# AUTOMATED KNOWLEDGE ACQUISITION

# FOR EXPERT SYSTEMS

## *AN OVERVIEW*

*Marie José Vlaanderen*

**1990**

ACKNOWLEDGMENTS

This paper is submitted as a partial requirement for a doctoral exam in philosophy and information sciences at the Faculty of Philosophy, Erasmus University Rotterdam, supervised by Prof.Dr.H.J.van den Herik, presently of the University of Limburg.
The Dr. Neher Laboratory in Leidschendam commissioned the research leading to this paper.

CONTENTS

# CHAPTER 1 INTRODUCTION

An expert system can be defined as a computer program which uses artificial intelligence techniques to perform or to guide a task that a human expert can do.
Feigenbaum (1977) shifts the emphasis from techniques and formalisms to the knowledge that an expert system contains. The expert knowledge seems to be a necessary and a nearly sufficient condition for developing an expert system (Hayes-Roth et al. 1983, p.7).
It seems to be clear that the acquisition of this knowledge as a technique (or is it still an art?) is an important field of study within the knowledge-engineering process.

Most authors of articles about knowledge acquisition quote Feigenbaum (1977) or Hayes-Roth et al. (1983) that knowledge acquisition is the bottleneck in the process of developing expert systems. They also complain that little has been published about it.

In the early years of expert-systems development - DENDRAL and HEARSAY-1 in the second half of the 1960s, MYCIN in the early 1970s - the knowledge engineer and the domain expert worked closely together over a long period of time. Sometimes the knowledge engineer possessed the relevant expertise himself[1] and could perform the role of domain expert as well. Working on an expert system usually took place in an academic setting where priority was given to the overall methodology of development rather than to efficiency and speed of production.
The more practical question of how to acquire all the relevant knowledge for a well-performing expert system came up in the 1980s. When construction of expert systems for commercial purposes began, the knowledge-acquisition process had to be efficient. The domain expert had to be available for interviewing and to be willing to go deeply into his field of expertise. At the same time another aspect in introducing expert systems became important. In the academic and research environment the domain expert was part of the team that was building the expert system. He was acknowledged in the research paper or article. After his participation he went back to his own work again. But when knowledge engineers started building expert systems within a company or other organization with the explicit intention to replace (at least partly) the domain expert with a computer program, the situation became different. The threat of losing jobs or losing prestige became apparent. Cooperation and, even more, enthusiasm of the domain expert could not be expected. In Welbank (1983) these problems are mentioned in detail.

It has become common knowledge that when the domain expert can directly communicate with a system, he is less reluctant to provide his expertise. One reason can be that he has a stronger, even more, physical, relation with the system-to-be.
The feedback of the knowledge-acquisition part of the system can give the impression that the system is understanding the input of the domain expert.[2] He can have the feeling that he is actually building the system and so accomplishing an important task (which is true for his part of the job).

[1]The use of "he" and "him" includes "she" and "her".

[2]Not unlike the ELIZA-effect: instead of being a dumb machine, the computer turns out to be a sympathetic conversation partner to whom it is easy talking. (Weizenbaum 1966)

This is in contrast with manual knowledge-acquisition methods. The most common method is interviewing by the knowledge engineer. The domain expert may have the impression that he is giving away his expertise without knowing how well this intermediate knowledge engineer is able to build an expert system that can stand up to his expectations. Yet worse may be the method of protocol analysis where the distance between expert and the finished expert system seems even larger.

An advantage of automated knowledge acquisition - especially in combination with other knowledge-engineering tools - in avoiding reluctance on the part of the domain expert is to construct its prototype at a very early stage of the building of the expert system. Many authors (e.g., Nii, in Feigenbaum and McCorduck 1983) mention that the expert should remain interested in the project. Seeing a prototype of the system within a short period of time will sustain his interest and can easily evoke improvements and extensions from him.
Another advantage of automated knowledge acquisition is that the process of elicitation can be structured in advance in order to build up a better structured knowledge base. Protocols, used in manual knowledge acquisition, on the other hand, are mostly unstructured. Less unstructured are interviews, where the knowledge engineer can follow certain guidelines. But even then he can easily be distracted by details that the expert wants to mention.

So far we may conclude that the field of automated knowledge acquisition is worth extensive exploration.

The complaint about the lack of literature becomes less severe. As publications about artificial intelligence per se tended to shift to expert systems, current articles about expert systems tend to be about special applications and about tools for building expert systems. In fact, during the last three years the amount of articles on (automated) knowledge acquisition has avalanched. Several issues of the International Journal of Man-Machine Studies are dedicated to papers presented at the Knowledge Acquisition Conferences in Banff, Canada. Also in Europe an annual conference on knowledge acquisition takes place. Several books (Kidd 1987, Hart 1986, and others) devoted exclusively to knowledge acquisition appeared during the last few years. The most recent publications are "The Knowledge Acquisition for Knowledge-Based Systems Newsletter" and "Knowledge Acquisition; An International Journal of Knowledge Acquisition for Knowledge-Based Systems". Both are edited by John Boose and Brian Gaines. A book series by Academic Press consisting of 2 volumes so far has also been published. (Gaines and Boose 1988, and Boose and Gaines 1988). These books contain reprints of articles about knowledge acquisition from the International Journal of Man-Machine Studies. A book that only appeared after the main part of this paper was written (Marcus 1988) gives an overview of six automated knowledge-acquisition systems. The last issue of the SIGART Newsletter (108, April 1989) contains an extensive overview of knowledge-acquisition systems and a bibliography with more than 400 entries.

Hayes-Roth et al. (1983) is one of the earliest books that devotes an entier chapter to knowledge acquisition. The Handbook of Artificial Intelligence (Barr, Cohen, and Feigenbaum 1981) contains relatively little on knowledge acquisition.
Notwithstanding the increasing amount of new literature about knowledge acquisition, real breakthroughs in method and technique are still not covered. Knowledge acquisition seems to remain a bottleneck in the knowledge-engineering process.

In this paper I propose to examine the problems of knowledge acquisition and more particularly the possibilities of automated knowledge-acquisition systems.

In the second chapter the problems of knowledge acquisition and expert-system building are cataloged.

The third chapter gives an overview of automated knowledge-acquisition systems that are already operational or still in development, or probably even aborted.

The fourth chapter evaluates the outcome of the previous chapters.

The study of literature for this paper was completed in April 1989.

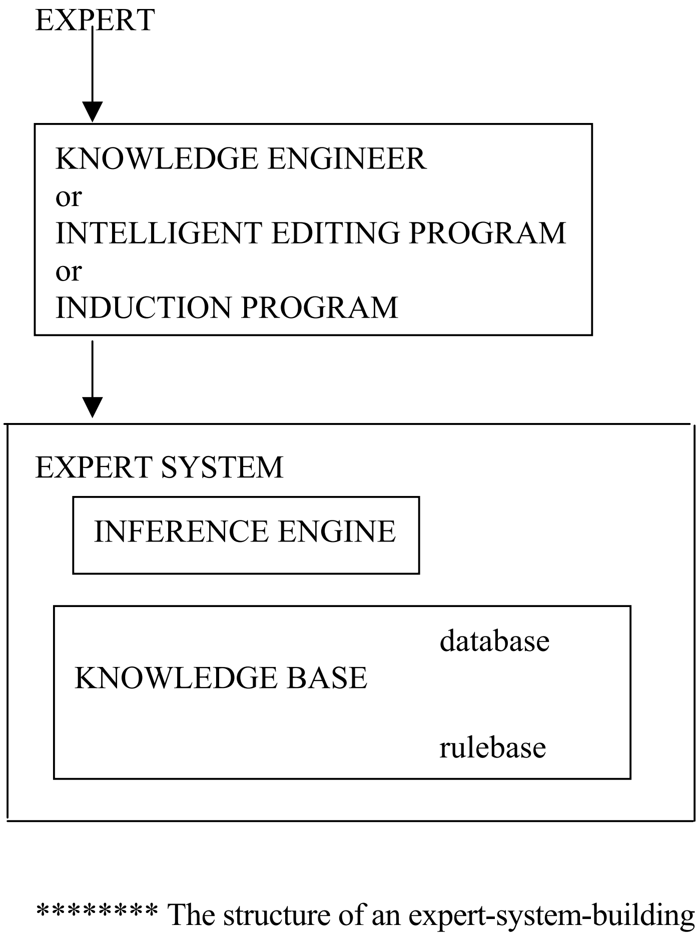
# CHAPTER 2 KNOWLEDGE ACQUISITION AND THE BUILDING OF EXPERT SYSTEMS

## *2.1 INTRODUCTION*

Reminded by Feigenbaum (1977) that the power of an expert system is the knowledge it possesses, most of the emphasis in expert- system developing should go to the knowledge-acquisition part of it.

In the classic book on expert-system building, Hayes-Roth et al. 1983, the building process is described as the process of knowledge acquisition (Buchanan et al. 1983). Throughout this section I follow their guidelines.

*******************************************************************************



******** The structure of an expert-system-building process ******************
figure 2.1

The basic rule is that knowledge acquisition and expert-system building is an interactive process. The two should not be separated.
Some other basic rules for the expert-system-building process are also important:

1. Get acquainted with the domain, using textbooks or an initial interview with the domain expert, and
2. Start building a prototype of the expert system as soon as possible.

The work scheme should be as follows:
1. Problem outline. The tasks and subtasks of the expert system must be clear.
2. Domain concepts. The major concepts of the domain must be acquired and arranged in a semantic way.
3. Strategy. Another kind of knowledge are the rules the domain expert uses. Start with the main strategies.
4. Formalization. The acquired knowledge should be arranged in a certain kind of representation. It is important to know at an early stage how this representation looks like.
5. At this point the conditions are fulfilled to construct a prototype.
6. Then the knowledge engineer can continue to deepen and refine the knowledge acquisition and test the prototype. This can be seen as a cyclical process of conceptualization, formalization, implementation and testing.

For the sake of clarity something must be said about the terminology. Throughout this paper I use the term "knowledge acquisition" to mean the acquisition of knowledge for a special purpose, e.g., the expert's answer to a certain question. "Knowledge elicitation" is the unstructured revelation of knowledge without a presupposed plan, e.g., thinking aloud protocols or textbook knowledge.
A different meaning of knowledge acquisition was found in Avignon 1987 where articles on information retrieval were arranged under the heading of knowledge acquisition. It should be clear that "knowledge retrieval" is normally used for getting knowledge from a knowledge base, that is to say knowledge acquisition from the other end of the system.

## The identification stage

In the identification stage the basic aspects of the problem are characterized.
The domain expert, the informant, is chosen. Although the knowledge engineer is the one who directs the knowledge-acquisition process, the relationship between knowledge engineer and domain expert is one of student-teacher. The problem and its supporting knowledge structure must be characterized. This can be divided into the following questions:
- What class of problems will the expert system be expected to solve?
- How can these problems be characterized or defined?
- What are important subproblems and partitioning of tasks?
- What are the data?
- What are important terms and their interrelations?
- What does a solution look like and what concepts are used in it?
- What aspects of human expertise are essential in solving these problems?
- What is the nature and extent of "relevant knowledge" that underlies the human solutions?
- What situations are likely to impede solutions?
- How will these impediments affect an expert system?

In this stage it must also be clear what resources the knowledge engineer can use. Resources are knowledge sources, time, computing (hardware and software) and financial resources. And finally the goal and use of the expert system should be specified.

## The conceptualization stage

Buchanan et al. (1983) lists several questions that should be covered.
The following questions need to be answered before proceeding with the conceptualization process:

- What types of data are available?
- What is given and what is inferred?
- Do the subtasks have names?
- Do the strategies have names?
- Are there identifiable partial hypotheses that are commonly used? What are they?
- How are the objects in the domain related?
- Can you diagram a hierarchy and label causal relations, set inclusion, part-whole relations,etc? What does it look like?
- What processes are involved in problem solution?
- What are the constraints on these processes?
- What is the information flow?
- Can you identify and separate the knowledge needed for solving a problem from the knowledge used to justify a solution?

The key concepts and their relations must be made explicit. In this stage the basis of a framework of the final system should be made. The problem and subproblems need to be sufficiently analyzed before proceeding to the next stage.

### The formalization stage

In the formalization stage concepts, rules, etc. are mapped in a formal framework. Three aspects are important.
First the hypothesis space. After the concepts are formalized one has to determine how they link to form hypotheses.
The second aspect is the uncovering of an underlying model of the process used to generate solutions in the domain.
And third, the understanding of the characteristics of the data that helps to understand the structure of the problem space.

By following these stages the knowledge engineer has specified the contents of the data structures, the inference rules and the control strategies.

### The implementation stage

The formalized knowledge is mapped into the representational framework. This framework specifies the form of the data gathered in the previous stage. At this point a prototypical system can be built.

### The testing, refining, and debugging stage

The prototype system can be evaluated, and when it is proven to be inadequate, new tools can be chosen.
Further testing can be done by a number of different and challenging examples to find weak spots in

the knowledge base and the inference structure. Incompleteness and inconsistency will appear in this stage. This means that the knowledge base needs refining. Reasoning errors mean faults in the inference rules.

Throughout all the stages it is necessary to gather knowledge from the expert. Knowledge acquisition is a continuous process during the building of an expert system, although the purposes are different (initial knowledge, refinement, etc.).
We should be aware that automated knowledge-acquisition systems - the issue of this paper - are more than "interview machines". They should be able to cover a range of various techniques and understand their different roles in the expert system building process.

## *2.2 THE MISMATCH PROBLEM*

The methods of knowledge acquisition and the building of expert systems seem to be straightforward and easy to apply. But in fact many problems arise while working with knowledge acquisition. Often we read about knowledge acquisition as the bottleneck of expert system building.

What is that bottleneck called knowledge acquisition?
The metaphor is well-chosen. We can regard human knowledge as a tangled mass of information stored in a not too wide necked bottle. The expert system is a container with compartments of various sorts. In the knowledge acquisition process the knowledge engineer must try to get hold of the human knowledge and to put it in the appropriate modes at the appropriate places in the expert system.
With this picture in mind we can see the problems of the knowledge-acquisition process more clearly.

### The knowledge mismatch

The most fundamental difficulty seems to be the mismatch between human and machine knowledge. First of all there exists a mismatch between the organization of knowledge. Human knowledge is stored in a yet unknown way. The most commonly used hypothesis is the idea of mental models (see section 2.12). Using knowledge is for a human not a conscious process of retrieval and applying of facts and rules, but an unconscious involvement with objects and state of affairs (or their descriptions). The analogy between a mental model and the real-world state of affairs gives the solution of what kind of knowledge should be applied. In this process of "applying" knowledge there is no need to verbalize the knowledge. When talking about an expert who "knows" about his field, we are talking about something vague. Knowledge is only shown in the behavior of this expert, but the logic or justification behind his actions is not shown. The applied rules remain unrevealed too. What we want to have is a sort of materializing of knowledge: a representation of knowledge. Only in a representation knowledge is transferable, can be understood, can be manipulated, etc. Verbal representation is a good solution. As a matter of fact it is the most commonly used kind of representation in human communication. But there seems to be an obstruction when it comes to verbalize knowledge in all its details.

**The representation mismatch**

There is seldom a need for an expert to verbalize his knowledge to non-experts. In the exchange of knowledge between experts (and even between skilled and less-skilled experts) a great deal of the information is presupposed. That is the common-sense knowledge about the world but also the general knowledge about the specific domain, such as vocabulary, concepts, general rules. Even in the situation of knowledge exchange in teaching a great part of knowledge is already presupposed.

In the knowledge-acquisition process - when expert knowledge has to be transferred into a system that does not contain any knowledge - the domain expert encounters great difficulty when he has to verbalize his knowledge.
The expert system, on the other hand, uses also a system of knowledge representation. Most commonly used are representations by frames for declarative knowledge and production rules for procedural knowledge. Essential is the formal mode of these forms of representation.

So far I have mentioned human knowledge represented by mental models, by behavior, by verbalizing (in a natural language that is difficult to formalize). At the expert-system side knowledge is represented, among others, by frames and production rules written in programming languages. This incompatibility between man and machine can be called the representational mismatch.

**The completeness mismatch**

Tacit knowledge, the knowledge that cannot be represented at all, causes another difficulty. The human expert is not aware of all the knowledge he possesses. Only when he is explicitly asked about it, or when he is engaged in a particular section of his expertise, can he verbalize this knowledge.
The expert system needs all the knowledge involved to solve the problems for which it will be built. This gap between the expert and the system is the completeness mismatch.

**The category mismatch**

An obvious, but less mentioned difference is the disparity between human and machine as two totally different systems. We are not talking about aspects such as difference in representation, but the integrated system of a man as problem solver versus the expert system as a problem solver. The overall approach is so different that comparing aspects seems fruitless. Even the above mentioned differences in representation are incongruent: vague, not verbalized versus formalized; incomplete versus complete. These differences are more profound than a difference, e.g., among programming languages, between frames and scripts, or between the knowledge of expert A and expert B in a particular field.
The system of human information processing is of a different category than machine information processing. We can call this the category mismatch.

These four mismatches return at the other end of the expert system: the interface with the end-user. It is less of an obstacle for human beings to understand the output of a computer system. Still some difficulties in understanding remain. The better designed expert systems have explanation facilities to assure the user of its reasoning. Although this is not so much a knowledge-acquisition problem as a problem of expert system designing, it is worth mentioning.

So far we have given the knowledge-acquisition bottleneck a name: the mismatch problem.
From this brief analysis of human versus machine knowledge, we can draw two conclusions.
First, during the knowledge acquisition process we should constantly be aware of the mismatch in approach between man and machine. Second, knowledge acquisition is not a part of expert-system building, but an integrated aspect of the whole knowledge engineering/expert-system building enterprise.

