Extracting expertise from experts: Methods for knowledge acquisition

Abstract: *Knowledge acquisition is the biggest bottleneck in the development of expert systems. Fortunately, the process of translating expert knowledge to a form suitable for expert system development can benefit from methods developed by cognitive science to reveal human knowledge structures. There are two classes of these investigative methods, direct and indirect. We provide reviews, criteria for use, and literature sources for all principal methods. Direct methods discussed are: interviews, questionnaires, observation of task performance, protocol analysis, interruption analysis, closed curves, and inferential flow analysis. Indirect methods include: multidimensional scaling, hierarchical clustering, general weighted networks, ordered trees, and repertory grid analysis.*

JUDITH REITMAN OLSON

Graduate School of Business Administration
The University of Michigan
Ann Arbor
Michigan
USA

HENRY H. RUETER

Vector Research Inc
Ann Arbor
Michigan
USA

1. Introduction

Expert systems are here to stay. Although they are out of fashion in the research laboratories, they are nonetheless growing in popularity inside business and industry. Armies of small expert systems are being built to solve routine, medium-difficult problems in areas such as diagnosis of engine failures, tax planning, and feasibility analysis of cases in union disputes about seniority. The literature on how to build expert systems is burgeoning. People flock to seminars and tutorials at professional conferences because they need training on how to build systems for very practical ends.

It has long been recognized that the system development process for expert systems and other artificial intelligence applications is different from that for standard software. Whereas standard software development has a small fraction of the time spent in planning and a lot of coding, the majority of the time in expert system development is in planning — in deciding what knowledge should be encoded into the system. The bottleneck in the development of expert systems is in extracting the knowledge from the expert, that is, in knowledge acquisition [1] and [2]. Even books about how to build expert systems have very little to offer about knowledge acquisition.

Expert system developers have relied on two standard methods for getting the knowledge out of the expert and into the system: 1) The developer, or knowledge engineer in this case, engages in intense interview with the expert, or 2) the knowledge engineer becomes an expert him/herself, relying on introspection to articulate the requisite knowledge. The knowledge engineer then encodes the knowledge into a language of choice, encoding facts into OBJECT-ATTRIBUTE-VALUE triples and inferences into IF-THEN rules. When run, the inference engines grind away at these knowledge structures working either with facts (e.g. the features that the particular case under diagnosis exhibits) *forward* to the goal, or working with the goal (e.g. one of several known diagnosis categories) *backward* to the facts.

Simple OBJECT-ATTRIBUTE-VALUE triples and forward and backward search strategies are a small subset of the knowledge structures and search strategies human experts have. Expertise is primarily a skill of recognition, of 'seeing' old patterns in the new problem. Chess experts, for example, have the same limited abilities as novices to hold information for analysis and look ahead only a limited number of steps. They excel because they have hundreds of thousands of chess configurations in their memories, and can quickly encode the current situation into constellations of previously-seen chess patterns. The choice of candidate 'good moves' for the expert is thus restricted to a small set of known good moves that fit the patterns, whereas the novice has no such expert pattern knowledge to filter out bad candidates [3], [4] and [5].

There is also evidence to suggest that experts see more richly encoded patterns than novices do. They have organized the concepts in their knowledge bases with much more depth and with many more central associations than novices. For example, expert Algol programmers had much more structure in the relationships among concepts held in memory than novices did. And, experts' organizations were highly similar, whereas novices' structures were scattered, based on a variety of irrelevant prior associations [6].

Not only do experts have information organized in a highly structured manner, but they use a variety of *kinds* of knowledge structures. Some things are stored in simple *lists*, e.g. the months of the year and the days of the week. Some information fits a stored *table* better, information such as calendar appointments and the periodic table. Some information is stored as a flow diagram, such as a *decision tree*, for example, representing the routing of telephone messages to the people who can handle them. There is information stored in *hierarchies* of relationships, nested categories or clusters, such as animal taxonomies. *Networks* store richly connected language associations. Such information as room arrangements or maps may be stored as *physical space*. And, some information may be stored about a device's internal components and how they are causally related as a *physical model*, commonly referred to as a mental model. Experts may hold what they know about objects in a myriad of different representations, each suitable for a particular kind of reasoning or retrieval.

Not only do experts see problem situations differently, they may also search differently through intermediate problem states. Recent work in medical reasoning, for example, has shown that experts primarily work forward through the problem (beginning with a 'good' representation), and that novices work backward from the goal. Furthermore, from work on planning, we see that experts also plan ahead loosely, filling in details when the situation dictates, and that they move back and forth from abstract to detailed entities when evaluating tactics and strategies [7] and [8].

Typically the knowledge engineer has only OBJECT-ATTRIBUTE-VALUE and IF-THEN rules to use to encode what the expert knows. If in acquiring the expert's knowledge, the knowledge engineer focuses only on knowledge that can be easily encoded in these forms, then significant expertise will be missed. The process of translating expert knowledge to this form will benefit greatly from a knowledge of 1) the fact that many different expert structures and processes are possible, and 2) which tools are appropriate to uncovering these various structures.

The purpose of this paper is to convey to the knowledge engineer the myriad of methods used in cognitive science research for revealing expert knowledge structures and processes. Some of these methods may be useful in the developing expert systems. The ultimate goal, however, is to alert the knowledge engineer to the fact that simple, intense, time-consuming interviews of experts leading to the coding of OBJECT-ATTRIBUTE-VALUE triples and IF-THEN rules may be both misleading and costly as well as difficult. Experts have rich structures and reasoning abilities. Truly representing the expert's expertise depends on knowledge of these structures and abilities.

2. Methods for knowledge acquisition

There are two classes of methods for revealing what experts know. 'Direct methods' ask the expert to report on knowledge he/she can directly articulate. This set of methods includes interviews, questionnaires, simple observation, thinking-out-loud protocols, interruption analysis, drawing closed curves, and inferential flow analysis. In contrast, 'indirect methods' do not rely on the expert's abilities to articulate the information that is used; they collect other behaviors, such as recall or scaling responses from which the analyst can make inferences about what the expert *must have known* in order to respond the way he/she did. 'Indirect methods' include multi-dimensional scaling, hierarchical clustering, general weighted networks, ordered trees from recall, and repertory grid analysis.

2.1 Direct methods

2.1.1 Interviews. Interviews are the most common method for eliciting knowledge from the expert. In conversation, the expert reveals the objects he/she thinks about, how they are related or organized, and the processes he/she goes through in making a judgment, solving a problem, or designing a solution. There are simple guidelines that can be followed to make interviewing efficient.

1. Enlist the expert's cooperation. Interviewing an unwilling expert dooms the project to failure. There is a number of reasons why an expert may not be cooperative. First, the expert may know that he/she performs the task in simple, intuitive ways that, if revealed, would reduce the esteem others hold for him/her. Second, the expert may not know how he/she performs that task and may be reluctant to express the uncertainty, thinking that experts are expected to be rational and articulate. And third, the expert may believe that if his/her expertise can be captured in a computer, his/her job will be eliminated. On the other hand, the expert may be flattered to think that the company is willing to invest the time and money to clone the expertise, to allow many more problems to be solved with the knowledge this one person has gained.

2. Ask free-form questions at the start, narrowing in specificity as the interview process progresses. The goal at first is to discover the vocabulary the expert uses and to allow him/her to begin to articulate the inferences drawn and the relationships seen. Asking, "How do you do your task?" can be followed with, "Recall the last case..." The interviewer should note the order in which the expert addresses topics, the relative importance of the way evidence is weighed.

3. Do not impose your own understanding on the expert. Allow the expert to talk, even if the current topic seems tangential to the main purpose. Do not interrupt. Ask about what you do not understand, but do not impose your own, naive bias on what the expert is saying.

4. Limit the sessions to coherent tasks, recognizing fatigue and attentional limits. The interviewer should be aware of the effort that goes into answering questions about expertise, and make sessions tolerably long. However, breaking in the middle of a thought or problem is unnatural and disturbing. It is almost impossible to 'take up where we left off' if major questions are not settled on a problem or task at the end of the previous session.

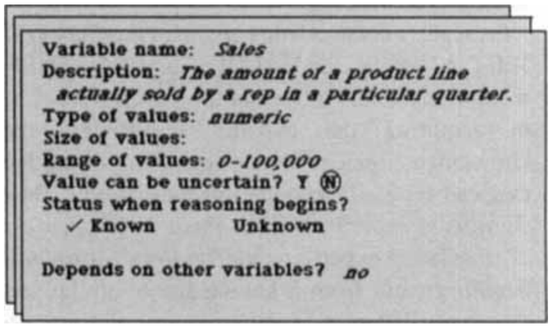
The interview questions should center on the expert's knowledge of objects, relationships, and inferences. Good questions to ask might be:

"What kinds of things do you like to know about when you begin to ponder the problem?"
"What facts or hypotheses do you try to establish when thinking about a problem?"
"What are the factors that influence how you reason about a problem?"
"What type of values can this object have? What range of values is permissible?"
"Does this factor depend on other factors? If so, which ones?"
"Is this factor needed for solving all problems in the domain or for just some?"[1]

It is probably best to begin knowledge acquisition with an interview, to identify a set of objects and their relationships (noting especially whether the expert is thinking of these objects in special relationships like lists, tables, physical spaces, etc). However, the specificity and completeness of what can be obtained soon reaches a limit in the free form style. To alleviate this problem, often the interviewer can mix in questions of other styles, such as focusing on a particular case in detail, called the 'critical incident technique'. Focusing on a case elicits particular descriptions, rules and objects, which can be examined for their *generality* in later sessions. By asking for 'symptoms' and 'characteristics' one elicits features, while asking for 'evidence' elicits inferences.

2.1.2 Questionnaires. Interviews have a distinct advantage in that they can elicit unforeseen information. Interviews are free form; experts can generate information in the order they wish in the detail they wish. Experts are in control. Interviews, however, are very time-consuming. Questionnaires, on the other hand, have the advantage of being a very efficient way to gather information. Furthermore, the expert can fill out questionnaires in a leisurely and relaxed atmosphere. Questionnaires can be particularly useful in discovering the *objects* of the domain, in uncovering relationships, and perhaps in determining uncertainties, if the expert system attaches uncertainty to its conclusions.

