Computers and Thought*, edited by E. Feigenbaum and J. Feldman (New York: McGraw-Hill).

Quinlan, J.R. (1979). Rules by induction from large collections of examples. In *Expert Systems in the Micro-Electronic Age*, edited by D. Michie (Edinburgh: Edinburgh University Press).

Quinlan, J.R. (1983). Learning efficient classification procedures and their application to chess endgames. In *Machine Learning: An Artificial Intelligence Approach*, edited by R. Michalski, J. Carbonell and T. Mitchell (Palo Alto: Tioga).

Schvaneveldt, R.W., Durso, F.T., Goldsmith, T.E., Breen, T.J., Cooke, N.M., Tucker, R.G. and De Maio, J.C. (1985). Measuring the structure of expertise. *International Journal of Man–Machine Studies*, **23**, 699–728.

Schweickert, R., Burton, A.M., Taylor, N.K., Corlett, E.N., Shadbolt, N.R. and Hedgecock, A.P. (1987). Comparing knowledge elicitation techniques: a case study. *Artificial Intelligence Review*, **1**, 245–253.

Shadbolt, N.R. (1989). Knowledge representation in man and machine. In *Expert Systems: Principles and Case Studies*, 2nd edition, edited by R. Forsyth (London: Chapman and Hall).

Shapiro, A. (1987). *Structured Induction in Expert Systems* (New York: Addison-Wesley).

Shaw, M.L.G. and Gaines, B.R. (1987a). An interactive knowledge elicitation technique using personal construct technology. In *Knowledge Acquisition for Expert Systems: A Practical Handbook*, edited by A.L. Kidd (New York: Plenum Press).

Shaw, M.L.G. and Gaines, B.R. (1987b). KITTEN: Knowledge initiation and transfer tools for experts and novices. *International Journal of Man–Mxchine Studies*, **27**, 251–280.

Weiss, S. and Kulikowski, C. (1984). *A Practical Guide to Designing Expert Systems* (Towata, NJ: Rowman and Allanheld).

Welbank, M.A. (1983). *A Review of Knowledge Acquisition Techniques for Expert Systems* (Martlesham Heath: British Telecom Research).

Young, R.M. (1987). *The Role of Intermediate Representations in Knowledge Elicitation*. Keynote Address to 'Expert Systems 87', Brighton, UK.

Young, R.M. and Gammack, J. (1987). Role of psychological techniques and intermediate representations in knowledge elicitation. *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge Based Systems*, Reading University, UK.