In the following sections several problems in knowledge acquisition are reviewed.

## *2.3 THE STATE OF THE ART*

Jackson (1986) says that knowledge engineering is still an art and not a science. The knowledge engineer is therefore still handicapped by the lack of guidelines for a maximum use of the available technology. Knowledge acquisition is an empirical exercise, guided by trial and error and common sense. While handling formal systems is not an uncommon field, handling human knowledge and formal systems together is still a difficult task - despite centuries of exploration in the theory of knowledge.
A large amount of articles on knowledge acquisition diminish this lack of guidelines for the knowledge engineer. I think that during the last few years, since Jackson wrote his book, knowledge engineering became more a skill than an art. Probably in the next couple of years we can speak of knowledge engineering as a science.
Since this paper is about automated knowledge-acquisition I shall focus my attention on the problems of knowledge acquisition. If we can analyze these problems, maybe we can give solutions, or rather, directions to solutions. We might be able to give better suggestions for automated knowledge-acquisition.

From the preceding section we can make a list of features that knowledge acquisition should encompass:
- elicit knowledge from multiple sources
- recognize structures to decompose and to build up in a knowledge base; the analysis and conceptualization process
- process declarative knowledge (facts) and procedural knowledge (rules)
- validate the knowledge base for consistency
- handle uncertain knowledge
- find omissions, expand, and refine the knowledge base
- make consistent changes throughout a large knowledge base
- give guidance to the entire process

Some of these features need metalevel knowledge. The knowledge engineer is directing the knowledge-acquisition process and possesses this metalevel knowledge. He is aware what he is doing. In an automated knowledge-acquisition system this ability must be built into the system.
We should also be aware that expert systems are built to solve various problems. One can distinguish analytical problems (such as classification and diagnosis), synthetical problems (configu-ration, design and planning), encyclopedic problems (facts finding) and combinations of these

problems (control, monitoring, prediction and repair). The knowledge for these expert systems differ from each other in structure. The techniques to elicit these various kinds of knowledge are therefore also different.

In the initial stage of expert-system building the structure of the domain should be clear. On the other hand the structure can only be revealed after one gets started with an initial knowledge acquisition session. The knowledge engineer must decide somewhere between planning and getting started how he tackles this problem.
Planning means looking for structures. They may be fixed by the nature of the domain, but also be embodied by the outlines of an expert-system shell that will be used. Further it is important what kind of representation will be used. On the one side it is important to choose a representation that corresponds to the knowledge that the domain expert reveals; on the other side the representation should be chosen in such way that the expert system works efficiently.

At a certain point in the expert-system-building process the knowledge engineer will encounter problems with incomplete knowledge. Special techniques are used to find omissions in the knowledge base and to elicit them from the domain expert. Another problem is inconsistencies in the knowledge base. Specially in this case automated systems can trace these faults and repair them.

It seems that a whole new branch of knowledge-acquisition methods has been developed, different from the interviewing approach. It is called machine learning or machine induction. Reactions vary to some extent. Some positive aspects are certainly present, but machine learning will probably never be the single method in a knowledge-acquisition process.
Throughout the knowledge-acquisition process the knowledge engineer encounters the typical characteristics of human knowledge. In the last couple of years more research has been done in cognitive science. The results will help understanding human information processing.

In the next sections I shall discuss these problems in more depth.

## 2.4 TRANSFER OF KNOWLEDGE

### 2.4.1 Human expert thinking

To get a better understanding of the problems of knowledge transfer it is necessary to know more about the way an expert deals with his expertise. From cognitive psychology we know several facts about cognitive processes.

The interaction between a knowledge engineer (and his tools) and the domain expert (and other sources of knowledge) is a delicate one. The knowledge needed for the expert system is more or less outlined by the knowledge engineer in broad terms. Let us assume that he knows he needs an amount of facts about the domain and an amount of rules used by the expert. He also has an idea of the inference mechanism of the system and the kind of representation he will probably use. The performance of experts depends on large quantities of domain knowledge. On the one hand, they are able to reflect on their own cognitive processes (meta-cognition) and on their own state of domain knowledge (meta-knowledge). But on the other hand, the domain expert possesses an amount of facts and rules of which he is usually not aware and thus, he does not know precisely what is needed

for the expert system.
The expert cognition lacks also computational and representational power.
Since there is no coherent theory yet, most cognitive psychologists adopt a common approach based on an information processing view of human cognition. Cognitive processes can be analyzed into sequences of ordered stages. This entails identifying the sequence of mental operations through which information flows (and is transformed) in the performance of a particular cognitive task. Expert thinking is studied within this framework.[3]
People's awareness of their own mental processes is rather limited. The proceduralization of knowledge and automatization of cognitive skills that accompany the development of expertise, serve to make expert thinking even less accessible to introspection.
The cognitive correlates of expertise, whether beneficial or otherwise, are essentially domain-specific in effect. Thus outside his specialist area any cognitive advantage the expert may have enjoyed inside the domain quickly disappears. Another underlying theme in the development of expertise is a greater reliance on pattern recognition and memory (stored knowledge) at the expense of deductive reasoning. At another level, though, experts often show an impressive ability to reflect on, and flexibly control, their high-level task strategies. But precisely how automated skills and control strategies combine in expert problem solving remains poorly understood (Slatter 1987, p 41).

## 2.4.2 Techniques for knowledge acquisition

Slatter (1987) mentions six techniques for knowledge acquisition:

1. Interviews - structured according to a plan or unstructured.
   Advantages: - explicit knowledge is easy to obtain.
   Disadvantages: - not for detailed or implicit and tacit knowledge.

2. Verbal protocols - thinking aloud while doing a task.
   Advantages: - reveals knowledge that is difficult to verbalize.
   Disadvantages: - analysis of protocols is difficult.

3. Machine induction - rules are automatically induced from given examples.
   Advantages: - no knowledge engineer necessary.
   Disadvantages: - needs a database with cases.
   - can be instable.
   - rules are often complex.

4. Observational studies - observation of an expert doing his task.
   Advantages: - avoids preconceived ideas.
   Disadvantages: - time consuming for the knowledge engineer
   - cooperation from the domain expert can be difficult.
   Further advantages and disadvantages are the same as with verbal protocols.

5. Conceptual sorting - also called card sorting, a cognitive psychology technique. Concepts are gathered and the expert is asked to sort the concepts (written on cards) according to resem-

[3]See Welbank (1983) for reasons why experts usually do not know their expertise in an organized way.

blances, hierarchies, etc.
Advantages: - easy for organization of a lot of information.
- gives global structure of domain.
Disadvantages: - requires special skill of the knowledge engineer.
- can produce artificial structures.

6. Multidimensional scaling - often used as repertory grid method, where similarities and differences of sets of concepts are identified.
Advantages: - can elicit well subtle or non verbal distinctions.
Disadvantages: - time consuming for the domain expert, specially when larger numbers of concepts are involved.
- expertise needed to understand the technique.

In the next sections I go deeper into the problems the knowledge engineer encounters when he is trying to apply these techniques.
Machine induction, or machine learning, is a typical technique in automated knowledge acquisition. It will be discussed later.

### 2.4.3 Interviews

Gaines (1987) gives from a psychologist's point of view an overview of the problems that occur when interviewing a domain expert.
Psychologists know that knowledge transfer among experts raises problems. Not only the expert cannot express his knowledge but also he is not aware of its significance.
There seems to be no necessary correlation between verbal data and mental behavior. So verbal reporting might be useless.
The main problems identified in accessing an expert's knowledge are:
- Expertise may be fortuitous. Results obtained may be dependent on features of the situation which the expert is not controlling.
- Expertise may not be available to awareness. An expert may not be able to transmit the expertise by criticizing the performance of others because he is not able to evaluate it.
- Expertise may not be expressible in language. An expert may not be able to transmit the expertise explicitly because he is unable to express it.
- Expertise may not be understandable when expressed in language. An apprentice may not be able to understand the language in which the expertise is expressed.
- Expertise may not be applicable even when expressed in language. An apprentice may not be able to convert verbal comprehension of the basis of a skill into skilled performance.
- Expertise expressed may be irrelevant. Much of what is learnt, particularly under random reinforcement schedules, is superstitious behavior that neither contributes nor distracts from performance.
- Expertise may be incomplete. There will usually be implicit situational dependencies that make explicit expertise inadequate for performance.
- Expertise expressed may be incorrect. Experts may make explicit statements which do not correspond to their actual behavior and lead to incorrect performance (Gaines 1987).

In all cases of expert-system building some sort of interview on some level should take place.[4]
The essence of interviewing is that the expert answers questions posed by an interrogator. Various kinds of interviews are possible.
The first important step is the choice of expert. He must have the right capabilities, be available and give support.
In selecting the domain expert one should not initially expect complete coverage. The task should be decomposable and the domain fairly stable.
In getting background knowledge about the domain, the expert should make a sort of tutorial about the subject.
The basis of an initial knowledge base can be formed by references, written material, handbooks, and so on. Begin the actual knowledge elicitation by having the expert go through the task, explaining each step in detail (Prerau 1987).

For the purpose of getting a basic idea of the problem the knowledge engineer might perform the "20 questions" method, a fast way to reveal the essence of the problem. From here he can do a "grand tour" to catch the main features of the domain in a wide meshed network. From these interviews the knowledge engineer can structure the plan for the next interviews. These should reveal the concepts and attributes, their relations, and the rules. This can be done by giving the domain expert cases to solve. ("Name a particular problem. How would you solve it?") The interviewer can direct this process in a strict way, or he can give the expert room to associate with other problems, and so on.
The interviews with the domain expert should also include cross-checking questions ("What will happen if you don't do?") which can reveal more relevant facts (LaFrance 1987).
It seems that people are better in describing procedures than recalling facts. So the interviews about cases are giving better results than asking for facts without a problem to solve.
Experts are able to give knowledge in an explicit way when they are confronted with mistakes. The TEIRESIAS system uses this principle. Facts are usually better retrieved from handbooks. Although it is advised that an expert should check the textbook facts for accuracy. Sometimes books still use outdated methods or more theoretical ways than are used in daily practice.

Another form of interviewing is on-site observation where the knowledge engineer watches the expert solving problems on the job. This could be augmented by a thinking aloud protocol from the domain expert (see section 2.4.4). Although experts find it difficult to identify the most appropriate aspects of problem solving components. The reason might be that he is usually not required to give explicit formulation of his expertise. It also has been shown that people tend to stop verbalizing when the task is difficult. (see also De Groot 1965)
The problem of implicit knowledge is often not recognised by researchers. According to Berry (1987) there is a distinction between implicit knowledge which was once represented explicitly or declaratively and implicit knowledge which arises as a result of an implicit learning process and has never previously been explicitly represented. In the former kind of explicit knowledge declarative knowledge gets transformed into a procedural form. After a while the expert loses his ability to report it verbally. The second kind of implicit knowledge is the knowledge that is assembled by experience (perception etc.) and was never verbalized.

A widely used form of interview, often used in automated knowledge acquisition, is the repertory

[4]With the exceptional case that the expert himself builds the expert system or while using induction algorithms.

grid method (see section 3.6), or multidimensional scaling, used to classify and characterize a domain.
Advantages of scaling techniques as knowledge-elicitation tools are:
Scaling techniques can overcome the criticisms of interviewing and protocol analysis. Scaling techniques provide also information concerning the structure of knowledge that might serve as a basis for representing that knowledge in the system (Cooke and McDonald 1987).
A negative aspect of the rating-technique method is that more is revealed about the expert's rating toward a series of objects or attributes without regard for the context.

### 2.4.4 Protocol analysis

In addition to or connected with interviewing the knowledge engineer might perform task analysis and protocol analysis.
It is assumed that the knowledge engineer has sufficient knowledge about the domain to understand the protocol analysis. Protocols are often incomplete. Experts cannot verbalize as fast as they reason. They often omit things that seem obvious to them. Experts have no experience to think aloud and when they do, it might effect the task they are performing.
In addition to this Ericsson and Simon (1984) give several negative aspects of protocol analysis:
1. The subject has to change his thought processes if he is told to verbalize the information in a certain way.
2. The reporting process may alter the task performance.
3. The reports may yield a very incomplete record of the cognitive processes.
4. Only the information that is reportable is heeded during the performance of a task.
5. What cannot be reported are the cues that allow the subjects to recognize stimuli. Only the results of the process of recognition can be reported but not the intermediate steps.
6. Some reports are idiosyncratic and reflect the unique experiences of individuals.
7. Encoding of verbal protocols cannot be made objective and sound.

In summary, task analysis and protocol analysis do not seem to be an effective method in knowledge acquisition.

In general one can distinguish the different kinds of interviews. Formal (= structured) interviews about problem cases and protocol analysis can elicit procedural knowledge. Grid methods (= decom-position) and multi-dimensional analysis reveal declarative knowledge.
However a case study did not show that these techniques divided sharply declarative and procedural knowledge. In comparative studies these methods show some considerable shortcomings. First it was obvious that protocol analysis would not score very high, because the nature of the knowledge of the domain (in this case the identification of igneous rocks) is too declarative for this method.
Second, the techniques gave varying results depending on the experts involved. Especially the time factor varies according to the personality of the expert involved.
A positive finding might be that experts who were faced to put their knowledge in an unfamiliar format gave more useful results (Burton et al. 1988).

Studies such as the above are not often published even today. It should be worthwhile to have larger scale comparisons of interview techniques, also in different kind of domains. In the study of Burton et al. the domain was typically classification oriented. One can conclude that the elicitation technique should depend on the type of problem involved.

It is clear that the interaction that takes place between the knowledge engineer and the domain expert must be guided or structured. This is in contrast with the unstructured flow of information that takes place in several kinds of elicitation techniques.

### 2.4.5 Expertise from multiple experts

The knowledge engineer should be aware that expert knowledge is more than one kind and not all this knowledge can be acquired from one person.
Mittal and Dym (1985) suggest that interviewing multiple experts gives a broader result.
Hawkins (1983) gives an example from the field of petroleum geology. There are two schools of thought. Hawkins calls them the "classificatory" and the "behavioral" schools.
In the classificatory school the features of a particular described object are compared with the characteristics of well-known objects, called typical examples or prototypes. These prototypes are organized in a taxonomy.
The behavioral school generally presents its results in the form of calculated values of critical properties.
The knowledge base should be available for both ways of thought.
Neither school by itself proves to be adequate to describe real and complex, geological objects. Each school has problems in communicating with the other because their knowledge of real-world objects is structured so differently. This forming of schools is applicable to most sciences. It is therefore advisable to acquire knowledge from various experts to understand the different attitudes.

Another reason to take advice from multiple experts is the complementary of information. What one person lacks might be supplied by another. Winograd and Flores (1987) mention also the absence of formalized knowledge and the existence of prejudices.
Experts do not need to have formalized representations in order to act. They may at times manipulate representations as one part of successful activity. However, it is fruitless to search for a full formalization of the pre-understanding that underlies all thought and action.
Besides we never have a full explicit awareness of our prejudices. Even while working in technical and scientific fields we will sustain certain ideas that are formed on prejudices. By formal training prejudices will be formed ("schools") but also by incidents that make us believe something, rather than really know it. In these cases it is impossible to get a formalized representation of one's knowledge (Winograd and Flores 1987).

The knowledge engineer should be aware of these problems throughout the knowledge-acquisition process.

## *2.5 INTERACTION BETWEEN KNOWLEDGE ACQUISITION AND KNOWLEDGE REPRESENTATION*

Once knowledge is acquired, whether from experts or from other sources, it must be encoded in a knowledge-representation language. On the one hand, the knowledge representation affects the working of the system. On the other hand, the problem solving strategies of the system determine the knowledge representation. If we make the knowledge representation and our knowledge acquisition technique compatible then somehow the knowledge acquisition process should be easier.

Two conclusions can be drawn.

1. The knowledge acquisition process should fit into the knowledge representation
2. The knowledge representation should be adequate for the problem.

A panel meeting at the conference of the AAAI in 1986 discussed these issues. Unfortunately, the results were not published in the proceedings.
Moreover little has been published about the interaction of knowledge acquisition and knowledge representation.

Usually the systems are rule-based, so the knowledge should be procedural. Human experts have no problem with stating their knowledge in procedural form. Disadvantages of production rules are the lack of structure in the case of large numbers of rules. Further there is still need for other kinds of knowledge.
Shachter and Heckerman (1987) add to this that human experts find it easier to represent a rule in the form of IF <hypothesis> THEN <evidence> instead of IF <evidence> THEN <hypothesis>.
Knowledge bases based on conceptual structures rather than inference rules have difficulties handling procedural knowledge.

Suggestions have been made (Becker and Selman 1986) to extend the knowledge-representation system KRYPTON (see section 3.17) to facilitate the representing of expert knowledge.
KRYPTON is a synthesis of frame-structures and logic. The frame-based language is used for forming descriptive terms, the logic-based language makes the assertions. A knowledge representation based only on frames and networks lacks reasoning capability and can also be ambiguous. KRYPTON is made to be functional (Brachman et al. 1985).

Another approach to bringing knowledge acquisition and knowledge representation together is made by Bylander and Chandrasekaran (1987). They emphasize not only the importance of knowledge representation but in particular the knowledge-base reasoning. They have developed a theory of generic tasks:
Whether the knowledge is acquired by a knowledge engineer or by a program, ultimately the knowledge must be encoded in some knowledge representation. Consequently, knowledge acquisition cannot be separated from a broader theory of knowledge-based reasoning; a solution to knowledge acquisition must be compatible with a solution to the general problem of knowledge-based reasoning.
A theory of generic tasks identifies several types of reasoning that knowledge-based systems perform and provides an overall framework for the design and implementation of such systems.