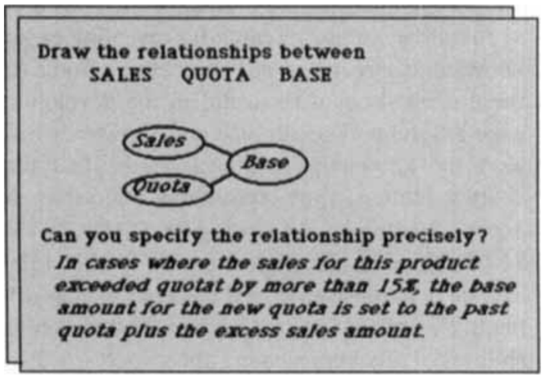
The questionnaires suggested for use here are not the kind used in survey research. Instead, they consist of cards or pieces of paper on which are printed some standard, but open-ended questions. Figures 1 and 2 illustrate cards used to elicit variables (in Figure 1 the object 'sales' is defined) and relationships (in Figure 2 the relationship between 'sales', 'quota', and 'base'



Variable name: *Sales*
Description: *The amount of a product line actually sold by a rep in a particular quarter.*
Type of values: *numeric*
Size of values:
Range of values: *0-100,000*
Value can be uncertain? Y [✓ N]
Status when reasoning begins?
✓ Known Unknown

Depends on other variables? *no*

Figure 1. *Questionnaire card for variable elicitation [9]*



Draw the relationships between
SALES QUOTA BASE

Sales
Base
Quota

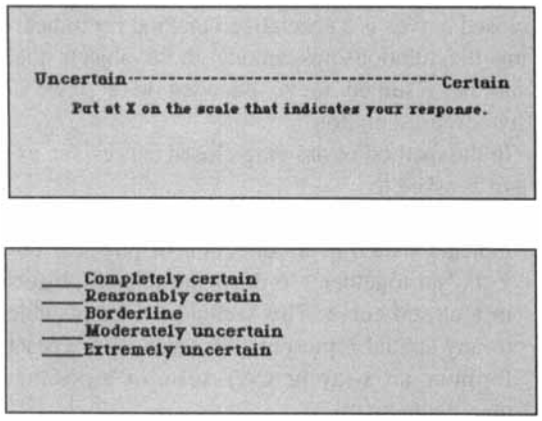
Can you specify the relationship precisely?
In cases where the sales for this product exceeded quotat by more than 15%, the base amount for the new quota is set to the past quota plus the excess sales amount.

Figure 2. *Questionnaire card for relationship elicitation [9]*

[1] From [9]

is drawn.)

Figure 3 illustrates the use of questionnaires to elicit uncertainties about particular inferences an expert has reported. Questionnaires are particularly appropriate for eliciting uncertainty information, since normal verbal responses from people are not very reliable in revealing this kind of information. People are not good at estimating probabilities; they overestimate low ones and underestimate high ones [10]. Consequently, simply asking for probabilities in an interview is not effective. Eliciting probability estimates using pre- formatted response scales can yield much more accurate estimates. There are two preferred formats: 1) the bar on which the expert indicates a point to reflect uncertainty, and 2) a five point verbal scale on which the expert marks the work most closely associated with his/her certainty. The five-point scale is one taken from Meister's compilation of verbal scales possessing equal spacing and reliability of measurement [11].



Uncertain-----------------------------------Certain
Put at X on the scale that indicates your response.

____Completely certain
____Reasonably certain
____Borderline
____Moderately uncertain
____Extremely uncertain

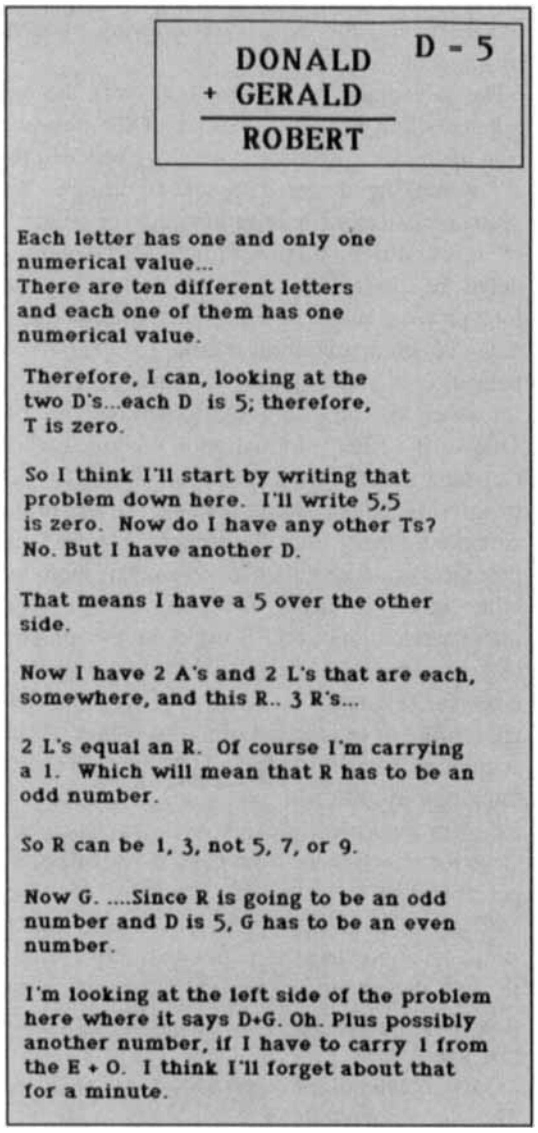
Figure 3. *Response scale for uncertainty elicitation [9]*

2.1.3 Observation of the task performance. Often the best way to discover how an expert makes a judgment, diagnosis, or design decision is to watch the expert work at a real problem. In this situation, the knowledge engineer has several ways of discovering the objects, relationships, and inferences that the expert is using. The first decision that must be made is how to record the expert's performance. One possibility is simply to watch, take notes, and try to follow the expert's thinking process on the fly. A second possibility is to videotape the process for later review with the expert. In choosing between these methods, remember that the first method suffers from time pressure and observer bias, while the second relies on the expert's less than perfect ability to recall the reasons underlying his/her performance.

2.1.4 Protocol analysis. A close cousin of simple observation is protocol analysis. Like observation, above, the expert engages in normal task behavior with particular typical problems. In addition to video-recording the session and annotating the behaviors after the fact, the knowledge engineer asks the expert to 'think out loud' while performing the task. The expert is to answer the questions, "What are your goals?" "What are your methods?" "What are you seeing?" Inference is then drawn from a transcript of this session about the objects, their relationships, and the inferences the expert was drawing moment by moment.

The advantage of this method over the annotated silent task performance of the observation method is that there is no delay between the act of thinking of something and reporting it. But protocol analysis is not appropriate for all kinds of tasks. Ericsson and Simon [12] carefully detail the kinds of tasks for which thinking-out-loud protocol might be acceptable, useful kind of data. To summarize, those tasks for which verbalization is a natural part of thinking are those for which we can take thinking-out-loud as data. That is, if verbal information is produced while someone makes inferences to him/herself, or in identifying salient features of the objects in the situation, then the information from the protocols is acceptable data. However, there are other kinds of tasks, those for which some idiosyncratic language is used in the process (e.g. in composing music, composers often have a special language for the parts of the piece they are writing or the section of the style they are instantiating currently), for which the process of thinking-out-loud and explaining might be distorted or even wrong. And, of course, there are tasks for which there is no natural verbalization; perceptual-motor tasks are examples of these. Verbalization of perceptual-motor tasks makes someone attend to aspects not normally attended to, and the attention required to report on the process usurps resources normally devoted to the task itself.

Once obtained, protocols must be analyzed. The goal in obtaining a protocol lies in identifying the kinds of objects the expert sees, the relationship that exists between the protocols, and the kinds of inferences drawn from the relationships seen. For example, in the protocol in Figure 4, the problem solver is trying to find a solution to the 'cryptarithmetic problem' DONALD+GERALD=ROBERT. In cryptarithmetic, each letter can be mapped into one digit such that the arithmetic operations apply. The problem solver has been given the fact that D=5. In the protocol, the analyst looks for the change in focus of attention: it moves from working forwards ("each D is 5; therefore, T is zero...Now do I have any other Ts? No, but I have another D"). Later it reveals some working backwards (...Since R is going to be an odd number and D is 5, G has to be an even number). One sees a shift in goals and subgoals, the kinds of things the problem solver is paying attention to, and the kinds of inferences made [13].



DONALD D - 5
+ GERALD
ROBERT

Each letter has one and only one numerical value...
There are ten different letters and each one of them has one numerical value.

Therefore, I can, looking at the two D's...each D is 5; therefore, T is zero.

So I think I'll start by writing that problem down here. I'll write 5,5 is zero. Now do I have any other Ts? No. But I have another D.

That means I have a 5 over the other side.

Now I have 2 A's and 2 L's that are each, somewhere, and this R.. 3 R's...

2 L's equal an R. Of course I'm carrying a 1. Which will mean that R has to be an odd number.

So R can be 1, 3, not 5, 7, or 9.

Now G. ....Since R is going to be an odd number and D is 5, G has to be an even number.

I'm looking at the left side of the problem here where it says D+G. Oh. Plus possibly another number, if I have to carry 1 from the E + O. I think I'll forget about that for a minute.

Figure 4. *Cryptarithmetic problem [13]*

2.1.5 Interruption analysis. One way of preserving the natural thought process of the problem solver is to let him/her proceed without thinking aloud. But, when the process gets to a point where the observer can no longer understand the expert's thought processes, the observer interrupts. At that point, the observer asks in detail why the expert did what he/she did, trying to capture *at the moment* the focus of attention and the kinds of inferences drawn for the features noticed [14]. This process can be very instructive about the process observed, but, or course, once a process has been interrupted, there is very little chance that it can be restarted. This procedure is likely to give most of its value after the expert system has been coded in its prototype stage, and the expert's performance is being compared to that of the system.