The interaction problem is this:
Representing knowledge for the purpose of solving some problems is strongly affected by the nature of the problem and by the inference strategy to be applied to the knowledge. In other words, how knowledge is represented has a close relationship to how knowledge is used to solve problems; knowledge is dependent on its use.
The interaction problem has serious implications for how knowledge acquisition should be done. Also, if different kinds of reasoning have different kinds of interactions, there is a need for a different knowledge-acquisition methodology for each kind of reasoning.

1. Choice of knowledge. The knowledge-acquisition process must choose what knowledge to ask for and what knowledge to encode. The choice is driven by the need to gain leverage on the problem by

obtaining knowledge with high utility and to reduce complexity by avoiding or discarding knowledge with low utility. Not everything the domain expert knows has the same level of usefulness, and in any case, it is not feasible to acquire everything that the domain expert knows.

2. Constraints of inference strategy. A knowledge representation requires some process that, given a description of a situation, can use (or interpret) the knowledge to make conclusions. It is this process which is called the "inference strategy" (or "inference engine"). The knowledge must be represented so that the inference reaches appropriate conclusions in a timely fashion. Consequently, the knowledge must be adapted to the inference strategy to ensure that certain inferences are made from the knowledge and not others. Also, given a choice of inference strategies, there will be an interaction between the strategy chosen and the form of knowledge.
Generic tasks are basic combinations of knowledge structures and inference strategies that are powerful for dealing with certain kinds of problems. The generic tasks provide a vocabulary for describing problems, as well as for designing knowledge-based systems that perform them.

It seems very important to elaborate more on this subject to facilitate the knowledge-acquisition process.

## *2.6 STRUCTURES AND LEVELS*

In section 2.2 I have discussed the discrepancy between human and machine information processing. One obvious difference is the structure of knowledge.
The lack of structure in human knowledge is not so much due to its variety of expressions or to the complexity of the thought processes. The fact that much of human knowledge is compiled knowledge, resulting from more or less long experiential learning, seems to be a more powerful reason.
A computer system needs structure, at least if we want it to work efficiently.
In the previous section I went over the necessity of the compatibility of knowledge acquisition and knowledge representation. Structure is needed for adequate representation of knowledge. But the knowledge needs to be in a certain order for the system to run. The better the structure is, the better the performance of the system will be.
Another necessity for structure is in the initial state of expert-system building. It seems fruitless to get started in a knowledge-acquisition process with the domain expert without a plan. The domain expert is usually not the right person to give outlines of designing a system.

We can distinguish various kinds of structures:
1. Structures can be found in the domain, but these are not explicitly known to the domain expert. It is up to the knowledge engineer to elicit these structures.
2. There are structures of the system, whether expert-system shell or knowledge-representation language. If starting from scratch, the knowledge engineer should design one.
3. There is also a structure in the knowledge acquisition process.

All these structures have to do with the organization of knowledge in one way or the other.

Structures in the domain can be revealed on certain levels. The most basic - and the only really necessary - level is the recognition of type of problem the domain covers (diagnosis, planning, repair, etc.). From here on one can use the proposed structure of the system to map the structure of the domain. This system structure should be compatible with the domain problem structure. In this way the domain gets a more detailed structure from the system. It does not matter if this structure is unknown to the expert (or anyone else in that field), but it should at least become familiar to the domain expert. It is more important that the system can offer a structure in which the domain expertise fits. This also means that the system structure should be flexible enough to adjust to the domain knowledge. The knowledge engineer does not need more than a library of problem structures to capture all kinds of domains. (see section 3.8)

Since the system is highly structured I follow the principles of the knowledge-acquisition system KADS (Breuker and Wielinga 1985, 1987, De Greef and Breuker 1985, Wielinga and Breuker 1985, 1987).
KADS distinguishes four levels of expert knowledge, corresponding to different roles that knowledge plays in reasoning processes:
On the domain level are concepts, relations and structures represented. On these objects the inference level can be applied. The task level controls the goals and tasks. The strategic level controls the whole process.
In section 3.8 these levels are presented in a more elaborated way.

In section 3.30 more will be said about knowledge levels and schemata of the knowledge acquisition system TEIRESIAS.

## 2.7 PROTOTYPING

First some remarks about the controversy between rapid prototyping and structured knowledge-acquisition. Somehow two schools of knowledge engineering have evolved. The school of rapid prototyping claims that the best way to get along with the domain expert and to keep him interested is to make a prototype of the system in an early stage of development. This principle is used in the knowledge-acquisition technique described in section 2.1.
The other school uses the structured approach. Structured knowledge-acquisition is mainly developed at the University of Amsterdam. It is based on the principle that the structure of a system should be outlined in detail before any knowledge acquisition can be done. In an interview (Thiemann 1989) Wielinga, one of the developers of KADS, states that they are not opposed to prototyping. He thinks that while prototyping one can use the techniques of KADS at the same time.[5]
Prototyping is the implementation of a preview of the system that the knowledge engineer tries to build. It can be a significant part of the system or the whole system without all the details. There are advantages of prototyping for the domain expert and for the knowledge engineer.
First, we can mention the psychological effect of visualizing a system that is so far only an abstract entity. The effect for the knowledge engineer will be that he gets a better insight in the procedure. The same insight influences the domain expert in this respect that he will get a better understanding of what is expected from him. Another effect for both is the recognition of still uncovered possibilities in the system.

[5]A similar solution is used by the Dutch software house BSO that exploits a technique that takes advantage of both techniques.

An extra effect upon the domain expert seems to be that he will be more interested in the system since it has obtained more shape. (Among many others, see Welbank 1983, Van Dijk et al. 1988, Hayes-Roth et al. 1983)

In their "Practical Guide to Designing Expert Systems", Weiss and Kulikowski (1984) emphasize the importance of building a prototype as soon as possible. The reasons they give is that often knowledge engineers gather information about the domain over a long period of time without getting a good insight of that domain. The prototype allows the expert to give feedback and improvement will easily occur. They give some advice how this prototyping should take place:

1. Design a model by focusing on a small set of hypotheses and include in a first prototype only those findings that are most predictive of these hypotheses.
2. Identify clusters of findings that are most discriminating.
3. In the decision rules, combine the smallest number of findings necessary to confirm or discriminate among hypotheses. Increase the number of conjunctive rules when the resulting rule will significantly increase the power of the system to confirm or deny the conclusion. There is potentially a large number of combinations of findings. A model should contain the smallest number of these that are sufficiently specific to confirm, deny, and discriminate.
4. Include findings that may not be strongly predictive or discriminatory on their own, but which can significantly improve the quality of decisions by setting a context or focus of attention for the decision-making process. The prototype system at this stage becomes more realistic.
5. Determine whether abstractions can be made. For example, some production rules can be satisfactory if one or more out of a list of findings are satisfied.
6. See whether additional intermediate hypotheses can be introduced to simplify reasoning.
7. Test the model on a data base of cases.

Prototyping has many advantages. It seems that rapid prototyping can be combined with any other knowledge acquisition approach. The idea of omitting the prototyping and waiting till the system can be implemented does not seem to be successful.

## *2.8 UNCERTAIN KNOWLEDGE*

In the knowledge-acquisition process the knowledge engineer has sometimes to deal with uncertain knowledge. Experts tend to be inaccurate and make errors when it comes to processing uncertain knowledge. A method to overcome this problem is to offer the domain expert a point of reference that divides the range of confidence into two parts. The domain expert can choose his confidence level. This procedure can be repeated several times. This method is called "providing anchor points" (Hink and Woods 1987).

Probabilistic judgments are more often based on a positive or negative impression, or on prejudices, than on logic reasoning. Humans use heuristics to process uncertain information:
1. Representativeness, which is based on the assumption that the more an object typifies a corresponding class, the higher the probability of a relationship between the two.
2. Availability, where easily recallable information has higher associated probability than less recallable information.
3. Adjustment and anchoring which cause over- and underestimation of usually infrequent and

frequent occurring events.

The best way to reduce the effects is to avoid using probabilistic or statistical judgments by the domain expert as much as possible (Hink and Woods 1987).

Another approach is suggested by Gruber and Cohen (1987). They have developed a medical expert system MUM (Manage Uncertainty in Medicine) that is designed to combine evidence and knowledge and to support control knowledge (knowledge about what to do).

The combining knowledge helps to avoid uncertain conclusions from given uncertain knowledge.

Combining knowledge specifies how belief in several pieces of evidence is combined to support a single conclusion.

1. It replaces the real-valued numeric representation of uncertainty with symbolic states of belief that are meaningful in domain terms.
2. It provides an explicit representation for clusters of evidence to encapsulate diagnostically significant subsets of evidence.
3. It replaces the global numeric function with local combining functions, specified by the expert, for each cluster of evidence.

MUM represents belief as ordinal values that characterize the expert's evaluation of evidential support (such as "confirmed", "supported", etc).

MUM represents combinations of evidence with clusters (frames) that represent diagnostically significant groupings of evidence. (Diseases are clusters.)

The representation of combining knowledge is designed to facilitate knowledge acquisition:

The ordinal states of belief are chosen to be sufficient to characterize diagnostically significant situations, and nothing more. The symbolic combining functions are explicit, declarative representation of decisions about evidential support. Instead of representing degrees of belief and computing the results, the evidential judgments are represented.

Some systems solve the uncertainty problem by using fuzzy logic.

## *2.9 KNOWLEDGE-BASE REFINEMENT*

Knowledge-base refinement can be regarded as the second phase in the knowledge-acquisition problem. After the knowledge engineer has extracted the initial knowledge from the expert the knowledge has to be refined into a high performance knowledge base.

Welbank (1983) enumerates the problems the knowledge engineer has to overcome:

1. 1. There are gaps in the knowledge, particularly missing constraints. Rules may turn out to be applicable in the wrong circumstances.
2. Rules overlap, leading to inconsistent or redundant conclusions.
3. Rules interact in unexpected ways.
4. New information could be put into several parts of the program. The knowledge engineer must decide the most appropriate place to put it.
5. Changes should propagate through all parts of the program.
6. The knowledge engineer loses his understanding of the knowledge base as a whole.
7. Rules are not strictly independent.

The system SEEK2 (Ginsberg et al. 1985) uses cases that are evaluated by the expert to refine the

rules involved.
The fundamental assumption is that case knowledge can be used to drive a process involving empirical analysis of rule behavior in order to generate plausible suggestions for rule refinement. Case knowledge is given in the form of a data base of cases with an expert's conclusions. Empirical analysis of rule behavior involves gathering certain statistics concerning rule behavior with respect to the data base of cases; suggestions for rule refinements are generated by the application of refinement heuristics that relate the statistical behavior and structural properties of rules to appropriate classes of rule refinements.

Semantic consistency checking helps to detect inconsistencies between the knowledge base and revisions.
A system like TEIRESIAS (see section 3.30) can match new rules against other similar rules to check consistency and completeness.
EXPERT and EMYCIN use automated testing of a large number of problems to see if the knowledge base is well revised. This reveals the best revisions and also the weak spots in the knowledge base (Buchanan et al. 1983).

Most of the automated knowledge-acquisition systems that are described in the next chapter have built-in facilities to refine and debug knowledge bases. In general it can be said that automated systems are the chosen tools to find inconsistencies and incompleteness. Specially when knowledge bases are large the knowledge engineer is unable to overview all rules and their intertwinements.

## 2.10 THE ULTIMATE KNOWLEDGE BASE

A distinctive solution to overcome the knowledge-acquisition bottleneck can be found in the CYC project (Lenat et al. 1986).
The CYC project is the building of a large knowledge base of real world facts and heuristics and methods of efficiently reasoning over the knowledge base. Common sense reasoning and analogy can widen the knowledge-acquisition bottleneck.
General knowledge can be broken down into a few types:
1. The real world factual knowledge, the sort found in an encyclopedia.
2. The common sense knowledge that an encyclopedia would assume the reader knew without being told.
3. The general knowledge we have acquired might be a source for new knowledge by means of analogy or heuristics.

The large knowledge base of general knowledge that CYC will hold helps to acquire knowledge in an easier way. We assimilate new information by finding similar things we already know about and recording the exceptions to that analogy. So, the more we know, the more we can learn.
Causal meta-knowledge will play a key role in deciding how to find and extend analogies. One of the tasks is to taxonomize slots and build that hierarchy into CYC's knowledge base.
Frames can have slots with identical names and values. Or frames can have identical slots but different values.
An important part of CYC is its large, organized body of reasoning methods. These are being described declaratively in CYC in a network of frames spanning both problem-solving architectures and specific heuristics. Included are analogical and common sense reasoning methods as well as more traditional problem-solving techniques.

CYC's representation language is frame-based and is similar to RLL (Greiner and Lenat 1980) and KRL (Bobrow and Winograd 1977).

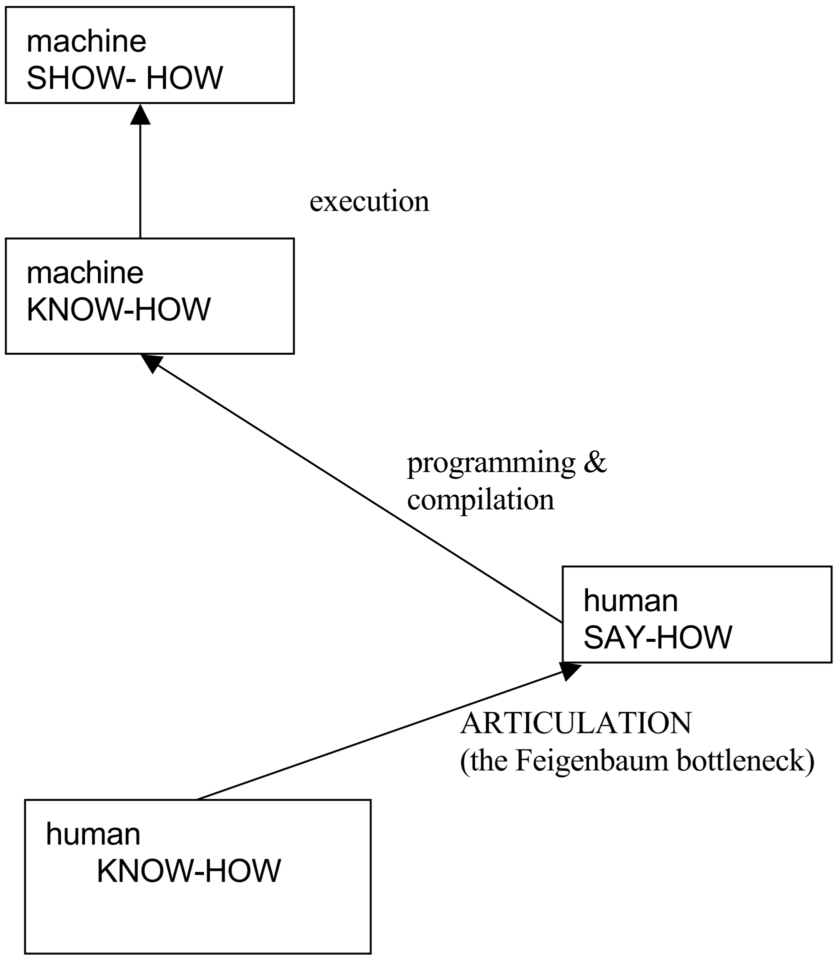
It seems that this knowledge base will replace human reasoning processes, common sense knowledge and encyclopedic knowledge. It is not clear yet if this will be necessary to acquire expert knowledge and to build an expert system.

## *2.11 MACHINE LEARNING*

The "Feigenbaum bottleneck" and the "Michie road"

The bottleneck in the knowledge acquisition process is specifically present in the transfer or articulation of the expert knowledge (his know-how) into the formulation (his "say-how") of this knowledge (see figure 2.2).