2.1.6 Drawing closed curves. The previous methods attempt to reveal the contents of the thought processes during the solution of existing problems. They highlight the vocabulary the expert uses to identify the objects and their relationships; they highlight the kinds of inferences drawn. These methods are free of assumptions about the form of the relationships among the items, be they lists or tables or networks or physical space. In contrast, the method of drawing closed curves is a specialized method for indicating the relationships among those objects that can be assumed to be encoded in a *physical space* representation.

In the method of drawing closed curves, the expert is asked to

indicate which of a collection of physical objects 'go together', to draw the related objects in a closed curve. This technique is applicable to any spatial representation, such as a typeset formula, an x-ray or CAT scan, or a position on a game board.

For example, a Go expert drew closed curves around the stones on a position in the chess-like game of Go [5]. Figure 5 illustrates several aspects of his responses. Four positions are displayed, A–D. Inside each stone is a number which represents the ordinal position in which that stone was placed on the board in a recall task. Note that the order matches the closed curves to a remarkable degree; all stones of a chunk are recalled before moving on to another chunk. Furthermore, on the right side of the Figure are shown three successive recall trials of the same position. As above, the numbers indicate the order in which the stones were placed in recall trial. It is noteworthy that groups of stones, indicated by closed curves on a separate occasion, are recalled consistently together before other groups are recalled. This regularity of behavior suggests the validity and reliability of the information contained in the originally drawn closed curves.

2.1.7 Inferential flow analysis. A variant on the interview is the method called inferential flow analysis. In this method, answers to particular questions about causal relations are used to build up a causal network among concepts or objects in the domain of expertise. Salter [15] used this technique to uncover laymen's models of economics.

This technique begins with a list of some of the key objects in the domain of expertise. In Salter's case, this list contained such items as business borrowing, personal savings rate, productivity, etc. Salter then asked an interviewee a series of pointed questions about the relationship between two of the objects. For example, he would ask, "What is the relationship between savings rate and inflation?" Answers revealed the linkages among items between these two key objects and the direction of the relationships. For example, the person might respond, "If inflation goes up, savings rates will go down, because savings interest rates are lower than the amount one can save by buying now instead of later." These two items are then linked in an inverse relationship, in which purchasing is an intervening variable.

Responses to a set of questions should uncover some consistency in the relationships between intervening variables. Each time an item is mentioned in an answer it is linked with the other items in the answer, with the links labelled positive or negative as indicated. Linked items are joined into one all-inclusive network of relations. At the first mention of a link, a standard weight is given to the link, e.g. 0.50. With each succeeding mention the link is raised in strength, some proportion of the remaining strength between the current value and 1.00. The resulting relations are displayed as a network, such as the one displayed in Figure 6 overleaf.

Although this technique appears somewhat *ad hoc*, the resulting networks have been shown to be both stable and consistent with other sets of behaviors [15]. This technique is simple to apply and powerful as a tool for displaying *to the expert* aspects of the expertise he/she has uncovered to that point. This display can be used effectively as a stimulus for further interviews.

2.2 Indirect methods

All of the previously described methods ask the expert directly what he/she knows. They rely on the availability of the information to *both* introspection and articulation. Of course, it is not always the case that the expert has access to the details of his/her knowledge or mental processing. In fact, it is not uncommon for experts to perceive complex relationships or come to sound conclusions without knowing exactly how they did it. In these cases, indirect knowledge elicitation methods are required.

In all the following methods, experts are not asked to express their knowledge directly. Instead, they are given a variety of other tasks, e.g. to rate how similar these two objects are, or to recall all these objects several times from several starting points. From the results, the analyst then

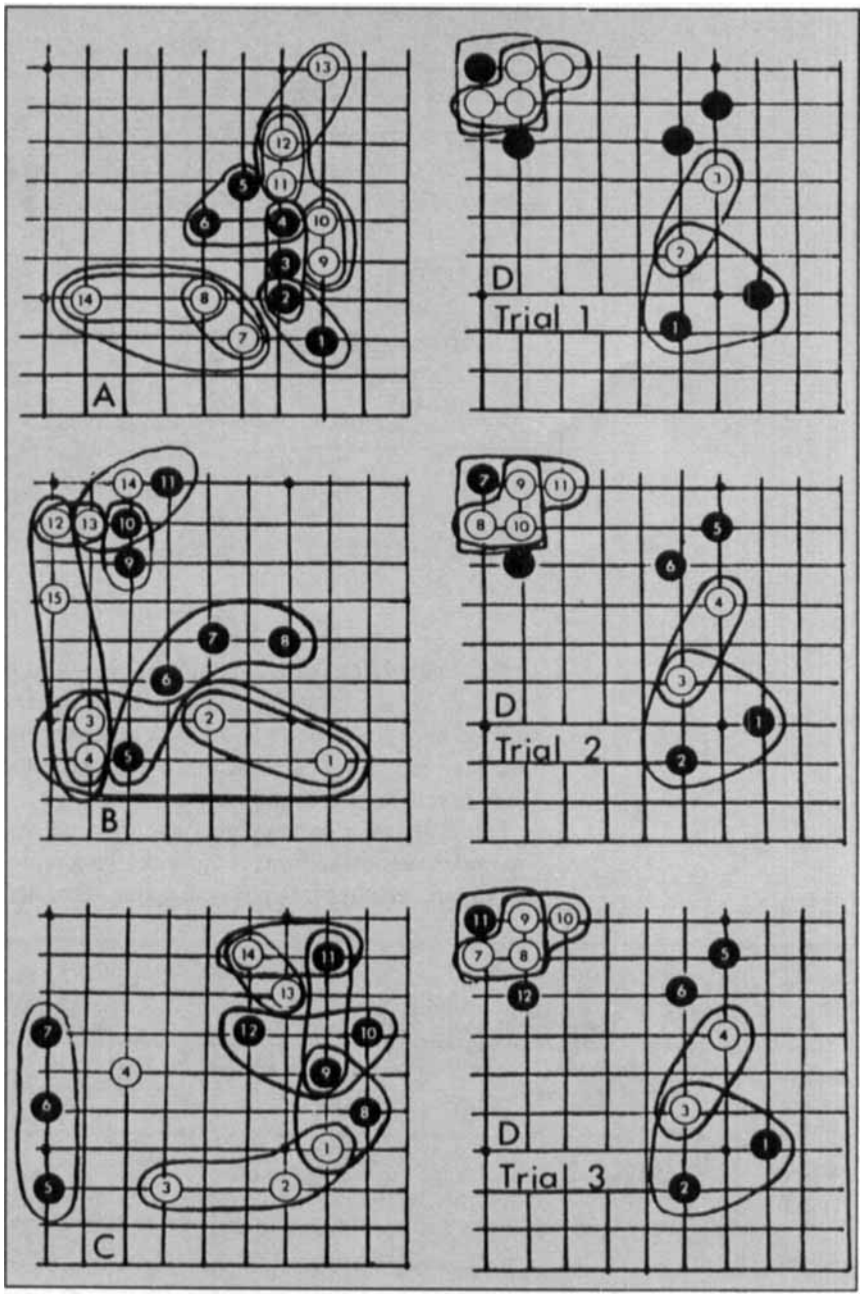


Figure 5. *Closed curves drawn by Go expert [5]*

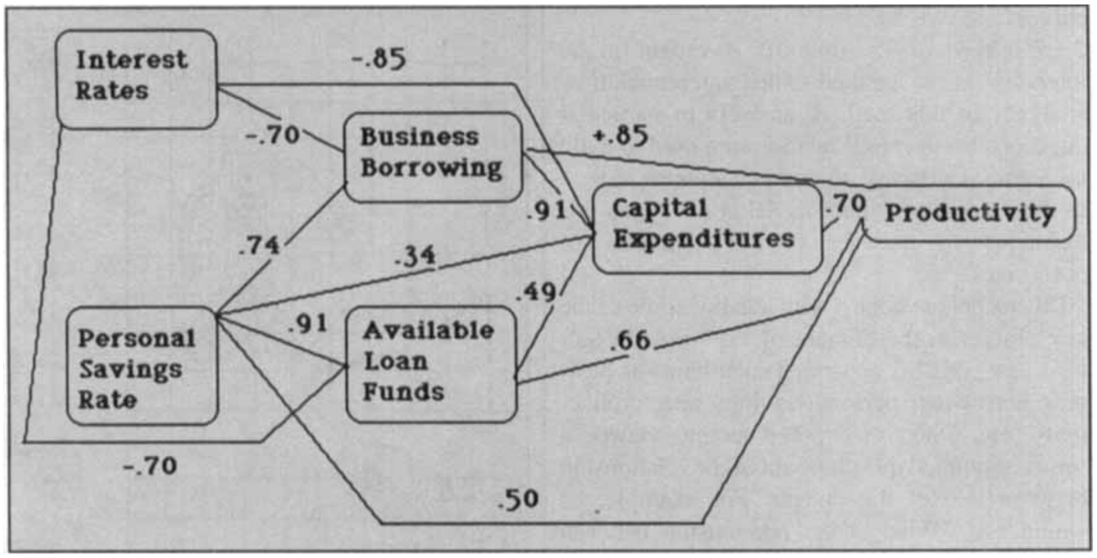
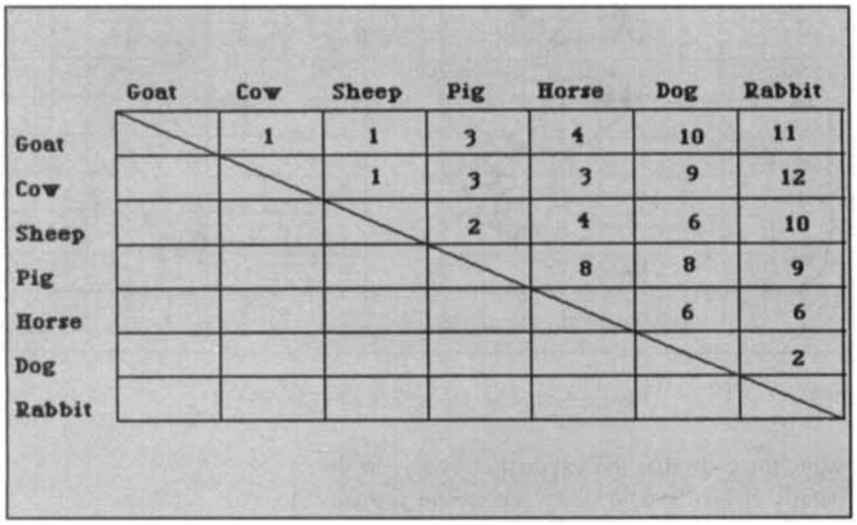


Figure 6. *Inferential flow network*

infers underlying structure among the objects rated or recalled. All the indirect methods discussed here have been validated in experimental studies that have convincingly demonstrated their psychological validity.