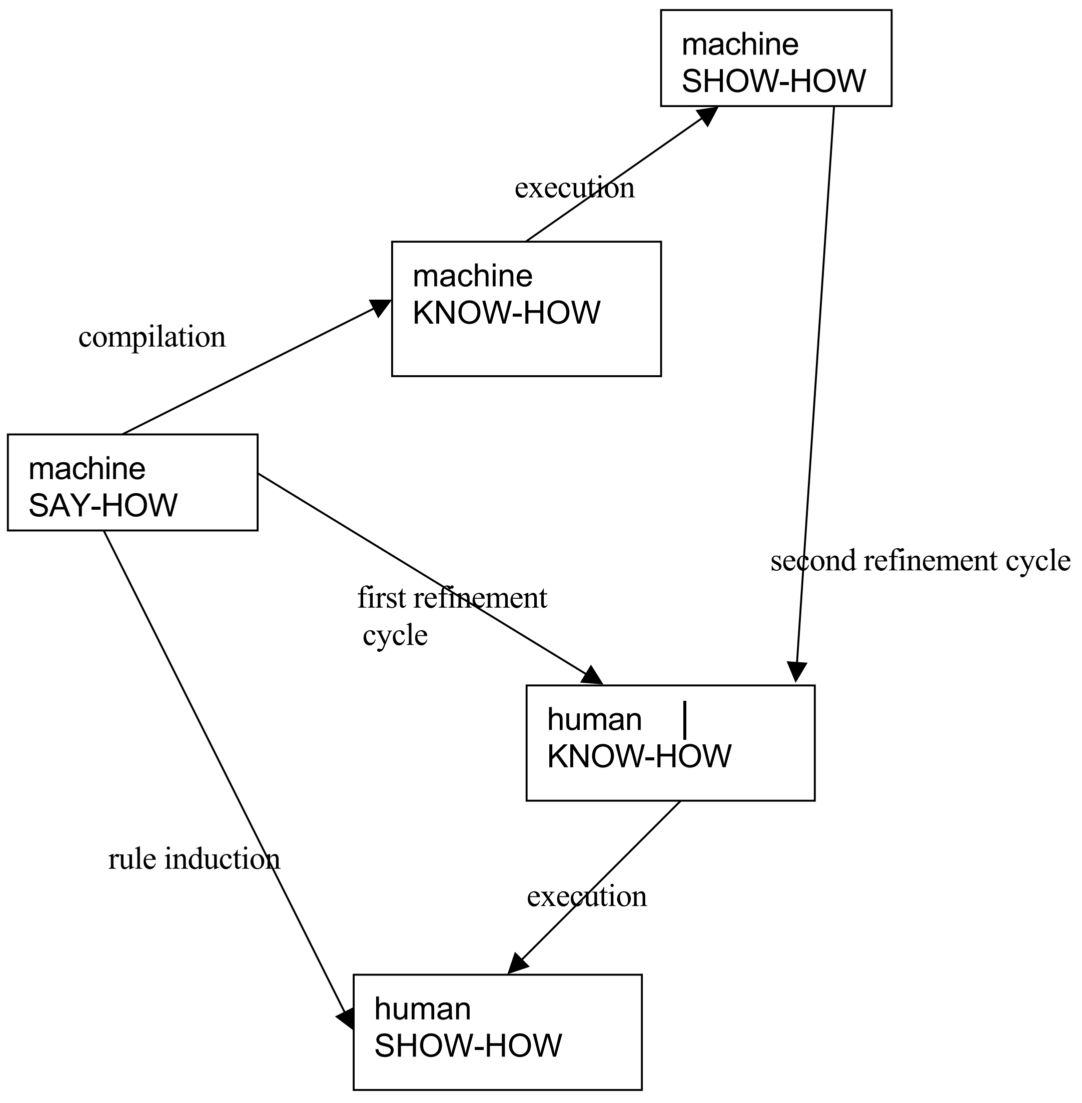
*********************************************************************************
*



***************Knowledge engineer's route map: old style *******************
figure 2.2 (from Michie 1987)

In Michie (1987) is shown how this articulation problem can be by-passed by a rule induction program or induction algorithm. Here the human know-how is represented by examples of his expertise (the human "show-how"), such as the expert uses to teach his skills to new recruits in a tutor-apprentice situation. From these examples the induction program can produce the rules for the knowledge base, the machine "say-how" and the know-how (see figure 2.3).
In a later stage of the process the "old route" can still be used. The human expert will be asked to refine the rules (the first refinement cycle). When the system is fully tested a second refinement cycle is necessary. The domain expert can debug, refine, and maintain the system (Michie 1987 and Van den Herik 1986).

*************************************************************



****************** Knowledge engineer's route map: new style **********************
figure 2.3 (from Michie 1987)

The induction principle originated in the ID3 algorithm (Quinlan 1979).
This ID3 algorithm produces a decision tree for differentiating positive and negative instances, given a collection of instances of a concept described in terms of attributes or properties.
The commercial version of ID3, Expert-Ease, is mentioned in section 3.32. RuleMaster (see section 3.26) is an extended version of the original ID3 algorithm.
A drawback of these induction programs is the absence of the necessary background knowledge to select the attributes and to determine the relations between attributes (Goodall 1985).

In their 1983 contribution Buchanan et al. were rather skeptical about using machine-learning techniques in the knowledge-acquisition process. However, the Meta-DENDRAL program that could learn rules from cases was developed and successfully used about 20 years ago.
Welbank (1983) also mentions the following limitations:
1. An induction system would need a database of documented cases to examine. Many domains of human expertise cannot supply this. 2. The rules that are induced from a set of cases will not be the same as a human expert uses.
Therefore this approach is not eliciting human knowledge at all. Human rules may be more computationally efficient than a random set of rules that completely covers the same domain. Human rules may also be more robust, whereas a set of rules induced by a machine from one collection of cases may account very poorly for a second set of cases. Some induction algorithms would produce a different rule every time a new case is added to the library.
Machine-induced rules which do not correspond to the rules that human experts use could also be very difficult to grasp. They would have lost the advantage of transparency.
As long as machine induction produces rules which split the domain up in different ways from those the expert uses, it has serious limitations as a knowledge-elicitation method.
In the years afterwards several automated knowledge-acquisition systems using machine learning have been developed. They are discussed in the next chapter.

There are many ways in which machine learning can be useful in the knowledge-acquisition process. First machine-learning systems might contribute to the initial construction of a knowledge base. Second, machine-learning systems might refine existing knowledge bases. Third, machine-learning systems might be helpful in adapting a knowledge-based system, for example, to accommodate user expertise or style. Finally, in general, machine learning might provide a principled method for constructing knowledge bases for expert systems, replacing the ill-defined engineer versus expert knowledge-acquisition interaction. Thus, it may provide a formalized mechanism for the knowledge-acquisition process (Shalin et al. 1988).
Michalski (1987) gives an overview of present-day machine-learning techniques for knowledge acquisition:
Machine-learning strategies reflect the type of inference performed by the learner on the input information in order to derive the desired knowledge. They include learning from instruction, learning by deduction, learning by analogy and learning by induction. Special attention is given to two basic types of learning by induction: learning from examples (concept acquisition) and learning from observation (concept formation without teacher). A specific form of learning from observation is conceptual clustering. Conceptual clustering is a process of structuring given observations into a

hierarchy of conceptual categories.
An inductive learning system generates knowledge by drawing inductive inferences from the given facts under the guidance of background knowledge. The background knowledge contains previously learned concepts, goals of learning, the criteria for evaluating hypotheses from the viewpoint of these goals, the properties of attributes and relations used to characterize observed events, and various inference rules for transforming concepts or expressing them at different levels of abstraction.

A long-term solution for the knowledge-acquisition problem is seen in the development of machine learning. Knowledge acquisition is specially emphasized through inductive learning, learning from examples, and learning by observation and discovery.
The knowledge-acquisition process can be greatly simplified if an expert system can learn decision rules from examples of decisions made by human experts, or from its own errors. This type of learning strategy is called learning from examples (or concept acquisition).
Learning from examples is one of several fundamental learning strategies. These strategies are identified by viewing a learning system as an inference system. Namely, they are distinguished by the major type of inference the learning system (human or machine) performs on the information provided, in order to derive the desired knowledge.

Learning strategies:
1. Direct implanting of knowledge.
2. Learning from instruction, also called learning by being told. A learner selects and transforms the knowledge from the input language to an internally-usable representation and integrates it with prior knowledge for effective retrieval and use.
3. Learning by deduction:
a. A learning system that uses this strategy conducts deductive (truth-preserving) inference on the knowledge it possesses and knowledge supplied to it.
b. A form of deductive learning, called analytical or explanation-based learning, has recently become an active research area. In analytical learning, the system is already equipped with a description of the target concept, but the description is expressed at the level of abstraction too high to be directly usable.
4. Learning by analogy. This strategy involves transforming or extending existing knowledge (or skill) applicable in one domain to perform a similar task in another domain.
5. Learning from examples. Given a set of examples and (optionally) counter-examples of a concept, the learner induces a general concept description. The amount of inference performed by the learner is greater than in learning by deduction or analogy, because the learner does not have prior knowledge of the concept to be learned, or knowledge of a similar concept. Learning from examples is also called concept acquisition. When a system determines examples by a search or other active effort this is called learning by experimentation. Learning from examples is one form of inductive learning. Another form is:
6. Learning by observation and discovery.
a. Passive observation, where the learner builds a description of a given set of observations.
b. Active experimentation, where the learner makes changes in the given environment and observes the results of those changes

The learning strategies mentioned above were presented in order of increasing amounts of effort required from the learner and decreasing amounts of effort required from the teacher. They reflect the increasing complexity of the inference performed on the information given to a learning system

in order to derive the desired knowledge.
There are lessons for machine knowledge-acquisition to be drawn from the above considerations. One is that if we know precisely how to solve a problem, we should tell the computer the solution directly (i.e., program it). Teaching by instruction will be simpler and more productive than using a deductive or inductive learning strategy. Such teaching will be facilitated by having an appropriate knowledge-representation language and debugging tools. As there are many areas in which precise solutions are known and relevant concepts can be defined, this strategy has wide applications. Therefore, the development of appropriate knowledge-representation languages and support tools (both general and specific to a given domain) constitutes a major research area.

There are many application areas where precise concept definitions or algorithms are unknown or difficult to construct even in an abstract, non-operational form. Examples of such areas are technical, medical or agricultural diagnosis, visual pattern recognition, speech recognition, machine design, robot assembly, etc. Also, people often have difficulties in articulating their expertise, even when they know well how to perform a given task or are able to recognize a given concept without any difficulty. In such cases, applying an analogical or inductive machine-learning strategy seems quite desirable.
Inductive learning is a process of acquiring knowledge by drawing inductive inferences from teacher- or environment-provided facts. Knowledge acquired through inductive learning cannot, in principle, except for special cases, be completely validated. This is a well-known predicament of induction.
In order to perform inductive inference one can thus need some additional knowledge (background knowledge) to constrain the possibilities and guide the inference process toward one or a few most plausible hypotheses. In general, this background knowledge includes the goals of learning, previously learned concepts, criteria for deciding the preference among candidate hypotheses, the methods for interpreting the observations, and the knowledge-representation language with corresponding inference rules for manipulating representations in this language, as well as the knowledge of the domain of inquiry.

Given:
a) premise statements (facts), F, that represent initial knowledge about some objects, situations or processes;
b) a tentative inductive assertion (which may be null); and
c) background knowledge (BK) that defines the goal of inference, the preference criterion for ranking plausible hypotheses, assumptions and constraints imposed on the premise statements and the candidate inductive assertions, and any other relevant general or domain specific knowledge.

Find:
an inductive assertion (hypothesis), H, that, together with background knowledge, BK, tautologically implies the premise statements.

This form of knowledge acquisition relieves the expert from the tedious task of defining rules himself. Moreover, it requires the expert to do only what he can do best: make decisions. Experts are typically not trained to analyze and explain to others their decision making processes, especially if they must express them in a formal way. Therefore, such tasks are usually difficult for them to perform. Once rules are acquired from examples, experts can usually do a good job in evaluating them (Michalski 1987).

Shalin et al. (1988) give methods to find the right sort of machine-learning technique:
This approach to identifying relevant machine learning systems for knowledge acquisition involves four steps:

1. Characterize machine-learning systems according to a general framework so that these systems can be evaluated comparatively according to a standard set of issues.
2. Characterize the general learning problems in an expert system that can be addressed by machine-learning systems.
3. Assess the match between a machine-learning system and a learning problem according to a set of identified dependencies (attributes and relations) between features of machine learning systems and features of application domains.
4. Evaluate a learning problem/machine learning system match based on the satisfaction of these dependencies.

Despite its limitations machine learning as knowledge-acquisition technique should be considered when the conditions for machine learning are advantageous.

## *2.12 COGNITIVE SCIENCE*

In section 2.2 I mentioned the mismatch between human and machine knowledge. Problems in expert-system building, and more particularly, in the knowledge-acquisition process, arise mainly from this mismatch. To grasp this problem we should know more about human knowledge.

Cognitive science is a recently new expanding field that explores human knowledge. It is a typical interdisciplinary area combining aspects of artificial intelligence, psychology, linguistics, and philosophy.
Here I refer to cognitive science as the study of human information processing. Human information processing deals with the acquisition and use of knowledge.
Two issues are important in cognitive science.

1. The representation: what is the nature of the knowledge structures.
2. The process: what is the nature of the human information processing system.

The human information processing system is the process of encoding, modifying and representing (storing) of information received by the various sense organs, the process that we call "thinking", and the verbal and motoric outputs based on the previous processes.
It is still a controversial topic how information is obtained and stored in the brain. How are symbol structures represented in the mind and how are operations executed on them.
In an attempt to explain more about the human information process system, cognitive scientists have introduced the hypothesis of mental models.
Among others, Gentner and Stevens (1983) and Johnson-Laird (1983) give plausible bases for mental models and their functions as explanation for the process of human psychology. Mental models are also called conceptual models or schemata.
With mental models we can explain the working of human thinking to a certain extent. Mental models can be formed about everything of which we have some sort of knowledge, perceptual or verbal. We can say that in mental models knowledge is encoded. A conceptual model is the problem space in which men considers understanding and dealing with the object of knowledge.

Interaction of a mental model of the environment with existing mental models of former experiences provide explanation for understanding new situations and how to deal with them.
Conceptual models are formed through analogy with other existing conceptual models and result in identical understanding reactions (Gentner and Stevens 1983).

Simon (1978) postulates that the brain operates basically as a system of labeled associations. The verbal representations are probably like the deep structures as postulated by the transformational linguists. Visual representations hold or maybe even generate information about spatial figures. Conceptual representations are more abstract and can handle abstract meanings. Mental models are simpler - as models usually are - than the entities they represent.
Mental models are analogical representations that are verbal or pictorical and in various degrees of abstraction.

Haugeland (1981) thinks that the mind is a human information processing system that is not related to language. Language is an external mapping, while the mind, or the human information processing system, is a private internal mapping without relation to language.
There are reports about children that have been deprived of any kind of language acquisition during their childhood but are able to tell later about their prelanguage experiences. More commonly known are studies in dyslexia about persons who cannot understand words, although they are known. This should prove that mental representations are not necessarily verbal. On the other hand it does not prove that mental representations are necessary nonverbal. Further experiments have shown that visualisation helps the understanding of verbal texts. Verbal recall without visualisation is seldom accurate.

How does this fit in with knowledge acquisition? It seems that the study of human information processing is an integrated part of the study of knowledge acquisition.
The transfer of knowledge is done by abstract concepts. The ability to formulate abstract concepts is a basic cognitive capacity.
Man has learned to categorize while he is perceiving. In that process he loses the particulars and stores only the concepts. The process of thinking moves also on a level of abstraction, where the mind is swept clean of particulars, i.e. verbal and perceptual material, i.e., words and images.
When he learns a skill he learns structures, rather than loose rules. He develops new structures by applying metaphors on structures he already knows. Experts are able to pick up knowledge patterns easier than novices. But also the knowledge engineers are not able to pick up these patterns as easily as the experts.
The reproduction of acquired knowledge in words needs a kind of translation, from "mentalese"[6] into a natural language like English. Natural languages consist of concepts that have mental background as well. The mapping of structured knowledge into English sentences without losing too much information is a problem.
If we transfer this process to the knowledge-acquisition process in expert-system building, the next step is the translation of these sentences by the knowledge engineer into his mentalese and from there he has to manipulate these structures to transform them in the right way in a knowledge-based system.

This whole process is so tightly interwoven with cognitive processes in general and mental models

[6]The language of the mental.

in particular that it seems to be of crucial importance for a better understanding of knowledge acquisition.

# CHAPTER 3

# AUTOMATED KNOWLEDGE-ACQUISITION SYSTEMS I: DESCRIPTIONS

In this chapter more than 30 knowledge-acquisition tools are described.

The quantity and depth of information that has been obtained from each system vary greatly. TEIRESIAS is one of the best documented knowledge-acquisition systems (Davis and Lenat 1982). This should not be a surprise since TEIRESIAS is a member of the well published Stanford-based family of MYCIN. Other well described systems were found in journal articles, especially in the International Journal of Man-Machine Systems.
Little material was gathered from the Proceedings of the AAAI and IJCAI conferences. The papers in these proceedings were often too short to give a full description of a system. Often only certain aspects of a system were mentioned.

Bibliographic searches revealed an amount of reports on knowledge acquisition, but after closer investigation it turned out that most reports were working papers on recently started projects. I decided to include only those systems that were at least reviewed by an editorial board of a journal or conference committee. Notwithstanding this restriction, the articles do not always give sufficient information on all the questions one can ask. It was also not often clear whether a system was still in its infancy of development or already a successful tool, to mention the extremes. However, it should not be in conflict with the intention of this paper when only the method of a knowledge-acquisition system is described and not the actual working of an existing program.
Literature about commercial available systems was surprisingly difficult to obtain. (see section 3.32)

One can discuss the sequence of the systems.
In several cases the chronological order of development, e.g., ETS before AQUINAS, would be preferable. In most cases however, the systems were developed independently from each other.
Another way might have been grouping the systems according to their principles. But, as we shall see in Chapter 4, there are many ways to classify these systems.
After ample consideration, the alphabetical enumeration of the systems was chosen for easy reference.

To preserve the meaning of the authors their systems are described as much as possible in their own words. For the sake of readability quotation marks are omitted.

## *3.1 AQUINAS*

AQUINAS is an expanded version of ETS (see section 3.6).
It is able to handle better than ETS:
- deep knowledge
- causal knowledge

- relationship chains
- different levels of abstraction in a single grid
- the manipulation of large single grids.

The tasks of AQUINAS are:
- to elicit distinctions
- to decompose problems
- to give methods for combining uncertain information
- to test knowledge
- to make automatic expansion and refinement of the knowledge base
- to use multiple sources of knowledge
- to provide guidance to the knowledge-acquisition process.

A knowledge base can be constructed along the following steps:
1. Elicit cases and the initial grid (solutions, traits and ratings)[7].
2. Analyze and expand the initial, single grid.
3. Test the knowledge in the single grid.
4. Build hierarchies (structured as solutions and traits in multiple grids) from the first grid.
5. Use several rating value types (transform ordinal ratings to nominal and interval ratings) to represent knowledge.
6. List knowledge in hierarchies, test knowledge from multiple experts.
7. Edit, analyze and refine the knowledge base, building new cases.
8. Further expand and refine the knowledge base.
9. Generate rules for expert system shells.
   (Boose and Bradshaw 1987, Boose 1988)

The Dialog manager, a subsystem of AQUINAS, provides guidance in the knowledge-acquisition process. It contains heuristics for the modeling of expertise and for the use of AQUINAS during knowledge acquisition.
The Dialog Manager has been implemented as an expert system. Its domain knowledge is the effective use of AQUINAS. A rule-based approach to implementation was selected because of its flexibility, maintainability, and modifiability. Heuristics are encoded as rules directly within the AQUINAS system.
In the automatic mode, the Dialog Manager applies knowledge acquisition heuristics to determine the best Aquinas operation to be performed next within the context of the current state of the knowledge base and the characteristics and preferences of the expert. The Dialog Manager displays the recommended command sequence, an explanation of what it is doing, and the reason it has chosen that option.