These different techniques make different *assumptions* about the form of the underlying representation, whether it is physical space, lists, networks, or tables, etc. It is important to use only those methods for which the assumptions fit the analysts' best guess as to what form the expert's underlying representation is in. This information can be gleaned from initial interviews with the expert as well as from careful questioning and noting of object names and/or notations the expert makes.



	Goat	Cow	Sheep	Pig	Horse	Dog	Rabbit
Goat		1	1	3	4	10	11
Cow			1	3	3	9	12
Sheep				2	4	6	10
Pig					8	8	9
Horse						6	6
Dog							2
Rabbit							

Figure 7. *Similarity judgment matrix*

2.2.1 Multidimensional scaling. Multidimensional scaling (MDS) is a technique that should be used only on data that are assumed to have come from stored representations of *physical n-dimensional space* [16]. The subject provides similarity judgments on all pairs of objects or concepts in the domain of inquiry. These judgments are assumed to be symmetric and graded; i.e. A is as similar to B as B is to A, and the similarities are assumed to take on a variety of continuous values, not just 0 or 1.

The scaling technique produces a layout of the items in space. All objects of a particular target domain are paired with all other objects in a set of queries to the expert: "How similar are A and B?" These similarity judgments are arrayed in a half-matrix such as that shown in Figure 7. The matrix in Figure 7 is *part* of a matrix that compares pairs of common farm and wild animals. This matrix is then the input to an analysis program which searches for the best placement of these objects in space of user-specified dimension. Each dimensional solution has a 'stress' associated with it, a measure of the deviation from a perfect fit.

The analyst looks for solutions with low 'stress'. Those using fewer dimensions are then plotted. (For higher dimensions, some of the more illuminating two dimensional projections can be drawn). The analyst then examines the plots to judge the 'best' placement of the axes and a plausible labelling for them. In Figure 8, the solution to a fuller similarity matrix of the form in Figure 7, the two axes might be recognized as 'size' (the abscissa) and 'ferocity' (the ordinate).

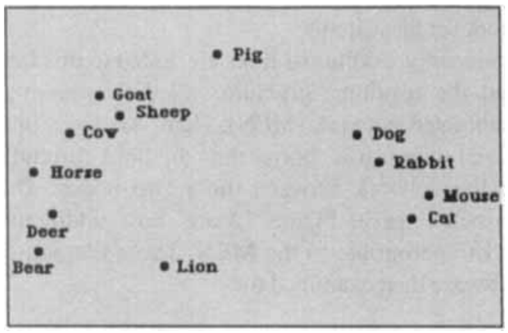


Figure 8. *Multidimensional scaling solution in two dimensions*

This technique is good for producing a diagram that the expert may then inspect and describe in more detail. It can reveal interesting clusters of objects, neighbor relations, and outliers. One difficulty with this technique, however, is the tedium of collecting the required pair-wise similarity judgments; for n objects, n(n-1)/2 judgments are required, a number that quickly grows to the hundreds and thousands with more than a few objects. Furthermore, it is difficult for the analyst to find the dimensionality with the best 'stress' value, and then perceive the best placement of the axes and the axes' names. Using the technique is straightforward; interpreting the results is not.

2.2.2 Johnson hierarchical clustering. Like MDS, hierarchical clustering begins with a half-matrix of similarity judgments. The *assumptions* for this technique, however, are in direct contradiction to those for multidimensional scaling. Whereas multidimensional scaling assumes symmetric distances and graded properties, hierarchical clustering assumes merely that an item is or is not a member of a cluster. Judgments are assumed to be a function of the number of nested clusters two items have in common, or the 'height' at which two items become members of the same superordinate category. Items cannot at once satisfy assumptions for both multidimensional scaling *and* hierarchical clustering [17]. [1]

Johnson hierarchical clustering is a fairly simple, straightforward algorithm that starts with the half-matrix of distances and ends with a hierarchical representation of the items. In broad strokes, pairs of items that are the closest in the matrix are joined to a single cluster, a new matrix drawn with this cluster serving as a new 'item'. This new matrix is examined again for that pair of 'items' that is closest together. These are joined as if the next new 'item', and a new matrix drawn. Each time a new matrix is drawn, interitem distances among unclustered items are copied from the original matrix to the new; distances between items and clusters are calculated as either the *minimum* distance of all cluster items to the item, the *maximum*, or the *average*. For example, using the matrix in Figure 7, the items that are the closest are COW-SHEEP-GOAT. The value in the re-written matrix for the distance between this cluster and PIG, for example, using the *minimum* joining algorithm, assigns '2' to the distance, the minimum of 2, 3 and 2 (for PIG-GOAT, PIG-COW, and PIG-SHEEP, respectively).

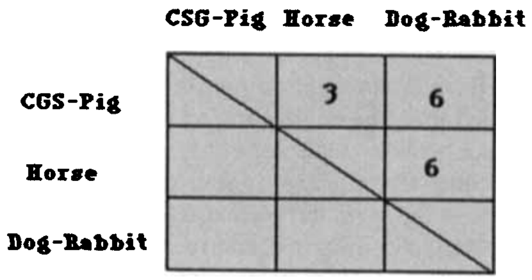
Figure 9 shows the full rewritten matrix with the COW-SHEEP-GOAT cluster now serving as an 'item'. In this matrix, PIG joins COW-SHEEP-GOAT (at 2) as does DOG to RABBIT. This matrix is rewritten into Figure 10. In Figure 10, HORSE joins the ((COW-GOAT-SHEEP)-PIG) cluster at 3, and the two clusters join at 6. The completed hierarchy is shown in Figure 11.

	Cow Sheep-Goat	Pig	Horse	Dog	Rabbit
Cow-Sheep-Goat		2	3	6	10
Pig			8	8	9
Horse				6	6
Dog					2
Rabbit					

Figure 9. *Similarity matrix of Fig. 7 with Cow-Sheep-Goat cluster*

An advantage of hierarchical clustering is that it can be done with paper and pencil. Unfortunately, it begins with a distance matrix that is just as tedious to collect as that for multi- dimensional scaling. Furthermore, without some theoretical justification for choosing a particular joining algorithm (the minimum, maximum, or average) one must choose arbitrarily; unfortunately, different algorithms can produce remarkably different hierarchies. In this sense, the analysis is somewhat subjective.

[1] Unfortunately, some people routinely do a multi-dimensional scaling solution in two dimensions (arbitrarily) and then indicate the nested curves found in Johnson hierarchical clustering by drawing closed curves around chunk elements. This cannot be done if one adheres to the underlying assumptions of both techniques.



	CSG-Pig	Horse	Dog-Rabbit
CGS-Pig		3	6
Horse			6
Dog-Rabbit			

Figure 10. *Similarity matrix of Fig. 9 after further clustering*

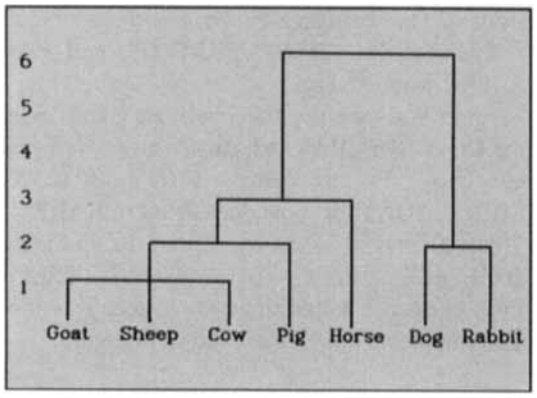


Figure 11. *Final cluster hierarchy for animal example*

2.2.3 General weighted networks. Like the preceding two techniques the expert gives symmetric distance judgments on all possible pairs of objects. These distances are assumed to arise from the expert traversing a network of associations, a network in which there is a single primary path between every two items, and, for some of them, a differently encoded, secondary path between them as well. A recent investigation using networks is [19].

From the distance matrix, a minimal connected network (MCN) is first formed. This network is formed by connecting the most closely linked items, such as COW-SHEEP in Figure 7.[1] In Figure 12, the solid lines show the resulting network for these items.

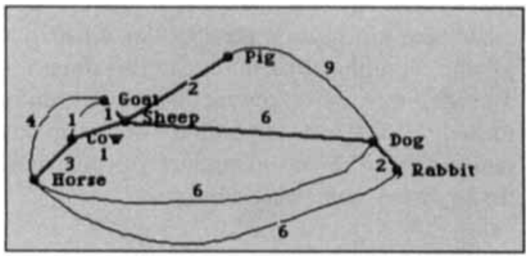


Figure 12. *MCN and MEN for animals example*

Secondly, additional links are added to this tree and the resulting structure called the minimal elaborated network (MEN). Here, we add a link if and only if it is shorter than the links currently in the network between those two nodes. The dashed lines in Figure 12 are those additional links appropriate to the MEN. These two structures are then examined for

1) *dominating concepts* — those that have a large number of connections to many other nodes, and
2) *members of cycles* — those items that are fully linked into circles.

In Figure 12, SHEEP is a dominating concept, the one with the most primary links, HORSE is somewhat less dominating, because although it has many links, most are from the elaborated network.

Figures 13 and 14 show the results of an exploration of the MCN and MEN for expert and novice pilots, rating a set of terms having to do with 'split plane concepts' [20]. Figure 13 illustrates the network for an expert; Figure 14 shows the network for the same concepts held by a novice. Several things were noted in the study:

Experts' structures were simpler than students'.
Elaborated links connected integrated larger conceptual structures.
Experts could easily identify link relations, using such terms as 'Affects', 'Is-a', 'Desirable', 'Acceptable', etc.

The fact that the experts were so clearly different from the novices suggests that this technique can reveal significant aspects of expertise, aspects that clearly should be encoded into an expert system.

2.2.4 Ordered trees from recall. Ordered trees come from work by Reitman and Rueter [21] in their exploration of how memory organizations differ among experts and novices in a particular domain. Unlike the indirect methods described above, ordered trees begin not with a distance matrix but with recall trials. The technique assumes that objects belong to a cluster or not, similar to the assumption of hierarchical cluster-

[1] SHEEP is taken as the most central item in this 'tied' cluster because it is, on average, closer to all other items than either GOAT or COW.

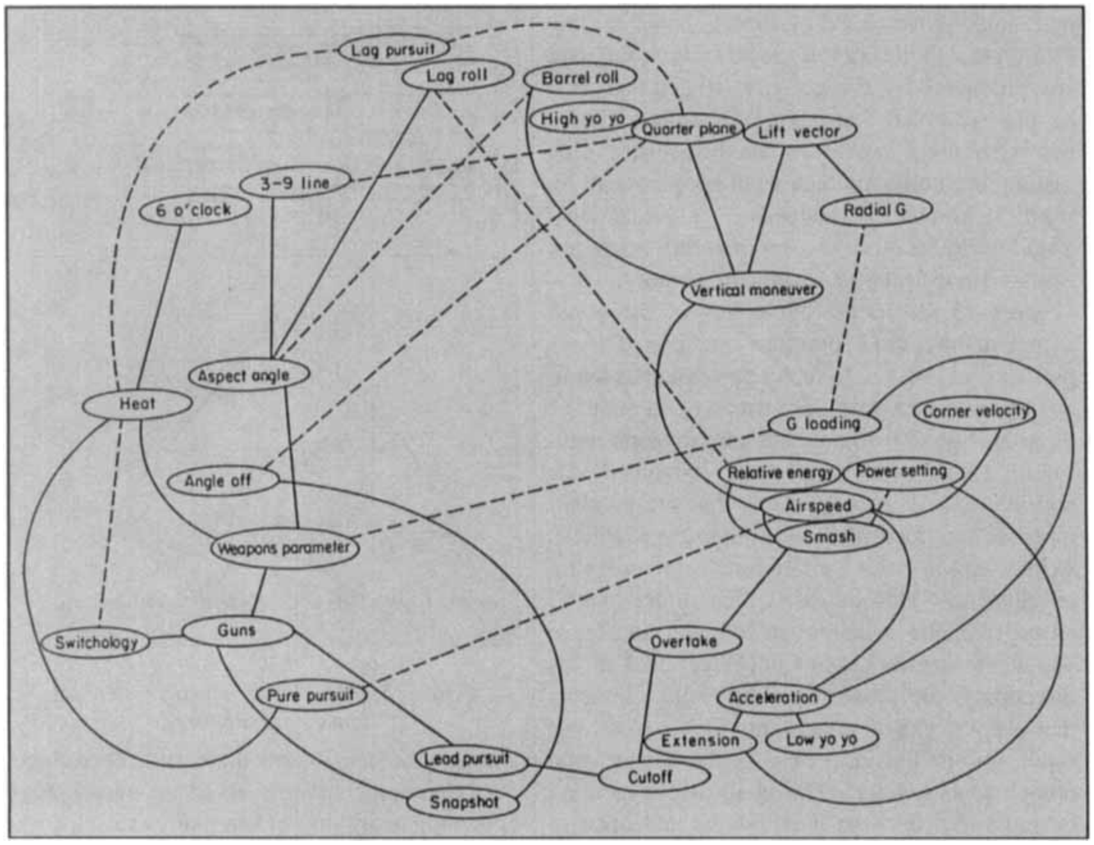


Figure 13. *MCN and MEN for expert fighter pilots [20]*

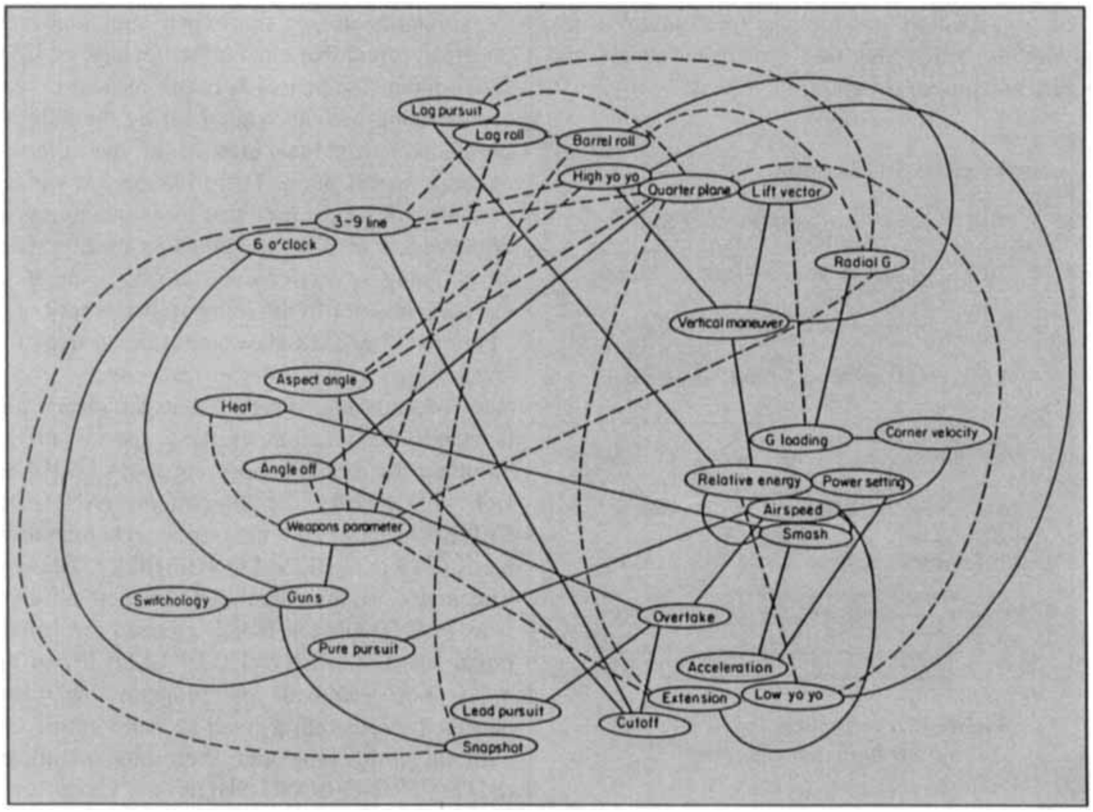


Figure 14. *MCN and MEN for novice fighter pilots [20]*

ing. Unlike hierarchical clustering, however, this technique is built upon a model of how the data are produced by the subject: it assumes that people recall all items from a stored cluster before recalling items from another cluster. This assumption builds on data from people recalling from known (learned) organizations. Regularities found over a set of recall orders are assumed to reflect organization in memory.

Figure 15 illustrates four orders of the seven animal names used in previous examples. The expert/subject is asked to recall object names ten to twenty times; to encourage variety, on some of the trials he/she is told which item to begin with. These recall trials are then examined for regularities. All sets of items that are recalled together are identified as chunks, the chunks written into a lattice (ordered inclusion covering relationship). This lattice is then re-drawn into an ordered tree structure, such as that in Figure 16, where arrows (either uni-directional or bi-directional) are drawn over the chunk elements that were recalled consistently in a particular order (or, in the case of a bi-directional, one order and its reverse). This analysis can be done by hand, but because it is tedious and open to perceptual error, a computer analysis is best. A program can also perform certain advanced analyses in addition, such as calculating an index of organization and looking for 'outlier' trials, without which the tree structure reveals significantly more structure.

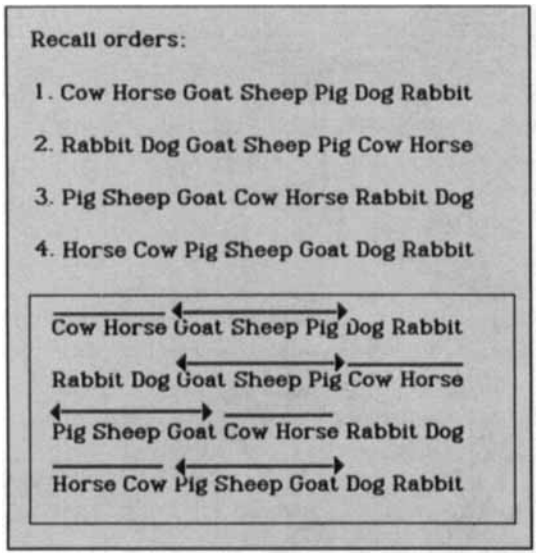

Recall orders:
1. Cow Horse Goat Sheep Pig Dog Rabbit
2. Rabbit Dog Goat Sheep Pig Cow Horse
3. Pig Sheep Goat Cow Horse Rabbit Dog
4. Horse Cow Pig Sheep Goat Dog Rabbit
Cow Horse Goat Sheep Pig Dog Rabbit
Rabbit Dog Goat Sheep Pig Cow Horse
Pig Sheep Goat Cow Horse Rabbit Dog
Horse Cow Pig Sheep Goat Dog Rabbit

Figure 15. *Four animal recall trials with low-level chunks marked*

This technique has been used in a variety of studies of expert-novice differences. In [6], for

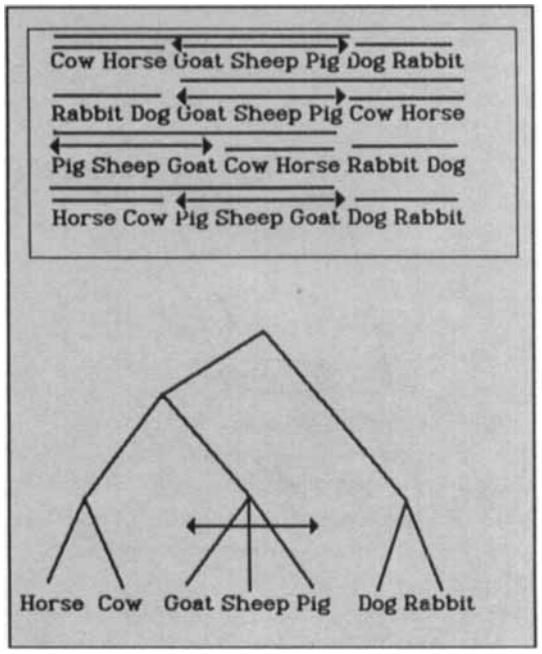

Cow Horse Goat Sheep Pig Dog Rabbit
Rabbit Dog Goat Sheep Pig Cow Horse
Pig Sheep Goat Cow Horse Rabbit Dog
Horse Cow Pig Sheep Goat Dog Rabbit
Horse Cow Goat Sheep Pig Dog Rabbit

Figure 16. *Higher-order animal recall chunks and final order tree*

example, novice, intermediate, and expert Algol-W programmers were asked to recall Algol keywords many times from many starting points while their performance orders were recorded. Experts differed remarkably from the novices. Experts showed much more organization, and the similarity among the expert structures was far greater than that among the novices. In [21], furthermore, the pauses between recalls of successive items was accounted for by the number of chunk boundaries crossed in the inferred memory organization. There has been a variety of studies that have used this technique to reveal organization in different domains of expertise, all showing a convergence among experts in their organization of the concepts in memory.