**Application areas**

Repertory grid-centered tools work best on analysis problems, or those portions of synthesis

[7] see section 3.6 for the meaning of "grid"

problems that can be reduced to analysis problems. Analysis problems are those whose solutions can be comfortably enumerated (classification, interpretation, diagnosis), while synthesis problems are those whose solutions are built up from components (configuration, design, planning). Even with this limitation, these tools can be applied to a wide range of application problems.

A general problem when modifying knowledge bases is that changes may degrade system performance. This is especially a problem when the knowledge base is large. It may be unclear how changing one item in a knowledge base containing thousands of items will affect overall system performance (Kitto and Boose 1987).
In Shema and Boose (1988) some analyzing and refining mechanisms to control this problem are discussed.

AQUINAS can be seen as representative of both repertory grid and general-purpose knowledge acquisition. Its strength is based in the variety of ways that experts are allowed into viewing their problem-solving knowledge (Kornell 1987).

## *3.2 ASTEK*

ASTEK (Acquisition of STructured Expert Knowledge).

The path that has been taken in knowledge-acquisition tools was to start with a simple natural-language solution and extend it where the natural-language paradigm failed to meet the needs. This went in the direction of a form and editor paradigm of knowledge transfer. This paradigm better supports utterance planning, fragmented articulation and variations in the medium of articulation. The system looks much like OPAL (see section 3.22), except that form entries themselves may be complex structures or references to complex structures.
Natural language plays a sufficiently important role in ASTEK. At the level of small knowledge fragments, natural language grammars support the medium of articulation when preferred. At a deeper level, ASTEK was designed with the realization in mind that automated knowledge acquisition is an extended dialog between a human expert and the computer. Drawing on work related to the needs of extended dialogs, features are added such as object reference and a recommendation function, which make knowledge acquisition more than the isolated editing of templates. The discourse tools help individual acts of articulation be performed as desired while bringing together the individually articulated knowledge fragments into a single, coherent knowledge base as the acquisition task reaches completion.

A fundamental tenet of this approach to knowledge engineering is that effective knowledge-based systems can and should be built by designing cognitively appropriate knowledge structures, and implementing inference strategies as an operational semantics for these structures in a process called knowledge analysis.
ASTEK is a multi-paradigm knowledge-acquisition tool. Unlike efforts directed at combining the tasks of knowledge analysis and knowledge acquisition and efforts towards machine learning, ASTEK is focused on the expressibility aspect of the acquisition problem.

An acquisition tool must serve to:

1. Provide an external form for the knowledge that has clear, intuitive semantics to an expert in the domain.
2. Mediate the knowledge transfer from the external form, that which is natural to the domain expert, to the internal form, that which is most easily manipulated by the inference engine.
3. Support input of knowledge that is "correct by construction"; the acquired knowledge should be constrained to be correct at the earliest possible time.
4. Support knowledge management, such as revision control, and incremental testing; the tool should act as a comprehensive knowledge-maintenance mechanism.

ASTEK is an extension of INKA.
The shift away from a pure language solution (as in INKA) has been the need to describe procedural knowledge in a convenient manner. The tree-like structure of diagnostic procedures lies in stark contrast to the inherently linear nature of natural language.
ASTEK is a general interface tool directed at aiding the user in efficiently communicating structured knowledge. As in OPAL the acquisition process in ASTEK is guided by a model of the domain knowledge. This model is constructed by specifying the types of knowledge structures for the domain in terms of their components and the appropriate editing mechanism to apply to each component. Also utilized by ASTEK in the diagnostic applications is a model of the system being diagnosed.

ASTEK is implemented (Jacobson and Freiling 1988).

## *3.3 AutoIntelligence*

AutoIntelligence is developed by IntelligenceWare Inc. It runs on an IBM-PC or compatible and costs $ 490.-.[8]

AutoIntelligence allows the experts to alternately identify important factors as well as working from concrete examples. Rules are then inductively generated from the accumulated knowledge and examples.
Designed to handle classification and diagnosis problems, AutoIntelligence enables an expert starting from selections or classes to define the characteristics of these selections, and to create examples using these definitions. These examples, coupled with the definitions, are then used to build a rule set, or expert system.

The AutoIntelligence program consists of five modules:

1. The Interview Manager interacts with the user, dynamically selecting the best interview style based on current progress.
2. The Structure Discovery System identifies the key components in user decisions, checking for redundancy and inconsistency.
3. The Example Manager tracks examples and handles the bookkeeping tasks for the user.

[8]AutoIntelligence, IntelligenceWare, ExpertEase, and IBM-PC are trademarks.

4. The Induction System classifies and generalizes the data and examples fed in by the user. Classification produces rules capturing the example set; induction produces general rules.
5. The Expert System Shell generates a working expert system which may be run or incorporated into another expert system shell.

Earlier attempts included ExpertEase and others, but they were too restrictive for general application. Other automated knowledge-acquisition systems do not take into account multilevel interview techniques.
With AutoIntelligence the user is cushioned with an interview-oriented technique, followed by rule-based induction.

## *3.4 BLIP*

BLIP (Berlin Learning by Induction Program) is mainly concerned with the construction of a domain theory as the first phase of the knowledge-acquisition process. BLIP is the learning part of LERNER. BLIP is an example of a machine-learning approach in knowledge acquisition. BLIP is implemented.
The philosophy behind this kind of knowledge acquisition is adopted from cognitive science. A "sloppy" domain model entered by the user and general structures that reflect structures of human thought are used, rather than structures of the world. The learning process is guided by meta-knowledge which can be viewed as the representation of cognitive structures that ease the knowledge-acquisition process.

BLIP requires the user to specify a "sloppy" domain model by defining predicates and the sorts of their possible arguments entering facts about the domain expressed with those predicates.
BLIP then discovers properties of these predicates and establishes relations between them, thus structuring the domain.
Finally, the domain model is transformed into the systems formal rules (Morik 1987).

Sloppy modeling is a paradigm for machine-supported knowledge acquisition that explicitly denies using structured elicitation dialogues for guiding the user through the knowledge acquisition process. Instead, by constructing an extremely flexible system, the user is provided a possibility to perform an unconstrained, exploratory first modeling phase with the help of the system.
In the sloppy-modeling paradigm, knowledge acquisition is viewed as a cooperative modeling process between user and system. This means that the user is not required to develop a complete and well-structured model beforehand in order to transfer it later on into the machine. Instead, the modeling activity itself becomes part of the system-supported knowledge-acquisition process: selecting the proper abstraction/granularity level for the knowledge to be represented, choosing the right concepts, determining their interactions, etc. In that respect, sloppy modeling shares a common view with advanced knowledge-elicitation systems (e.g., AQUINAS), and machine-learning approaches, where the above tasks are system-supported as well. It differs from those approaches in its emphasis on a cooperative, mixed-initiative modeling process. In a knowledge-elicitation system, the user is guided through a more or less fixed dialogue, answering the questions posed by the system. There is no way for the user to take the initiative, to go back and reverse decisions made earlier. In the sloppy-modeling paradigm, the goal is a modeling style where the user and the system's structuring and learning routines are regarded as independent partners that both work

towards having a well-structured and complete model.

As a knowledge-acquisition system, BLIP is designed to acquire encyclopedial knowledge. This is the basic problem-solving independent knowledge about a domain including terminology and simple empirical knowledge. BLIP can also be used directly with existing performance systems like HERACLES.
BLIP offers comprehensive facilities for entering, inspecting, and manipulating encyclopedial knowledge. Sloppy modeling is supported by using machine-learning techniques that discover inferential relations between existing knowledge.

BLIP's knowledge representation is based on a sorted logic augmented with selected higher-order constructs. The basic domain-level entities that a user works with are facts, predicates, and rules (Wrobel 1988).
The main components of the learning process are the generation, rating and testing of hypotheses.

Meta-knowledge in BLIP plays the central role in this approach to knowledge acquisition. Meta-knowledge is used for checking consistency; including rules from facts (the learning process); and deducing rules from other rules.
Meta-knowledge in BLIP (unlike that of AM, EURISKO, TEIRESIAS) refers to predicate constants, describing properties or relations of properties or relations.
In order to verify whether or not a meta-predicate holds for certain predicates, the criteria for the validity of a meta-fact are represented in such a way that they can be used as pattern for a search process in the factual knowledge of the inference engine. These patterns are characteristic situations (Morik 1987).

Design requirements for sloppy modeling systems:
Flexibility. First, the interaction with the user is organized around independent commands that are available at any time, and not around extended structured dialogues. That lets the system easily follow sudden ideas on the part of the user. Second, and more important, the "agenda of open ends" allows the user to postpone treating integrity constraint violations and return to handle them later, thus making sure such tasks don't get in the way of domain modeling.

Reversibility. BLIP is completely reversible with respect to the knowledge sources that are maintained by the inference engine, i.e., facts and rules at domain, meta, and metameta levels. For those knowledge sources, deletions and modifications are supported at any point. For other knowledge sources, those operations are supported, but not in the best possible way.

Integrity and consistency maintenance. Consistency maintenance is guaranteed in BLIP by the reason-maintenance procedures of its inference engine.

Transitionality. The BLIP system is transitional both in the traditional sense and in the sloppy-modeling sense. First, it allows users to begin working in a domain-oriented style by just inputting facts, predicates, and rules. This is supported by providing tools that automatically reexpress rules in the metafact-representation needed by the system. Second, and perhaps more important, in BLIP, the same representations and knowledge sources are manipulated by the user and the learning algorithms that support the system's part of the sloppy-modeling process. That way, the interaction with the system is always the same, whether the automatic modeling tools are used or not (Wrobel 1988).

## 3.5 (Meta-)DENDRAL

The heuristic DENDRAL program is designed to help organic chemists determine the molecular structure of unknown compounds.

Meta-DENDRAL
Because of the difficulty of extracting domain-specific rules from experts for use by DENDRAL, a more efficient means of transferring knowledge into the program was sought. Two alternatives to "handcrafting" each new knowledge base have been explored: interactive knowledge-transfer programs and automatic theory-formation programs. In this enterprise the separation of domain-specific knowledge from the computer programs themselves has been critical. Meta-DENDRAL has been constructed to help.
One measure of the proficiency of Meta-DENDRAL is the ability of the corresponding performance program to predict correct spectra of new molecules using the learned rules (Buchanan and Feigen-baum 1978).

The rule-formation task that Meta-DENDRAL performs is similar to grammatical inference, sequence extrapolation, and concept formation and is known in AI as learning by example.
In contrast to statistical approaches, Meta-DENDRAL utilizes a semantic model of the domain.

The Meta-DENDRAL program itself is organized as a series of plan-generate-test steps, as found in many AI systems. After scanning a set of several hundred molecular structure/spectral data-point pairs, the program searches the space of fragmentation rules for plausible explanations and then modifies its rules on the basis of detailed testing. When rules generated from a training set are added to the model and another block of data is examined, the rule set is extended and modified further to explain the new data. The program iteratively modifies rules formed from the initial training set (adding to them), but it is currently unable to "undo" rules (Barr, Cohen, and Feigenbaum 1981).

## 3.6 ETS

The Expertise Transfer System (ETS) is a tool which:
- interviews an expert
- analyses the information gathered
- produces production rule knowledge bases.

ETS can help in the following ways during prototype development:
1. Identification phase; data and terms.
2. Conceptualization phase; key concepts and relations.
3. Formalization phase; ETS automatically maps the key concepts into several representations that can be used for automatic rapid prototyping.
4. Implementation phase; ETS automatically implements an initial prototype expert system by building a knowledge base for a target production rule expert system building tool.
5. Testing phase; several tools and techniques are provided for incrementally improving ETS's

knowledge base, based on knowledge expansion methods and feedback from the prototype expert system.

ETS uses interviewing methods based on the psychology of personal construct developed by George Kelly. His Repertory Grid Test can be used to list, compare and rate a collection of items.
ETS helps the expert to construct and analyze the initial set of heuristics and parameters. The consistency of the method used by ETS guarantees a better result over manual methods.
ETS uses an interview technique in which elements are elicited one at a time. The expert is asked to compare packages of elements, name attributes and name attributes that distinguishes members from each other. The knowledge remains in the expert's own terminology.
ETS asks the expert to rate each element against each pair of traits. Thus a rating grid of values is formed.
The system builds an entailment graph of implication relationships. The strength of the implications is listed numerically. The expert might add to and delete from these knowledge-base structures.
ETS can now generate conclusion rules from ratings in the grid, after the expert has assigned concept names to each pair of traits. He is then asked to rate the relative importance of these concepts in his problem-solving behavior.
The other kind of production rules that are generated are intermediate rules. These rules are implication rules based on the relations in the implication graph with their relative strength assignments.
The expert may review these rules, improve them, and test the prototype.
ETS is applicable to structured selection ( analytic) types of expert system problems, particularly classification problems.
ETS has been used to build consultation systems that combine expertise from multiple experts in the same domain.
ETS has been used to build consultation systems that combine expertise from different domains and different aspects of the same general domain.
ETS has been developed at the AI Center of Boeing Computer Services. Several hundred prototype consultation systems have been built using ETS (Boose 1985, Boose 1986b).

**The limits of ETS**

ETS can handle analysis class problems such as debugging, diagnosis, interpretation, and classification.
The system cannot readily handle synthesis class problems (design and planning) or combinations of analysis and synthesis such as control, monitoring, prediction, and repair.

ETS is not suitable for "deep" causal knowledge, procedural knowledge, or strategic knowledge. It is not about "how" and "when".
AQUINAS (see section 3.1) can be regarded as an extension of ETS's possibilities.
Further knowledge can be added manually.

A combination with MDIS (Antonelli 1983), a maintenance, diagnostic, and information elicitation system is worth trying.
The Maintenance and Diagnostics Information System (MDIS) is a multi-domain knowledge acquisition system that supports integrated diagnostics, maintenance, training, data collection and data analysis used for military systems.

It is not much more than a guide through the expert system(s). It asks the expert for information and review (Antonelli 1983).

Others (Becker and Selman 1986) mention the following limitations:
ETS can only make correlations between data, but not with the real world.
The rules that ETS produces can only be accepted by production ruled systems.
The system is not fully automated. It needs still augmentation with human interviewing.
This last argument is presently not a relevant argument, because all the so called automated knowledge acquisition systems need manual intervention (monitoring, refining, etc.).

Boose (1986a) mentions the desirability to get knowledge from various experts. A system has been developed that can combine expertise from several experts into one expert system with help of the ETS technique. The end user can choose which experts he wishes to consult. He can receive recommendations based on a majority opinion and a dissenting opinion among the selected experts.

**Appendix; Kelly's Personal Construct theory.**

Kelly's theory of a personal scientist (Kelly 1955) was that each individual seeks to predict and control events by forming theories, testing hypotheses, and weighing experimental evidence.
Certain techniques for use in psychotherapy were developed by Kelly based on this philosophy. In a Repertory Grid Test for eliciting role models, Kelly asked his clients to list, compare, and rate role models to derive and analyze character traits. Aspects of these role models were used to build a rating grid. A non-parametric factor analysis method was then used to analyze the grid. The results helped Kelly and his client understand the degree of similarity between the traits. He named a trait and its opposite, a construct, and hypothesized that each construct represented some internal concept for the client (Boose 1984).

## 3.7 INFORM

INFORM (INfluence diagram FORMer) is an expert-directed knowledge-acquisition aid and interface for building knowledge-based systems in IDES (the Influence Diagram-based Expert System for probabilistic inference and planning).

The INFORM architecture is based on information requirements and modeling approaches derived from both decision analysis and knowledge engineering. It is best suited to heuristic classification problem-solving, in particular domains with diagnosis or decision making under uncertainty.
INFORM is a top down design aid using descriptions of the domain concepts and structure, rather than on examples of problem solving in the domain.
The INFORM architecture falls between the technique of rapid prototyping and testing, and the structured knowledge-engineering approach, which guides and supports the initial knowledge-acquisition phase.
INFORM uses model refinement techniques from decision analysis and knowledge engineering in an environment that is predominantly structured knowledge acquisition.
Decision analysis separates the process into deterministic structuring, probabilistic assessment, and

informational phases. Assessment and modelling procedures direct the formation of choices, information, and preferences into the decision set.
Influence diagrams are conceptual and operational representations for domain expertise. They are used as a knowledge structure: an operational way of organizing knowledge, without cognitive claims.

There are three hierarchical levels: relational, functional, and numerical.
At the relational level the interdependence of uncertain events is represented. They superficially resemble semantic nets and frames.
The functional level is a specification of the type of relationship between events.
The numerical level is a quantitative measure of the "extent" of the relationship. The influence diagrams are based on Bayesian probability.

INFORM applies best under heuristic classification problem-solving. The decision analysis approach may be a viable way to approximate expert performance.
INFORM is responsible not only for meeting the information needs of the computational knowledge representation, the influence diagram, but for meeting the information needs of a knowledge engineering process: context definition, model structuring, model refinement, and process decision-making.

Three basic types of information INFORM must represent:

1.Model information.
Computational model of information: nodes, states, probabilities etc. - the representational requirements of the formal influence diagram.
Structural model information - the influence diagram with context and assumptions.
Uncertainty model information - people's numeric estimates of uncertainty do not accurately represent their underlying judgement without some structured revision and debiasing.
2. Procedural information.
This is information about the knowledge-engineering process.
3. Insight about the model.
"Insight" is the creation and revision of a mental picture of the domain and its processes. INFORM provides a medium for this insight.