Figures 17 and 18 show the ordered trees for one expert and one novice, respectively, in the study of Algol knowledge. The expert clearly understands the function of these special words, grouping the words concerning loops (WHILE-DO, FOR-STEP), the logic items (AND-OR, TRUE-FALSE), and the string representation items (STRING-BITS, LONG-SHORT, REAL). The novice, on the other hand, grouped the short words (AND-OR-OF-FOR), grouped the conditional words (THEN-WHILE, ELSE-IF) in an order not standard to programming, and clustered words into a small scenario connected with do with "long and short bits of string" (BITS-STRING-LONG-SHORT). Close examination of the resulting ordered trees can reveal aspects of what the expert 'sees' in a situa-

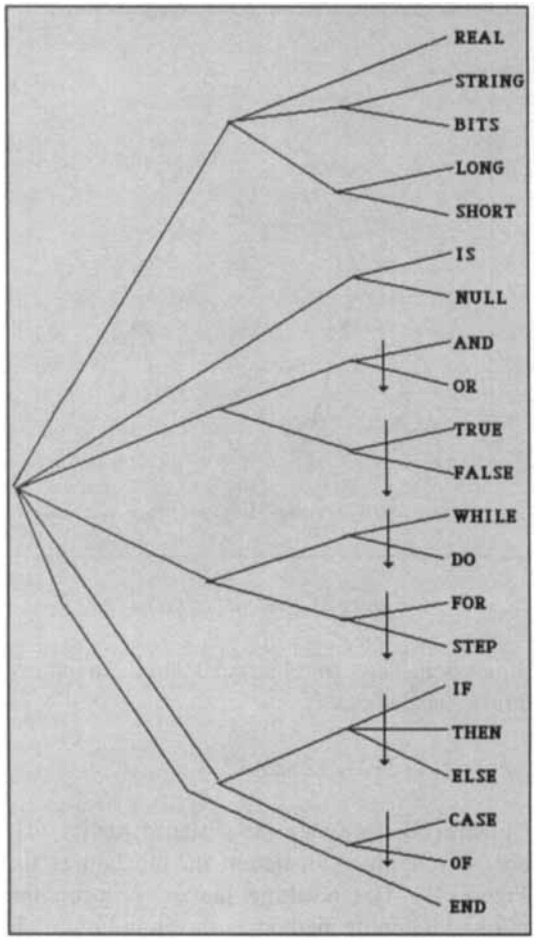


Figure 17. *Ordered tree for expert Algol programmer* [6]

tion in his/her domain of expertise.

2.2.5 Repertory grid analysis. This technique, the last to be presented in this review, is the most complete. It includes an initial dialog with the expert, a rating session, and analyses that both cluster the objects *and* the dimensions on which the items were rated. Essentially, it is a free-form recall and rating session in which the analyst makes inferences about the relationships among objects and the relatedness of the dimensions the expert pays attention to.

Repertory grid analysis is a technique whose origins are in personal construct theory in clinical psychology [22]. Used in the clinical setting, it was intended to reveal to the patients the kind of attributes they normally or abnormally attend to in their emotional lives. Boose [23] and [24] has adapted it for the explicit development of rules in expert systems.

The initial session begins with an open interview of the expert, asking him/her to name some objects in the domain of expertise. After a small set is generated, the analyst picks three of these objects and asks, "What trait distinguishes any two of these objects from the third?" The expert names a dimension in whatever vernacular is natural. He/she then indicates which are 'high' on this trait, and which are 'low'. The analyst records the dimension and assigns a scale value (e.g. 1–3) to the three objects. The analyst then picks three other objects and asks the same question about what trait distinguishes two of these from the third. This process of asking for salient dimensions continues with a significant number of triples, enough so that the analyst is satisfied that he/she has uncovered the major dimensions of similarity and dissimilarity.

Having collected a 'grid' with objects at the top and dimensions across the left border, the analyst then asks the expert to fill in all the missing values. That is, all objects are then to be rated on all dimensions. Figure 19 is an example grid that rates quality of students on nine different elicited dimensions, where each student is rated on a three point scale for each of the dimensions that

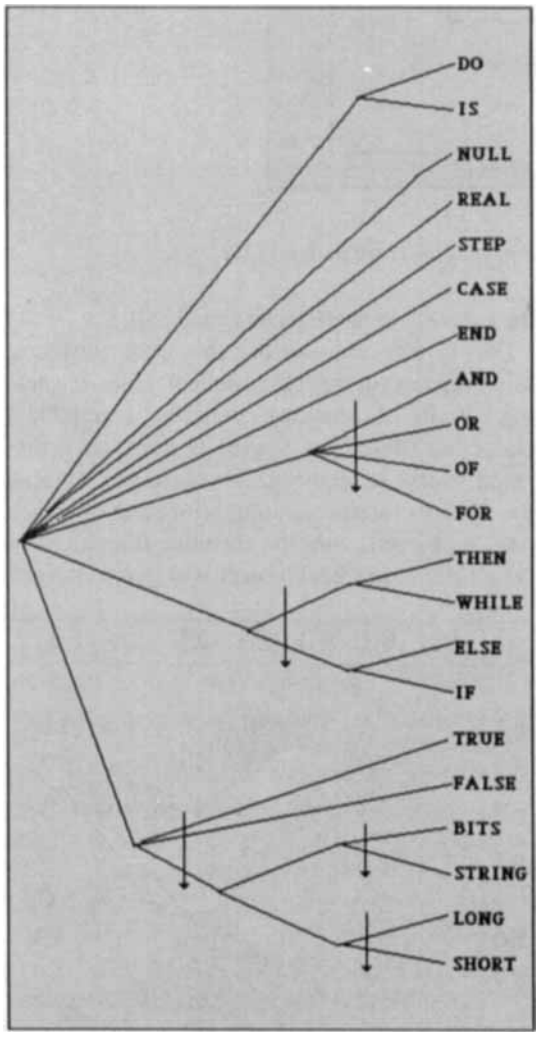
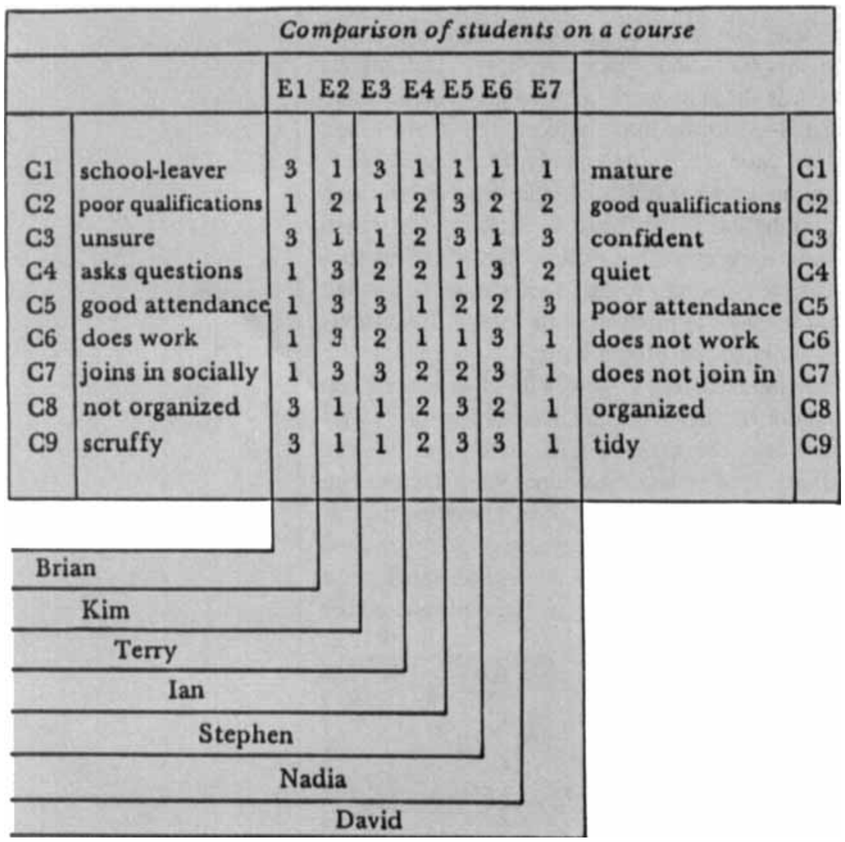


Figure 18. *Ordered tree for novice Algol programmer [6]*

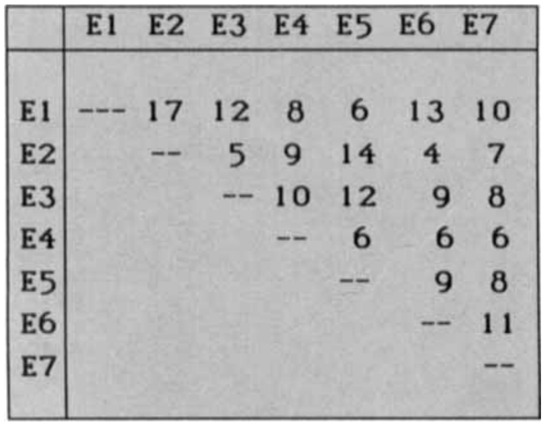


Comparison of students on a course										
		E1	E2	E3	E4	E5	E6	E7		
C1	school-leaver	3	1	3	1	1	1	1	mature	C1
C2	poor qualifications	1	2	1	2	3	2	2	good qualifications	C2
C3	unsure	3	1	1	2	3	1	3	confident	C3
C4	asks questions	1	3	2	2	1	3	2	quiet	C4
C5	good attendance	1	3	3	1	2	2	3	poor attendance	C5
C6	does work	1	3	2	1	1	3	1	does not work	C6
C7	joins in socially	1	3	3	2	2	3	1	does not join in	C7
C8	not organized	3	1	1	2	3	2	1	organized	C8
C9	scruffy	3	1	1	2	3	3	1	tidy	C9
		Brian	Kim	Terry	Ian	Stephen	Nadia	David		

Figure 19. *Rating grid for seven students [25]*

the expert generated (taken from [25]).