INFORM intends to achieve:
sufficiency, getting the encoded information right in terms of 1. the influence diagram representation,
2. correctness of expert judgement, and
3. providing insight.

Architecture of INFORM. There are 4 conceptual levels: The first level is to fill the diagram. The second is to capture the activities that decision analysis and knowledge engineering employ: diagram drawing, etc. The third level is the "heuristic" approach for the encoding process. The fourth level provides requests for explanation and reformulation.

Finally, INFORM has two approaches to knowledge structuring
and refinement:
1. Short modeling at the most general level of precision.

2. Increase specificity only for the best improvements in model performance (Moore and Agogino 1987).

## 3.8 KADS

KADS (Knowledge Acquisition and Document Structuring)[9] is both a methodology for structured knowledge acquisition and a system.

The major purpose of the construction of the pilot version of the KADS system is to support the methodology construction.
The aims for KADS as a knowledge-engineering tool are the following:
Formalization, guidance, documentation support, consistency checking, on-line information retrieval, advise on feasibility, and automatic generation of parts of a knowledge-based system.
KADS is a system with four layers or levels of knowledge. These layers correspond to the different roles of knowledge in the reasoning process.
The first layer contains the static knowledge of the domain (concepts, relations, and structures).
The second layer is the inference layer, in which is described what inferences can be made from the domain knowledge. In this layer two types of entities are represented: meta-classes and knowledge sources. Meta-classes describe the role that domain concepts can play in a reasoning process. Knowledge sources describe what type of inferences can be made on the basis of the relations in the domain layer.
The third layer is the task layer with goals and tasks as basic objects.
The fourth layer is the strategic layer. On this level plans are made (e.g., to create a task structure), the execution of tasks is controlled, and faults are repaired (Wielinga and Breuker 1987 and Hayward, Wielinga, and Breuker 1987).

The KADS system consists of the following components:

1. Knowledge base which contains knowledge about the analysis task, a number of domain independent concepts and the domain knowledge. In the center of KADS is a semantic network, based on KL-ONE.
2. Inference machine. The KADS system has an inference machine in which rules can be formulated. The rules can be used for interpreting the obtained data. Rules can also be used to control input of data. The rules are part of the KL-ONE network.
3. Lexicon which contains lexical entries for the domain.
4. Analysis component which control the interactive analysis of the domain by instantiating and updating knowledge structures which represent the analysis task.
5. Knowledge-base editor and browser which provides facilities for changing and updating the knowledge base.
6. Document generator generates a document describing the content in the knowledge base and the lexicon for a particular domain.

Knowledge acquisition consists of the collection, elicitation and interpretation of data on the functioning of expertise in some domain, in order to design, build, extend, adapt or modify a knowledge-based expert system. In this way knowledge acquisition is a permanent activity throughout all stages of designing, implementing and maintaining an expert system.
Within the knowledge-acquisition task, two major subtasks can be distinguished:

[9] Also known as Knowledge Acquisition and Design Support

1. The elicitation of data on expertise.
2. The analysis of the (verbal) data. (Breuker and Wielinga 1985)

The choice of conceptual structure of a knowledge-based system is the most crucial step in the development process. From the analysis of the static domain knowledge, the function and task analysis, information is gathered to select and/or construct an interpretation model, using a library of standard models for prototypical domains. The particulars of the task at hand and the detailed analysis of expert knowledge are used to refine the model and to establish the detailed relations between the elements of the interpretation model, yielding the conceptual model of the prospective system. The decomposition of the knowledge-analysis task supports this incremental selection and construction of an interpretation model for a particular domain. The stages and sub-tasks outlined above can be seen as refinement of methodologies for building expert systems (Wielinga and Breuker 1985).

Verbal data do not speak for themselves; they have to be interpreted. However, there are no ready-made interpretation frameworks available that satisfy both the requirement that they should map (easily) onto some implementation formalism and that they should structure the data into a coherent description.
Data interpretation and instruction are modality tasks rather than proper problem-solving tasks (Breuker and Wielinga 1987).
The purpose of the interpretation process is to establish a mapping between verbal data and knowledge structures. This mapping can be performed on different levels, depending on the types of constructs that are used to express the knowledge.

For the purpose of mapping verbal data onto knowledge, five levels representing a synthesis between Sloman's classification and Brachman's representational levels are proposed (Findler 1979).

1. Knowledge identification.
This level of analysis corresponds to simply recording what one or more experts report on their knowledge. Although the result may be in a formalised form, the representational primitives on which this formalization is based are linguistic (in the sense that Brachman uses this term). The same knowledge of different experts may have to be represented differently, because they use different terminology, or because their knowledge is structured in a different way.

2. Knowledge conceptualization.
Knowledge conceptualization aims at the formalization of knowledge in terms of conceptual relations, primitive concepts and conceptual models. The knowledge of different experts, and possibly of different subdomains, is unified within one conceptual framework.

3. Epistemological analysis.
At the epistemological level the analysis uncovers structural properties of the conceptual knowledge, formalised in an epistemological framework. Such a framework is based on epistemological primitives representing types of concepts, types of knowledge sources, structuring relations (such as hierarchical relations, inheritance), and types of strategies.

4. Logical analysis.
This level of analysis applies to the formalisms in which the knowledge on higher levels is expressed and which is responsible for inference making.

5. Implementational analysis.
At this level of analysis, mechanisms are uncovered on which higher levels are based. The representational primitives are the ones which are normally used when an implementation of an artificial intelligence program is described (e.g., matching, testing, slot-filling).

Interpretation models
An interpretation model consists of a typology of basic elements and structuring relations for a certain class of domains. The basic elements are distinguished as (cf. Clancey 1983): objects, knowledge sources, models and strategies.
An object typology for a class of domains characterises the types of objects that have to be identified during the knowledge-acquisition process.

Following Clancey (1985) a knowledge source is defined to be a piece of knowledge that derives (infers) new information from existing data.
Models are knowledge structures which represent a set of complex relationships in a coherent structure, which can be used to predict new information.
The structure of the knowledge base can support particular problem solving strategies. Different ways of structuring the knowledge base to support different strategies have been discussed by Clancey (Wielinga and Breuker 1985).

The analysis process
The overall analysis process is defined as consisting of three phases. In the initial phase, the knowledge engineer becomes acquainted with the domain and the expert. Creating the domain lexicon by listing the key concepts, and an initial interpretation model will be selected or constructed. Once the scope of the system has been specified, this defines in broad terms the function(s) of the system and its users. The aim is then to construct
a detailed specification of the function the system has to perform. This specification includes the knowledge and strategies employed in the expertise. At this stage, processing of data becomes mainly model-driven. After a definition of the main- and sub-tasks involved in the performance of the systems's functions, the interpretation model may be refined and moves towards becoming the inference structure for the particular domain/task of interest. This involves fully specifying the meta-classes in terms of domain concepts, and the knowledge sources using domain relations; plus defining the task structure adequate for the performance of the desired functionality.

With an interpretation model identified, data can be collected and interpreted from expertise-in-action. Thinking aloud protocols may provide these data more adequately than interviews. Such on-line protocols are preferred to interviews or retrospective data, because there is ample evidence that experts do not necessarily employ the types of strategies and knowledge they may claim to use.

The interpretation model is used in such a way that the data from the domain and expert can be fitted within the structures provided by the model, thus producing a fully specified description of the expertise in a domain.
It may also be noted that one may adopt a more synthetic approach to the definition of the task level, since the available inferences in the domain may be manipulated in a way which does not directly mirror their uses by an expert. This may be the primary distinction between cognitive modelling and expert-system building.

It should be noted that the analysis process must also include an analysis of the environment for the proposed system and of the characteristics of the intended users (Hayward, Wielinga, and Breuker 1987).

There are three types of knowledge-engineering tasks in KADS:
1. An analysis of the functions, the environment and the users of the expertise to arrive at a definition of the operational characteristics of the prospective system. The functional analysis defines the modality of the expertise. A knowledge-based system contains two types of tasks: problem-solving tasks representing the expertise and communication tasks. The communication tasks are by no means trivial; they form the interface between the operational environment and the expertise. Modality may involve negotiating, explanation, coaching, documentation, etc.

2. An analysis of the static domain knowledge, starting with the collection of a lexicon, ending with concepts structured in KL-ONE concept hierarchies.

3. Analysis of expertise in action, i.e., the way problems are solved. This starts with a task-analysis: selection of one or more interpretation models that appear to represent the structure of the problem-solving process. By matching the verbal data from interviews and in particular thinking aloud protocols, this initial model gets refined and modified into a detailed structure of knowledge objects, knowledge sources and strategies; much in the same way as ROGET's (see section 3.25) conceptual structures. The final conceptual structure of expert reasoning represents the basic architecture of the prospective system. In the conceptual structure the static knowledge and the actions performed on them become integrated (De Greef and Breuker 1985).

The knowledge and expertise should be analyzed before design and implementation starts; i.e., the major efforts in knowledge acquisition should occur before an implementation formalism is chosen. Benefits are the following:
1. Feasibility of the domain for constructing an expert system can be assessed with few costs and at an early time.
2. The construction or choice of knowledge representation and inference formalisms can be motivated.
3. The analysis provides a detailed overview of the architecture of the prospective system.

Preferably, the analysis should be model-driven as early as possible. Models of expert problem solving not only enable the analysis of data but also provide references to known implementation solutions.
To bridge the gap between verbal data on expertise and implementation a model of expert problem solving should be expressed at the epistemological level.
The analysis should include the functionality of the prospective system; i.e., data on the environment and users should be collected and analyzed. These data are used for defining the communication tasks - modality - of the system.
The analysis should proceed, preferably, in an incremental way.
New data should be elicited only when previously collected data have been analyzed; i.e., elicitation and analysis should alternate.
Collected data and interpretations should be documented.

**Differences with ROGET**

In ROGET the data consist of direct answers from the user about concepts and facts of the domain. KADS however, also supports the elicitation and identification of knowledge.
A second difference is that in ROGET the refinement strategy follows a depth-first course after selection of the interpretation model; in KADS the breadth-first refinement is the consequence of broadening the scope of the analysis so as to include the modality of the system (Breuker and Wielinga 1987).

**A valuation of KADS**

A real problem in KADS is the partial attention for what is called "the problem solver". Confusion has appeared in several case studies. In the analysis the problem solver gets unexpectedly several responsibilities in the interaction with the user. This is the reason why complicated and unclear decompositions of the task are created. Already in an early stage it becomes inevitable to think in a procedural manner about the way the tasks will be executed. The problem might be the absence of a separation between the analysis of the problem-solver task and the task assignment (De Greef, Schreiber, and Wielemaker 1988).

KADS seems to be a highly structured methodology for expert-system building through verbal data interpretation.
KADS is more a knowledge-modeling tool than a knowledge-acquisition tool. Certainly it has contributed to a methodology of knowledge engineering.

The KADS system is implemented in PROLOG and runs.

## *3.9 KAE*

KAE (Knowledge Acquisition Environment) helps capture expertise from domain experts involved in analyzing scenes from aerial imagery. The aim of KAE is to integrate the domain inputs, the translation into internal representations and the actual execution and feedback.

Typical features:
The knowledge is visually oriented.
A tight coupling is required between the expert and the expert system being developed. Visual concepts are difficult to express, so interaction with the system is critical.
Multiple types of expertise are often required and multiple experts are likely to be involved.

The KAE architecture should provide support of a collection of computer-based tools facilitating:
- viewing and editing domain knowledge in both textual and graphic format
- translation of raw domain information onto an intermediate representation and finally translation into an executable format
- knowledge base execution and testing
- expert system performance analysis
- knowledge-base management.

A goal of KAE is to maintain as much domain independence as possible and still be useful (Tranowski 1988).

## *3.10 KAS*

KAS is the knowledge-acquisition system that was developed to facilitate the construction and maintenance of PROSPECTOR's and HYDRO's knowledge bases. PROSPECTOR is an expert system that was designed for decision-making problems in mineral exploration. HYDRO is a PROSPECTOR-based parameter estimation system.
These two systems employ various kinds of networks to represent knowledge - inference networks for expressing judgmental knowledge, semantic networks for expressing the meaning of the propositions that correspond to nodes in the inference networks, and taxonomic networks for representing static knowledge about relationships among domain objects.

The core of KAS is an "intelligent" network editor that can assist the user in building, testing, searching, and maintaining these networks. Its basic operations allow it to create, modify or delete various kinds of nodes and arcs. It knows, however, about the representation constructs and inference mechanisms employed by PROSPECTOR and can therefore protect the user against certain kinds of syntactic errors.
It also includes a bookkeeping system that keeps track of incomplete data structures. Whenever he desires, the user can turn control over to KAS, which will systematically question him to fill in the missing parts of the structures. A semantic network matcher gives the user a limited ability to access the knowledge base by content rather than by form. The matcher also supports features such as protecting against numerical inconsistencies in the inference networks, generating meaningful explanations, and enhancing the communication between the user and the system.
Finally, because KAS contains PROSPECTOR's inference mechanism as one of its components, it permits controlled execution of individual sections of an inference network, enabling the knowledge engineer to monitor his progress in refining the knowledge base
(Duda and Reboh 1984).

## *3.11 KEATS*

The development of KEATS was motivated by the idea of building a knowledge engineering toolkit that could provide a comprehensive range of tools to help the knowledge engineer fill the gap between the raw data and the final system.
Most toolkits for supporting the knowledge engineer are literally sets of tools rather than the implementation of a coherent theory of knowledge engineering. KEATS provides a semi-automated assistance throughout the knowledge-engineering process. At least it can facilitate knowledge elicitation and domain understanding.

KEATS consists of 4 integrated subsystems:

CREF is a dedicated editor that helps the knowledge engineer to organize the data gathered in protocol analyses.

KDL, a knowledge description language, is a frame-based representation language. It builds structures from atomic concepts and can draw links between concepts or concept structures. Properties of concepts can be described. The inheritance mechanism takes care of the sharing and deducing properties of the objects.

A graphical interface system functions as a blackboard and supports the building of the knowledge base. It displays the structures between the data.

A rule interpreter controls the integration in the knowledge base.

KEATS is not an automated knowledge-acquisition system with which a domain expert can communicate. The domain expert provides protocols in the ordinary way by tapes and transcripts. After the interviewing of the expert KEATS can be used in analyzing the transcripts. The authors correctly call KEATS a knowledge "elicitation" system, rather than a knowledge "acquisition" system (Motta et al. 1988).

## *3.12 KITTEN*

KITTEN (Knowledge Initiation & Transfer Tools for Experts and Novices) is an extension of PLANET (Shaw 1982).

A typical sequence in KITTEN is text input followed by text analysis through TEXAN which clusters associated words leading to a scheme from which the expert can select related elements and initial constructs with which to commence grid elicitation. The resultant grids are analyzed by ENTAIL which induces the underlying knowledge structure as production rules that can be loaded directly into an expert shell.
An alternative route is to monitor the expert's behavior through a verbal protocol giving information used and decisions resulting and analyze this through ATOM which induces structure from behavior and again generates production rules.
These two routes can be combined. KITTEN attempts to make each stage as explicit as possible, and, in particular, to make the rule base accessible as natural textual statements rather than technical production rules.

The KITTEN implementation is an initial prototype offering a workbench with minimal integration of the knowledge base, but each of the tools has already proven effective, and their combination is proving very powerful in stimulating experts to think of the knowledge externalization process from a number of different perspectives (Shaw and Gaines 1987).

Shaw and Woodward (1988) give a continuation of validation studies with multiple experts within the KITTEN environment. By knowledge acquisition from multiple experts there must be intra-subjective consistency and intersubjective consistency. The objective validity can be accomplished by measuring the degree of convergence of knowledge support system output with established facts from reputable experts and or from reputable printed material.
Another form of validity at this level concerns the usefulness of the output of the knowledge support system. Operative validity is defined as that knowledge which is necessary for accomplishing a task

or class of tasks. The usability of the system requires assessment. This refers to the ease of use and the understandability of the knowledge-acquisition tool by the expert.

## 3.13 KLAUS and NANOKLAUS

The problem is how to enable computer systems to acquire sets of facts about domains from experts. The type of acquisition process that is explored here is "learning by being told", in contrast to the more often used method of "learning by example".

The feasibility of such ideas is explored by developing a series of Knowledge-Learning And -Using Systems (KLAUS). A KLAUS is an interactive computer system that possesses a basic knowledge of the English language, is capable of learning the concepts and vocabulary of new subject domains, and has sufficient expertise to apply its acquired knowledge effectively in problem-solving situations.
Research issues:
- a powerful natural-language processing capability
- seed concepts and seed vocabulary
- other linguistic abilities, such as pattern recognition and uses of analogy
- a flexible scheme of knowledge representation.

So far a pilot-KLAUS has been developed called NANOKLAUS.
Most of the seed concepts in NANOKLAUS are classes of things and relations.
A fundamental task of the deductive system is to determine whether or not a given entity belongs to a particular sort
(Haas and Hendrix 1980).

Several years later the authors added the following: NANOKLAUS is best described as a fragile, proof-of-concept system that was built to establish the feasibility of achieving the broader KLAUS goals. NANOKLAUS has no provision for learning by analogy, acquiring or reasoning about the internal structures of processes, dealing with causality, handling mass terms, allowing users to change their minds about previously asserted "facts", or dealing with multiple senses of words (Haas and Hendrix 1983).

The natural language research group at SRI-International is now implementing MICROKLAUS that will cover a broader range of English.
In MICROKLAUS the parsing and translation system has been redesigned to provide for a declarative semantics that is easier both to extend and to maintain.
Also significant progress has been made on several fundamental problems of natural-language semantics (Grosz and Stickel 1983).