Two further analyses use this grid: clustering of its objects (in this case, students), and clustering of its dimensions. Johnson hierarchical clustering (described above) is used, so a distance matrix is required. For clustering objects, the distance metric is straightforward: for each pair of objects, count the absolute difference between the scores each object was given on each



	E1	E2	E3	E4	E5	E6	E7
E1	---	17	12	8	6	13	10
E2		--	5	9	14	4	7
E3			--	10	12	9	8
E4				--	6	6	6
E5					--	9	8
E6						--	11
E7							--

Figure 20. *Between-object (student) distance matrix*

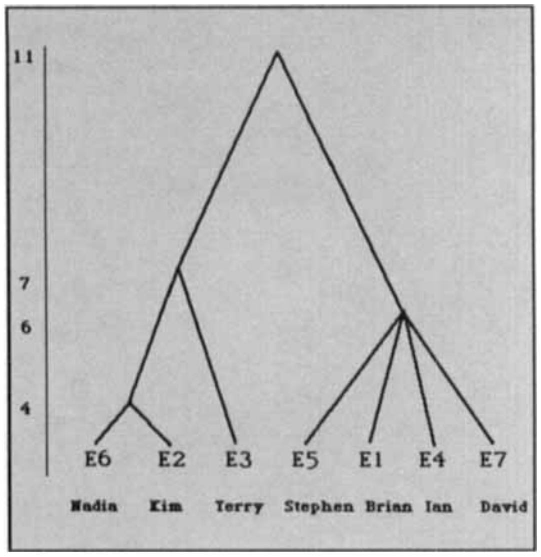


Figure 21. *Student hierarchy*

dimension. Thus, for objects E1 and E2 in our example, the distance is:

2+1+2+2+2+2+2+2+2 = 17

Figure 20 illustrates the distance matrix that arises from this calculation on the ratings for Figure 19. The resulting hierarchy, using the minimum joining method, is shown in Figure 21. This display should be given to the expert for further comment and analysis.

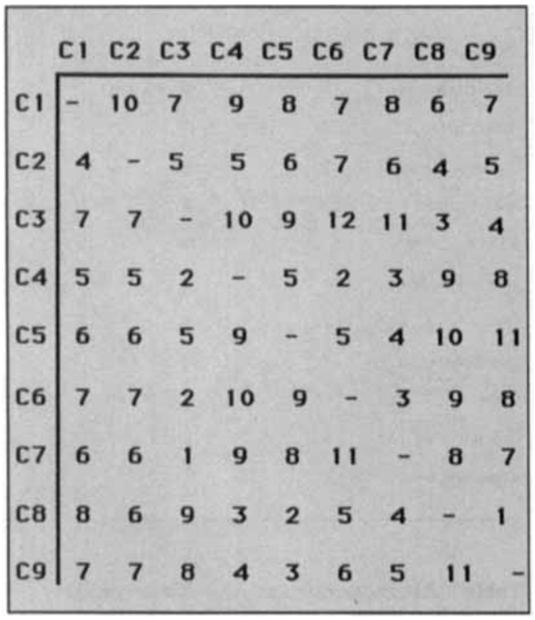
The second analysis done on the original grid examines the similarity of the dimensions. Here the definition of distances is not so straightforward. Since no judgment was ever made as to which end of a scale received a '3' and which a '1', there may be cases where very similar dimensions are highly correlated but were assigned opposite scale values. Therefore, part of the calculation of the distance matrix involves 'flipping' appropriate dimensions. This is done in the following manner: first, the full distance matrix is calculated as the actual (not absolute) difference between the ratings of all objects on two dimensions. Above the diagonal are the distances between two dimensions the way they are written; below the diagonal, the second dimension (the 'row' dimension) is reverse in scale. That is, comparing C1 and C2 across all objects produces a score of:

(C1-C2) = 2+1+2+1+2+1+1 = 10

Next, the C2 values would be 'flipped', each scale value 3 translated as a 1, each 1 as a 3. Consequently, comparing C1 with C2' (flipped) would result in:

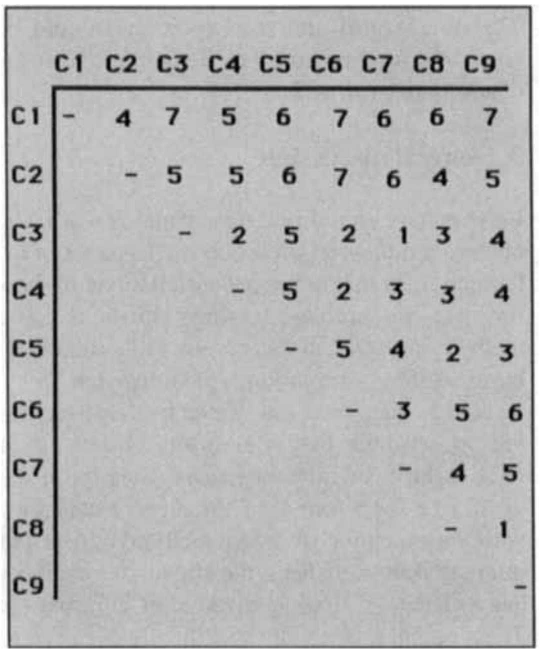
(C1-C2') = 0+1+0+1+0+1+1 = 4

Figure 22 is the full matrix of the distances among all pairs and 'flipped' pairs of dimensions from the grid in Figure 19. since Johnson hierarchical clustering cannot use assymetric distances, a symmetric half-matrix is needed. To translate this into a half-matrix, we select the highest of the two relevant cells, (C1-C2) and (C1-C2'). This resulting half-matrix is shown in Figure 23; the hierarchical clustering solution is shown in Figure 24.



	C1	C2	C3	C4	C5	C6	C7	C8	C9
C1	-	10	7	9	8	7	8	6	7
C2	4	-	5	5	6	7	6	4	5
C3	7	7	-	10	9	12	11	3	4
C4	5	5	2	-	5	2	3	9	8
C5	6	6	5	9	-	5	4	10	11
C6	7	7	2	10	9	-	3	9	8
C7	6	6	1	9	8	11	-	8	7
C8	8	6	9	3	2	5	4	-	1
C9	7	7	8	4	3	6	5	11	-

Figure 22. *Asymmetric between-concept (rating scale) distances*



	C1	C2	C3	C4	C5	C6	C7	C8	C9
C1	-	4	7	5	6	7	6	6	7
C2		-	5	5	6	7	6	4	5
C3			-	2	5	2	1	3	4
C4				-	5	2	3	3	4
C5					-	5	4	2	3
C6						-	3	5	6
C7							-	4	5
C8								-	1
C9									-

Figure 23. *Symmetric between-concept (rating scale) distances*

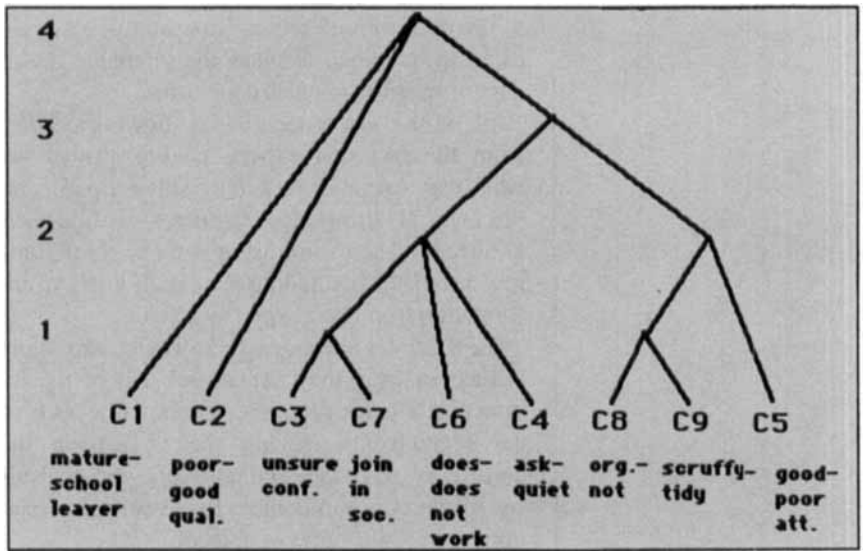


Figure 24. *Rating scale hierarchy*

Boose's expertise transfer system further examines these clustered dimensions to find those that are highly correlated, those that imply one another, and those that are super-ordinate to each other. Rules are obtained in a second interview with the expert, wherein the traits or dimensions elicited in the first phase are reviewed and named. The program then generates production rules. Some rules reflect the correlation between dimensions, some combine dimensional values to predict an end category.

Boose [24] reports that several hundred prototype systems have been created using this technique. It seems to be particularly applicable to classification type problems, where features of a new object (case) are observed and the object sorted into one of a known set of categories. The system has several advantages: it produces similarity matrices with a procedure that is much less tedious than directly rating the similarity of all pairs. Furthermore, it has been used to combine the expertise from two different experts in the same domain, and it has been used to combine two experts in different aspects of the same general domain. Individual repertory grids can also be used as a basis for discussing or negotiating disagreements among experts. In this situation, individual grids are generated by each expert; verbal discussion ensues from both experts viewing both grids and clustering solutions. Similarly, grids could be used as an aid in transfering expertise from an expert *tutor* to a novice.