## 3.14 $K^n{}_AC$

$K^n{}_{A}c$ is a system that modifies an existing knowledge base through a discourse with a domain expert.

An often overlooked aspect of knowledge acquisition is the assimilation of information into an existing knowledge base. $K^n{}_{A}c$ accomplishes this assimilation by:

1. Comparing entity descriptions provided by the domain expert with existing knowledge-base descriptions,
2. evaluating these matches in the context of the knowledge-acquisition discourse,
3. making the modifications to the existing descriptions implied by the expert's information, and
4. generating (and managing) expectations of further changes to the knowledge base.

This knowledge-acquisition task may be viewed as a recognition problem.

$K^n{}_{A}c$ supports the domain expert by trying to assume much of the responsibility for assimilating the expert's information. To accomplish this, $K^n{}_{A}c$ models the knowledge engineer's role by anticipating modifications to the existing knowledge base using heuristic information about the knowledge-acquisition process. These anticipated modifications allow $K^n{}_{A}c$ to focus on "relevant" portions of the knowledge base and provide a context in which to integrate the information provided by the domain expert.

The descriptions obtained from the expert must be presented to the matcher in the knowledge base's representation language.
$K^n{}_{A}c$ provides a context in which to interpret information provided by a domain expert by anticipating modifications to an existing knowledge base. These anticipated modifications, or expectations, are derived from $K^n{}_{A}c$'s heuristic information about the knowledge-acquisition process.
Examples of typical heuristics: "Fields with too few components will be augmented" for dealing with incomplete knowledge.
Modification heuristics when a new entity description is added: "Detailed information usually follows the introduction of a new entity", etc.

The $K^n{}_{A}c$ system is implemented, but still in experimental use (Lefkowitz and Lesser 1988).

## 3.15 KNACK

KNACK is a knowledge-acquisition tool for building expert systems, called WRINGERs that evaluate the design of electro-mechanical systems.
KNACK's knowledge-acquisition approach is based on the assumption that an expert can adequately present his knowledge in the form of a skeletal report and report fragments. The skeletal report provides a framework around which report fragments relevant to the design of a specific electro-mechanical system can be organized. KNACK also elicits knowledge about how to customize the selected report fragments for a particular application.
In order to acquire the knowledge necessary to solve the information gathering and the evaluation tasks, KNACK exploits a WRINGER's problem-solving methods and knowledge roles to guide the expert through the knowledge-acquisition process. KNACK determines what an expert has to provide to define the knowledge. Finally KNACK uses heuristics to infer additional knowledge.

The problem-solving method and the knowledge roles the WRINGERs use to solve the evaluation problem can be summarized as follows:
1. Check the gathered information for consistency and completeness.
2. Evaluate the design description for possible flaws by using a worst-case analysis.
3. Evaluate the parts of the design description which showed indications of flaws again, this time using a precise analysis.
4. Make constructive suggestions about which pieces of information need to be modified or completed. Gather any missing information. Generate worst-case values for any required pieces of information still missing.
5. Integrate the gathered information into the associated report phrases, assemble the report phrases, include any evaluation messages, and write the report to an output device.

KNACK uses object-attribute-value tuples and relations as basic elements to represent knowledge. Each object may have multiple attributes. Dependencies between objects are represented by relations. These basic elements, object-attribute-value tuples and relations, are used to build the condition parts and the action parts of OPS5 rules. An OPS5 rule represents a piece of knowledge. The pieces of knowledge are organized into knowledge roles. A knowledge role is described by a corresponding knowledge role template.
In detail, the skeletal report defines the outline of an actual report and the order of report fragments within an actual report. KNACK assumes that the expert knows what information is needed, how to evaluate this information and how a designer should present this information.

KNACK uses heuristics to infer additional knowledge. The heuristics can be specific to infer additional knowledge for a particular knowledge role or they can be applicable to more than one knowledge role.
KNACK uses heuristics to insert conditions and relations between the objects of conditions into the question rule. These conditions and relations define the circumstances in which asking that question is appropriate.
KNACK proceeds through the stages of knowledge acquisition that are similar to the conventional approach of a knowledge engineer (Klinker et al. 1987).

In Klinker et al. (1988) it is described how KNACK can evaluate systems that are designed by multiple experts.

## *3.16 KREME*

KREME (Knowledge Representation, Editing and Modeling Environment) is built for developing large knowledge bases which future expert systems will require. Within KREME different kinds of representations (initially frames, rules, and procedures) can be used.

The approach to consistency maintenance has been to develop a knowledge integration subsystem that includes an automatic frame classifier and facilities for inter-language consistency maintenance. The frame classifier automatically maintains logical consistency among all the frames or conceptual class definitions in a KREME database.
Another area of investigation in developing KREME is the attempt to provide facilities for large-

scale revisions of portions of a knowledge base.
Finally, techniques for automatic generalization of concepts defined in a knowledge base have been investigated.

KREME attempts to deal with the inextricably related problems of knowledge representation and knowledge acquisition in a unified manner by organizing multiple representation languages and multiple knowledge editors inside a coherent global environment. Underlying the entire system is a strong notion of meta-level knowledge about knowledge representation and knowledge acquisition.

A central component of the KREME system design is that it incorporates tools for consistency maintenance both within and across representation languages. These tools are collectively referred to as the knowledge integrator. When new knowledge is entered or existing knowledge modified it is the task of the knowledge integrator to propagate, throughout the knowledge base, the changes that this new or modified knowledge entails, and to report any inconsistencies that have been caused by the change.
The knowledge integration subsystem for frames is basically an extension of the classification algorithm developed for the NIKL (the frame language component of KL-ONE) representation language.

Knowledge extension.
Experts have difficulties to formulate abstract classifications of problem types which are often unconscious generalizations about their domains of expertise. Currently KREME's frame generalization algorithm is able to search for sets of concept features that are shared by several unrelated concepts.

The goal is to explore a number of approaches to knowledge acquisition and knowledge editing that could be incorporated into existing and future development environments (Abrett and Burstein 1987).

## ***3.17 KRITON***

KRITON is a hybrid system for automatic knowledge acquisition.
Artificial intelligence and cognitive science are employing different knowledge-representation formalisms to construct knowledge bases.
Automated interview methods are used for elicitation of human declarative knowledge.
Protocol-analysis techniques are used for acquisition of human procedural knowledge.
Incremental text analysis is used for textbook knowledge.
The goal structure of KRITON is an intermediate knowledge representation language on which frame, rule and constraint generators operate to build up the final knowledge bases.
The overall knowledge-acquisition process consists of three levels:
- knowledge elicitation
- intermediate knowledge representation
- knowledge-base generation.

Methods for knowledge elicitation:
1.Interviewing techniques for rule acquisition (Grover 1983):
- forward scenario simulation

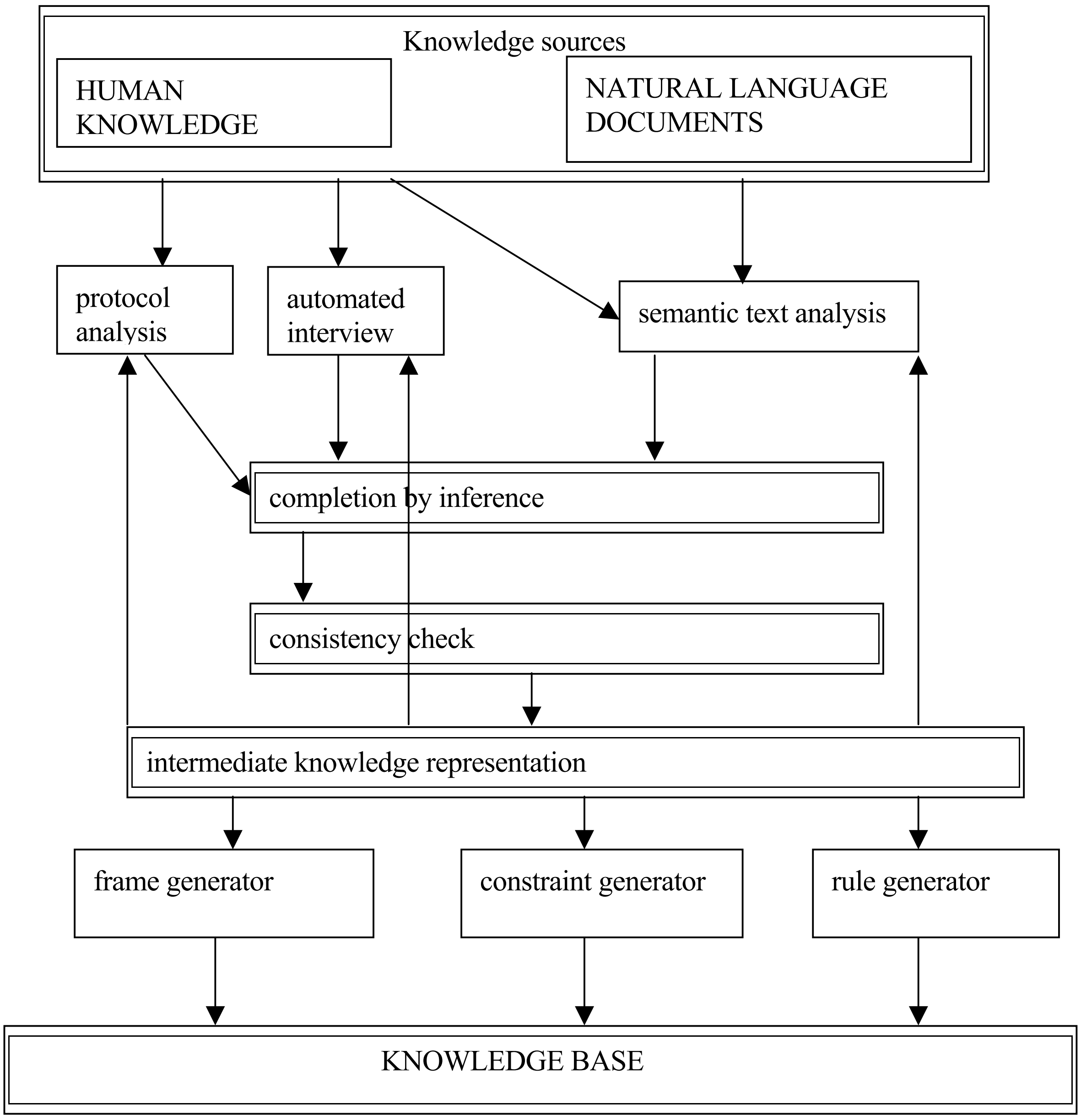
- goal decomposition
- procedural simulation (= protocol analysis)
- pure reclassification
- laddering (not mentioned in Grover)

Interviewing technique is the repertory grid approach (Kelly 1955, see section 3.6): triples of semantic related concepts are presented and the expert is asked to ascribe attributes to distinguish two concepts from a third.

2. Protocol analysis. Analysis of thinking-aloud protocols. It is still difficult to analyze verbal data.

3. Text analysis according to context analysis.

**********************************************************************



****************** the KRITON architecture ********************************
figure 3.1

Utilization of the acquired knowledge depends on the quality of the existing knowledge:
- guidance of the acquisition process through discovery of incompleteness.
- completion of domain-dependent deep models.
- employment as an Interpretation Model for the discovery of new situations (see Breuker and Wielinga 1985).

The intermediate knowledge-representation level works as a blackboard for frame, rule and constraint generation.

The purposes are:
- openness for extensions
- knowledge acquisition for different knowledge-representation tools
- storage of incomplete knowledge for the ongoing elicitation process
- integration and employment of acquisition knowledge bases
- maintaining information closer to the sources (e.g., expert utterances)
- management of knowledge bases with varying degrees of completeness in different knowledge representation languages.

Knowledge-guided knowledge elicitation deals with incomplete knowledge. A "watcher" plays a significant role by controlling the intermediate knowledge representation for missing components.

Knowledge-base generation.
The task of the frame generator is to translate the information from the protocol analysis stored in structured objects and their relation into a frame language (using the BABYLON frame) (Diederich et al. 1987).

## 3.18 LEAP

A new class of knowledge-based consultant systems designed to overcome the knowledge-acquisition bottleneck has been proposed. Recently developed machine-learning methods to automate the acquisition of new rules are incorporated, in particular a LEarning APprentice system for VLSI circuit design.
One key aspect of these systems is that they are designed to continually acquire new knowledge without an explicit "training mode".
LEAP is currently being constructed as an augmentation to a knowledge-based VLSI design assistant called VEXED.

A fundamental feature of LEAP is that it embeds a learning component within an interactive problem-solving consultant. This allows it to collect training examples that are closely suited to refining its rule base. In particular, training examples collected by a Learning Apprentice have two attractive properties:
1. Training examples focus only on knowledge that is missing from the system. The need for the user to intervene in problem-solving occurs only when the system is missing knowledge relevant to the task at hand, and the resulting training examples therefore focus specifically on this missing knowledge.
2. By working with training examples that are single steps, LEAP circumvents many difficult issues of credit assignments that arise in cases where the training example corresponds to a chain of several rules.

A second significant feature of the design of LEAP is that it uses analytical methods to form general rules from specific training examples, rather than more traditional empirical, data-intensive methods.
LEAP's explain-then-generalize method, based on having an initial domain theory for constructing the explanation of the example, allows LEAP to produce justifiable generalizations from single training examples.

While analytical generalization methods offer a number of advantages, they require that the system begin with a domain theory that it can use to explain/validate the training examples. This requirement, then, constrains the kind of domain for which our approach can be used. In the domain of digital circuit design, the required domain theory corresponds to a theory for verifying the correctness of circuits. In certain other domains such a theory may be difficult to come by.

A third significant feature in the design of LEAP is the partitioning of its knowledge base into:
1. Implementation rules that characterize correct (though not necessarily preferred) circuit implementations, and
2. control knowledge for selecting the preferred implementation from among multiple legal options. This partitioning is important because it helps in dealing with the common problem that when one adds a new rule to a knowledge base one must often adjust existing rules as well (Mitchell et al. 1985).

## *3.19 MOLE*

MOLE is the successor of MORE (see section 3.21).
MOLE can help domain experts build a heuristic problem-solver by working with them to generate an initial knowledge base and then detect and remedy deficiencies in it. The problem-solving method presupposed by MOLE makes several heuristic assumptions about the world, which MOLE is able to exploit when acquiring knowledge and by allowing covering knowledge to drive the knowledge-acquisition process. MOLE is able to disambiguate an under-specified knowledge base and to interactively refine an incomplete knowledge base.

MOLE is an expert-system shell that can be used in building systems that do heuristic classification.
MOLE belongs to a family of knowledge-acquisition tools which get their power by paying close attention to the problem-solving method used by their performance systems. Examples are TEIRESIAS, ETS, MORE, KNACK, SALT and SEAR. MOLE differs from them in that its problem-solving method incorporates certain explicit assumptions about the world which, along with several assumptions about how experts express themselves, are exploited during the knowledge-acquisition process.
The goal has been to make MOLE smart, i.e., to enable it to build a reasonable knowledge base with a minimal amount of information elicited from the expert.
MOLE the knowledge-acquisition tool gets its power from its knowledge of the problem-solving method of MOLE the performance system.
MOLE's problem-solving method is a variant of heuristic classification. Central to MOLE's method is the distinction between evidence that needs to be explained or covered by some hypothesis and evidence that helps differentiate among hypotheses.
The hard problem in knowledge acquisition is eliciting the right sort of knowledge from the expert. The first step is to identify explicitly the appropriate problem-solving method for the task and the types of knowledge roles relevant for this method. From here one can go with an automated system. Next is the problem of indeterminateness: the expert tends to be vague about the nature of these associations of events. Finally one has the problem of incompleteness.
The incompleteness problem is the problem of how to identify missing or incorrect knowledge. Two

problems dominate the two phases of knowledge acquisition:
1. The gathering of information for constructing the initial knowledge base; and
2. the iterative refinement of this knowledge base.

During the first phase, MOLE mainly relies upon static techniques of analysis. MOLE examines specific associations and events in the light of the context provided by the surrounding structures. MOLE concentrates on disambiguating the information provided by the expert, although MOLE also tries to recognize areas where the expert interacts in order to refine the knowledge base.

Constructing the initial knowledge base.
MOLE initiates the knowledge-acquisition process by asking the expert to list the events, i.e., hypotheses and evidence that are commonly relevant to the expert's domain and to draw associations between pairs of events.
Additional information that is needed: type of event; the type of evidence an association provides; the direction of an association (does e1 explain e2, or vice versa); and the numeric support value attached to an association. However experts are not very good in supplying numeric support values. Fortunately, it turns out that the support values do not have to be very accurate. MOLE can assign default support values, that are just as good if not better, than those assigned by the expert.

Differentiating knowledge.
If MOLE's diagnosis does not match that supplied by the expert, MOLE first determines whether or not the diagnosis would have been reachable if the hypotheses had been differentiated differently.

Covering knowledge.
If differentiating is not the problem, MOLE looks for missing covering knowledge.

MOLE illustrates how much power a knowledge-acquisition tool can obtain from a set of domain independent heuristics about the knowledge-acquisition process and the nature of the world as it relates to diagnosis. MOLE plays the role of an experienced knowledge engineer who is able to work in conjunction with a domain expert and build a diagnostic system, even though the knowledge engineer has little or no knowledge of the domain. By interpreting its assumptions about the world in terms of explicit knowledge roles that guide heuristic classification and by exploiting a few heuristics about how domain experts are likely to express themselves, MOLE is able to extract intelligently from the expert information relevant for building a reasonable knowledge base for performing the given diagnostic task (Eshelman et al. 1987).

In Eshelman (1988) it is described how MOLE handles uncertain knowledge. MOLE is thus able to be flexible in the knowledge-acquisition process.