The novice grid and the expert grid could be compared, and mismatches determine the focus of additional instruction.

3. General discussion

Experts have stored rich representations of facts, objects and their attributes, as well as a set of inference rules that connect constellations of facts for use in problem-solving situations. The methods collected in this review differentially illuminate these various kinds of knowledge.

Table 1 categorizes the twelve methods according to whether they illuminate objects, their relationships, or inference rules. Clearly, in the open, free-form format of the *direct* techniques, with the exception of drawing closed curves and inferential flow analysis, the knowledge engineer has a chance of finding *any* kind of information. These techniques are not specifically designed to elicit one particular kind of information over another. Drawing closed curves explicitly illuminates the relationships among objects in the problem space; inferential flow analysis, as indicated in its name, displays the inference chains experts may use to reach conclusions.

All of the *direct* techniques, however, suffer from the fact that experts cannot always say what they know or how they solve a particular problem. If knowledge engineers confine their knowledge acquisition techniques to these, they run the risk of excluding important kinds of information from their expert systems.

The *indirect* techniques, however, are more limited in what they can reveal. All of the indirect techniques illuminate particular aspects of the relationships among the objects in the domain of expertise. The repertory grid analysis can also produce inferences based on the correlations among attributes of objects.

The *indirect* techniques, however, involve assumptions about the underlying *form* of the representation of objects and their relations. Table 2 aligns the *indirect* techniques with each of the forms listed in the introduction: lists, tables, categorical hierarchies, inferential flows (decision trees), networks, physical space, and physical models. The two direct techniques, drawing closed curves and inferential flow analysis, are included here because they similarly make assumptions about the format of the underlying representation. For each technique, the primary assumed form is indicated with an 'X', and those forms that can additionally be revealed are marked with a '+'. An additional category of form, 'undefined OBJECT-ATTRIBUTE-VALUE', is included to account for the primary data elicited in repertory grid analysis.

All of these techniques add richness to the base of information about expertise that can be elicited from interviews. Care should be taken, however, with the amount of faith that is placed in the validity of the displays of knowledge each of these techniques produces. For example, protocol analysis, if applied to a task that is not normally verbalized, may distort the problem solving process and reveal what the expert thinks he/she *should* report rather than what he/she normally uses. The indirect techniques are more resistant to these distortions because they do not suggest to the expert/subject the 'right' way to respond.

Since indirect techniques are based on assumptions and underlying theories, they can be abused

Table 1. *Kinds of information methods can reveal*

	OBJECTS	RELATIONS	INFERENCE RULES
Interviews	X	X	X
Questionnaires	X	X	X
Observation	X	X	X
Protocol Analysis	X	X	X
Interruption	X	X	X
Closed Curves		X	
Inferential Flow			X
MDS		X	
Hierarchies		X	
GWN		X	
Ordered Tree		X	
Repertory Grid	X	X	X

Table 2. *Kinds of structures the methods can show*

	Lists	Tables	Hierarchies	Flow	Networks	Physical Space	Physical Model
MDS	+	+				X	
Hierarchy			X				
GWN	+	+	+		X		+
Ordered Tree	+		X				
Rep. Grid			X	+			
Closed Curves					X	+	
Infer. Flow				X	+		+

to the extent that their basic assumptions are not met by the data. Multidimensional scaling, for example, can be applied to situations for which there is no reason to assume an underlying n-dimensional space. Johnson hierarchical clustering is too often used to indicate the 'clusters' of points on a multidimensional scaling solution. This co-representation is a direct violation of the underlying assumptions; the data cannot simultaneously satisfy both sets of assumptions. General weighted networks assume an underlying network; ordered trees assume a tree structure in which some clusters have prescribed orders of recall imposed on them. Repertory grid analysis assumes that objects are stored with their values on bi-polar dimensions, and connected in networks of dimensional values.

Only one technique makes explicit assumptions about how the expert produces the data. Ordered trees assume that items are stored in nested clusters and that all items of a cluster are recalled before all items of any other cluster. The remainder of the techniques rely loosely on the ability of the subject/expert to make a single valued 'similarity' judgment from whatever the stored form is.

Just as a statistician makes judgments about the suitability of a data set to the assumptions of a proposed analysis, the knowledge engineer must make judgments of the suitability of a method for knowledge elicitation to the kinds of knowledge the expert is assumed to possess. There is a number of ways these techniques can be misapplied for scientific discovery of mental organizations. However, if used as *exploratory analyses* of the way specific experts may store and use knowledge, these techniques can bring a great deal of information to the knowledge engineer. Using these techniques, knowledge engineers can hope to uncover more of what experts know than is currently accessible through interviewing or introspection.

4. References

[1] P. Harmon and D. King, *Expert Systems: Artificial Intelligence in Business*, John Wiley and Sons, New York, 1985.

[2] D.A. Waterman, *A Guide to Expert Systems*, Addison- Wesley, Reading, Massachusetts, 1986.

[3] W.G. Chase and H.A. Simon, 'The mind's eye in chess', in W.G. Chase (ed.) *Visual Information Processing*, Academic Press, New York, 1973.

[4] A.D. deGroot, *Thought and choice in chess*, Mouton Co., Paris, 1965.

[5] J.S. Reitman, 'Skilled perception in Go: Deducing memory structures from inter-response times', *Cognitive Psychology*, **8**, 1976, pp. 336–356.

[6] K.B. McKeithen, J.S. Reitman, H.H.Rueter and S.C. Hirtle, 'Knowledge organization and skill differences in computer programmers', *Cognitive Psychology*, **13**, 1981, pp. 307–325.

[7] V.L. Patel and G.J. Groen, 'Knowledge based solution strategies in medical reasoning', *Cognitive Science*, **10**, 1, 1986.

[8] B. Hayes-Roth and F. Hayes-Roth, 'A cognitive model of planning', *Cognitive Science*, **3**, 1979, pp. 275–310.

[9] C.W. Holsapple and A.G. Whinston, *Manager's Guide to Expert Systems*, Dow-Jones-Irwin, New York, 1986.

[10] C.H. Coombs, R.M. Dawes and A. Tversky, *Mathematical Psychology: An Elementary Introduction*, Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

[11] D. Meister, *Behavioral Analysis and Measurement Methods*, John Wiley and Sons, New York, 1985.

[12] K.A. Ericcson and H.A. Simon, *Protocol analysis: Verbal reports as data*, MIT Press, Cambridge, Massachusetts, 1984.

[13] A. Newell and H.A. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[14] R. Schweickert, A.M. Burton, N.K.Taylor, E.N. Corlett, N.R. Shadbolt and A. Hedgecock, *A comparison of knowledge elicitation techniques for expert systems: A case study in lighting for industrial inspection*, Technical Report, Department of Psychology, Purdue University, 1986.

[15] W.J. Salter, 'Tacit theories of economics', *Proceedings of the 5th Annual Conference of the Cognitive Science Society*, Rochester, New York, 1983.

[16] R.N. Shepard, A.K. Romney and S.B. Nerlove, *Multidimensional Scaling: Theory and Applications in the Behavioral Sciences*, Volume I, Seminar Press, New York, 1972.

[17] E.W. Holman, 'The relation between hierarchical and Euclidean models for psychological distances', *Psychometrika*, **37**, 4, 1972, pp. 417–423.

[18] S.C. Johnson, 'Hierarchical clustering schemes', *Psychometrika*, **32**, 1967, pp. 241–254.

[19] R.W. Schvaneveldt, F.T. Durso and D.W. Dearholt, *Pathfinder: Scaling with network structures*, Memoranda in Computer and Cognitive Science, Report No. MCCS-85-9, Computer Research Laboratory, New Mexico State University, 1985.

[20] R.W. Schvaneveldt, M. Anderson, T.J. Breen, N.M. Cooke, T.E. Goldsmith, F.T. Durso, R.G. Tucker and J.C. DeMaio, *Structures of memory for critical flight information*, Air Force Human Resource Laboratory, Technical Report 81-46, 1982.

[21] J.S. Reitman and H.H. Rueter, 'Organization revealed by recall orders and confirmed by pauses', *Cognitive Psychology*, **12**, 4, 1980, pp. 554–581.

[22] G.A. Kelley, *The Psychology of Personal Constructs*, Norton, New York, 1955.

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

[24] J.H. Boose, *Expertise Transfer for Expert System Design*, Elsevier, New York, 1986.

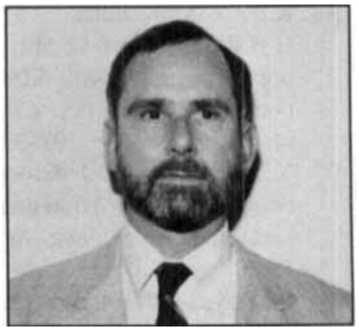
[25] A. Hart, *Knowledge Acquisition for Expert Systems*, McGraw-Hill, New York, 1986.

About the authors

Judith Reitman Olson

Judith Olson is an Associate Professor of Computer and Information Systems at the Business School at the University of Michigan, and an Adjunct Associate Professor in the Michigan Psychology Department. She is the Director of the Human-Computer Interaction Laboratory, a collection of researchers from computer science, computer and information systems, organizational behavior, industrial and operations engineering, technical communication, and psychology. Her work focuses on the design of interfaces for multi-media document preparation and mailing, the analysis of office work with the goal of suggestion computer-based support or automation, and knowledge acquisition for expert systems.

Henry H. Rueter



Dr Rueter is a Senior Systems Analyst at Vector Research, Inc. in Ann Arbor, Michigan, and a Visiting Scholar at the Human-Computer Interaction Laboratory at the University of Michigan. His current research interests include training and expert systems. One of the methods described in this paper, the Reitman-Rueter ordered tree analysis, was the subject of Dr Rueter's PhD dissertation at Michigan. Prior to his current position, Dr Rueter received the MS degree in mathematics and the MA degree in psychology from Georgia State University. His undergraduate degree was a BS in chemistry from the University of Georgia