## *3.20 The knowledge-acquisition tool for MOLGEN*

This tool is a system which allows domain experts to enter themselves procedural knowledge into a knowledge base. The system, a stylized form of scientific English embodied within the Unit System for knowledge acquisition and representation, has been used successfully within the domain of molecular biology, i.e., the MOLGEN expert system.

Expert knowledge comes in two forms: declarative and procedural. Acquisition of declarative knowledge seems reasonably well understood. However, a significant component of expertise takes a procedural form, ranging from low-level rules for data manipulation to abstract strategies for problem-solving.
Procedural knowledge for MOLGEN are: data manipulation procedures, simulation procedures, selection heuristics, and experiment design strategies.
The MOLGEN knowledge bases have been built by the domain experts themselves. The accuracy and completeness of the knowledge bases might suffer when the expertise is channeled first through a knowledge engineer and the authority of first hand knowledge is higher esteemed.
The trick is in making the domain expert comfortable in his new mode of expression.
The description of the procedural knowledge is done in RULE Language within the Unit System (Friedland 1981).

## 3.21 MORE

MORE is a tool that assists in eliciting knowledge from domain experts. The acquired information is added to a domain model of qualitative causal relations that may hold among hypotheses, symptoms, and background conditions. After generating diagnostic rules from the domain model, MORE prompts for additional information that would allow a stronger set of diagnostic rules to be generated.
MORE's primary value lies in its understanding of what kind of knowledge is likely to be diagnostically significant. By formulating its questions in a way that focuses on such knowledge, it makes the most effective use of the domain experts' time.
MORE elicits diagnostically significant knowledge from domain experts; it is similar in spirit to systems like TEIRESIAS (see section 3.30) and ETS (see section 3.6).

MORE provides a mechanism for interviewing. MORE differs with the others in that it takes a model-theoretic approach to the acquisition of diagnostic knowledge. It uses a qualitative model of causal relations together with a theory of how causal knowledge can be used to achieve more accurate diagnostic conclusions to guide the interview process.
MORE has the capacity to build domain models from a fixed set of qualitative relations that may hold among hypotheses, symptoms and background conditions. MORE generates rules from the domain model. After a rule is constructed, the user is asked to associate positive- and negative-support values with each rule.
Once an initial knowledge base is built up, MORE looks for weaknesses in the rules it has generated. In another role, MORE looks for potential inconsistencies in the way a user has assigned confidence factors to diagnostic rules.
MORE uses strategies for improving diagnostic performance such as: differentiation, frequency conditionalization, symptom distinction, symptom conditionalization, path division, path differentiation, test differentiation, and test conditionalization.
MORE has been applied to parts of a drilling fluids domain (MUD) as well as to diagnostic problems provided by a physician. The next step is to use MORE to develop a number of knowledge-based consultation systems in a wide variety of domains (Kahn et al. 1984, 1985a, and 1985b).

MOLE (see section 3.19) is a more recent extension of MORE.

## 3.22 OPAL

An effective paradigm may be to use the semantics of the application domain itself to govern access to an expert system's knowledge base. This approach has been explored in a program called OPAL, which allows medical specialists working alone to enter and review cancer treatment plans for use by an expert system called ONCOCIN.

The approach is to provide the experts with some type of knowledge editor to aid in updating and reviewing the contents of the knowledge base.
OPAL is based on a more abstract kind of conceptual model - that of the structure of the domain itself. This structural characterization of the application area is referred to as a domain model.
The ONCOCIN knowledge base is encoded heterogeneously using three basic representations.
First, a hierarchy of frames encoded using an object-oriented language defines the various structural entities for each protocol in the knowledge base.
Production rules, the second form of knowledge representation in ONCOCIN, are linked to each object in the planning hierarchy.
The procedure-oriented knowledge is represented in finite state tables - lists of potential states in the treatment plan and the conditional transitions that define how one state may be followed by others.
Defining knowledge of a new protocol for ONCOCIN therefore requires:

1. Creating an object hierarchy describing the component elements;
2. linking appropriate production rules (and the parameters they include) to the various objects; and
3. specifying the temporal sequence of chemotherapies and radiation treatments in terms of a finite-state table.

Each of these representational issues is handled transparently by OPAL.

The conceptual model used in OPAL, based on the domain model itself, is perhaps the most categorical way in which the contents of a knowledge base can be viewed. Unlike previous knowledge-acquisition tools, OPAL's model is simply one of what knowledge should be expected. As a result, the user is not given the flexibility found in other systems to specify new concepts. The domain model tends to be sufficient because of the highly structured, stylized nature of oncology treatment plans.

The goal in OPAL is to maximize the knowledge that experts can enter independently by providing a conceptual model that matches the way oncologists seem to think about the application area. The model then serves as the basis for a visual language that makes it easy for experts to express ideas relevant to their domain (Musen et al. 1987).

## 3.23 PROTOS

The major contribution of this research is a theory of the acquisition and application of domain knowledge for heuristic classification. The goal of building PROTOS, an inquisitive learner which evolves into an expert, forces a thorough analysis of three fundamental issues in concepts formation.
First, how are ill-defined, "fuzzy" concepts learned and represented? Induction is not believed to be the primary learning mechanism since classical and probabilistic representations are inappropriate

for most concepts. PROTOS adopts an exemplar-based representation to support the inherent variation in natural concepts.
Second, what functions must learned knowledge support? Traditionally the task of object classification has been supported to the exclusion of other tasks. The range of functions that should be supported by concept formation includes summarization of training instances, generation of examples of a concept, prediction of unseen features of a new object, interpretation of "fuzzy" examples, and explanation. PROTOS supports these important applications of learned knowledge by de-emphasizing the role of generalization in concept formation.
Third, what is the role of teacher-supplied explanations in the learning process? Learning from classified examples alone is an artificially difficult task. This research examines broadening the channel of communication between the teacher and the learner to include explanations of the examples. This reduces reliance on large training sets and allows the construction of a domain theory.

PROTOS interacts with a human expert to elicit knowledge. PROTOS then independently applies this knowledge to perform the expert task.

PROTOS learns by attempting to solve problems posed by a domain expert; focused interaction with the expert uncovers the causes of problem-solving failures and guides learning.
The problem-solving task for PROTOS is classification. Clancey (1985) defines heuristic classification to be the use of non-hierarchical, uncertain inference application of domain knowledge for heuristic classification.

Research on concept formation usually makes the simplifying assumption that a concept can be represented by a classical definition. A classical concept definition is a set of necessary and sufficient conditions for an object to be an instance of the concept.
Benefits from assuming a classical definition of concepts:
The first is that concept formation is reduced to induction. The second benefit is that object identification is reduced to deduction. The identification is all or none; unclear classifications are not considered because of the restrictive nature of classical concept definitions. Unfortunately, classical concept definitions only work in artificial domains.
The shortcomings are:
1. The defining features of most natural concepts cannot be enumerated.
2. Classification of some objects is unclear.
3. Many concepts are disjunctive.
4. There are variations in typicality among instances of a concept.

Rather than attempting to describe category members using necessary and sufficient features, a probabilistic representation uses weighted features and a threshold for identification. For example, a vehicle might be represented as:
(engine(.5), steering wheel(.3), pedals(.4), handlebars(.2)).

Classification of a new object is performed by finding the exemplars in the knowledge base which match it most closely and assigning the new object to the same category. Explanation of a classification is facilitated by reporting a similar, known exemplar. Prediction of unseen features of an object is based on feature correlations in the closest matching exemplar(s). Example generation simply involves exemplar retrieval (perhaps ordered by typicality).

PROTOS1 is an implemented system which demonstrates knowledge-based pattern matching to support the identification task. PROTOS1 learns and uses exemplars as models for guiding identification.
PROTOS2 indexes exemplars according to their appropriate uses as models and learns how much effort to expend on knowledge based pattern matching.
PROTOS1 learns the requisite knowledge and provides a richer language for explanations (Porter and Bareiss 1986, Bareiss et al. 1988).

Further studies:
The explanation language by learning the semantics of relations is expanded. Each domain requires a special vocabulary of relations for use in explanations.
Another issue is learning event sequences. Event sequences direct the acquisition of data for object descriptions (Porter et al. 1986).

## 3.24 The knowledge-acquisition system for REX

Knowledge-based knowledge acquisition means restricting the domain of knowledge that can be acquired and developing a conceptual model of the domain. This system is for the domain of data analysis.
REX is a consultation program in regression analysis, a statistical technique for data analysis.
Knowledge-based knowledge acquisition in this context means specifying how the contents of each slot will be acquired.
The preponderance of cases was handled by interviewing.
With knowledge-based knowledge acquisition, the statistician is encouraged to think of optional inputs at the beginning of the construction process, thus avoiding the costs of reprogramming.
A conceptual framework is necessary. In building a first, ground-level, system it will help to seek regularity and common cases. A frame based programming system helps to identify these commonalities.
The framework must be readily presentable. The subject matter specialists may need to be encouraged to think within the specific framework provided, even if it is natural (Gale 1987).

## 3.25 ROGET

ROGET helps a domain expert perform several critical design tasks during the early phases of knowledge-base design. The most important of these tasks is the design of the conceptual structure of the target consultation system, a description of the kinds of domain-specific inferences that the consultant will perform and the facts that will support these inferences. Finally the conceptual structure should fit into the EMYCIN system.
The initial dialogue that ROGET conducts with the expert to acquire the conceptual structure is based on abstract categories that are independent from a particular domain.
The representation of a problem can be viewed as having two primary components, one structural and the other inferential.
The expert system must capture both components: the vocabulary and the rules.

ROGET claims that it assists in the proper selection of the domain terms and the organization of the knowledge.
A comparison of several expert systems designed for diagnostic consultation shows a similarity in the kinds of concepts that these systems employ and that they are essentially the same.
Each kind of diagnostic problem-solving task has an associated subset of abstract categories. After identifying the primary problem-solving task, ROGET interacts with the expert to identify a set of domain-specific counterparts to these categories. These terms form a skeletal design for the new conceptual structure.
The advice that ROGET can provide is based on the classification of the type of diagnostic problem-solving tasks the new expert system will perform. This classification serves as the basis for strong expectations about the types of expertise the expert will need to identify and represent. The knowledge about problem types and the general structure of expert system that solve these problems provides ROGET with a domain-independent method of understanding the specific expertise in a new application domain and acquiring a conceptual structure for that domain.

A ROGET consultation consists of four major steps:
1. Determine the problem-solving task type of the expert system.
2. Acquire the conceptual structure for the system.
3. Rescope the conceptual structure.
4. Reformulate the conceptual structure for a particular knowledge-engineering tool.

ROGET contains an enumeration of several diagnostic problem-solving tasks that an expert system might perform. ROGET maintains an association between the names of different, well-known expert systems and their primary problem-solving task type.
One other method that ROGET employs to determine the problem task type is to provide the expert with examples of previous expert systems.
After determining the primary problem-solving task for the new consultant, ROGET begins the acquisition of the conceptual structure. First, the problem task type identified at the beginning of the consultation suggests an initial set of tasks and goals that might be applicable to this kind of consultant. Then the conceptual structure is elaborated by the selection of catagories of advice and evidence that the expert and ROGET determine are applicable and appropriate for this application. The initial skeletal configuration can be thought of as made up of generic conceptual structures that are suggested by ROGET and that are specialized by the expert.

The design of ROGET is based on the assumption that domain-specific examples from other applications, coupled with simple, descriptive phrases associated with each of the categories, are sufficient to guide the expert's choice, by analogy, of the proper domain-specific counterparts in the new application.
In the stage of pruning the conceptual structure the expert is given the opportunity to check each instance in the conceptual structure.
The final step in a ROGET consultation is the conversion of the conceptual structure into a form suitable for operation with the system building tool.
At present, ROGET is only able to perform this conversion for expert systems in the EMYCIN system (Bennett 1985).

## 3.26 RuleMaster

RuleMaster is a knowledge system application generator.
RuleMaster appears to be better suited to classification than to construction problems, although it is claimed to be useful for both. It provides a language for representing reasoning patterns, but provides no assistance to the knowledge engineer in identifying and acquiring those patterns. It is based on induction rules (see section 2.11) (Kornell 1987).

The two principal components of RuleMaster2 are RuleMaker, an automatic rule generator, and Radial, a block-structured rule language. Two forms that knowledge can take are declarative and procedural. Knowledge can be entered either in example format or directly in rules.
The RuleMaker feature automatically induces rules from sets of examples supplied by the expert. In this way, declarative knowledge is easily integrated into the knowledge base.
Knowledge that already exists in a procedural form can be entered directly in a rule format via the Radial language.

RuleMaster2 is commercially available. It is developed by Radian Corp. and can be used on IBM-PC or compatible[10] (RuleMaster2 1987).

## 3.27 SALT

SALT (kNowledge ACquisition Language) is a knowledge-acquisition tool for generating expert systems that can use a propose-and revise problem-solving strategy.
So far little attention has been paid to automated knowledge acquisition for systems that solve problems by constructing solutions.
SALT was developed as a knowledge-acquisition tool for VT, an elevator system configurer. The input to the configurer was to include functional requirements for the completed configuration, preferences for specific parts and a description of the spatial structure within which the configured system must fit. The system's output was to consist of quantities, descriptions and model numbers of parts selected and a specification of spatial relationships among parts and between parts and structural landmarks (Marcus et al. 1985).

The system will start incrementally constructing a design by proposing values for design parameters. The system will also identify constraints on design parameters. Whenever it detects a constraint violation, the system will use domain expertise to consider past decisions that could be revised, choose the most preferred revision that remedies the violation, remove anything potentially inconsistent with that change, and continue extending the design from that point.
At the start of a SALT interview, the user is shown the menu for indicating the type of knowledge to be entered or viewed. Three basic kinds of knowledge make up a propose-and-revise system:
1. Procedures for proposing values for the pieces of the design the system will output.
2. Identification of constraints on individual pieces of the design.
3. Suggestions for ways of revising the design if the constraints are not met.
Once the user enters these procedures, SALT stores that knowledge within a dependency network.

[10] RuleMaster, RuleMaker, Radial, and IBM-PC are trademarks.

Conclusion
SALT makes a strong commitment to the nature of the problem-solving strategy that will be used for any task it will acquire. This allows SALT to represent domain knowledge according to the role it will play in finding a solution for any task that can use this basic strategy. This commitment gives SALT considerable power in guiding its interrogation of domain experts in the area where they most need guidance - in making decisions that require consideration of the potential interactions of a single piece of knowledge with everything else in the knowledge base.

SALT currently understands only a few variations of a problem-solving strategy. The ideal knowledge-acquisition tool would be a true knowledge-engineering expert that understands a large range of AI techniques. A research strategy that makes progress toward a goal of developing such a tool is the one followed so far for SALT:

1. Focus the knowledge-acquisition tool on the problem-solving strategy that will be used by the system it creates for one domain.
2. Try the tool on another domain for which the problem-solving strategy looks promising.
3. When the problem-solving strategy breaks down, identify characteristics of the domain that made it break. This task will be tractable if the knowledge-acquisition tool makes explicit what the problem-solving strategy is and how knowledge is used by the strategy.
4. Automate the analysis of the knowledge base to diagnose the breakdown and treat it (Marcus 1987).

In Stout et al. (1988) a more general use of SALT is described.

## *3.28 SEAR*

R1 is an expert system to configure computer systems. It contains a very large amount of knowledge. SEAR is a knowledge-acquisition tool for R1.
Knowledge acquisition tools like TEIRESIAS, ETS and MORE are tools for classification problem-solvers. SALT and SEAR are different. The problem is how a problem-solving method can influence the development of a knowledge acquisition tool.
Knowledge has been represented in R1 in various ways; regulari-ties, to the extent they exist, have gone unnoticed. One has to know R1 well to modify its behavior in some desired fashion.
The primary problem-solving method used by Rime has been derived from work done on R1-SOAR - an experiment in knowledge intensive programming using a general problem architecture called SOAR
(see Rosenbloom et al. 1986).

SOAR can create new productions, or chunks, based on the results of its goal-based problem solving and then use these chunks to speed up its performance on subsequent goals. Because each new chunk is logically entailed by the pre-existing knowledge base, this technique has been considered symbol-level learning. It can also use this chunking mechanism to learn at the knowledge level, that is, to acquire new knowledge (Greiner et al. 1988).
It is easier to add knowledge to Rime. The problem-solving method used by Rime provides more direction to someone adding knowledge than does R1's method. This is in part because Rime's

method integrates explicitly defined knowledge roles and in part because the person adding the knowledge can specify the conditions under which one piece of knowledge should be applied in preference to another.

SEAR as a knowledge collector and organizer.
The problem of knowledge acquisition can be more appropriately viewed as the problem of knowledge maintenance. After eliciting the knowledge from a domain expert the knowledge is put into an intermediate representation. This is a storehouse of domain knowledge in a declarative form. The SEAR rule generator converts the intermediate representation into OPS5 rules that "proceduralize" the knowledge; that is, the knowledge is represented in a way that tailors its usefulness to a problem-solving method so that there is no need to search the knowledge base when solving a problem.
In SOAR, expertise can be added to a base system either by hand crafting a set of expertise-level rules or by automatic acquisition of knowledge. Automatic acquisition of new rules is accomplished by chunking, a mechanism that has been shown to provide a model of human practice, but is extended here to much broader types of learning (Van de Brug 1986, Rosenbloom et al. 1984).

## *3.29 TDE*

The TEST Development Environment (TDE) enables knowledge engineers and trained domain experts to interactively build knowledge bases representing troubleshooting knowledge. TEST is an application shell.

Expert-systems developers typically find that their knowledge-acquisition techniques change with the course of system development. In first approaching a new problem, knowledge acquisition tends to be exploratory. The goal is not only to acquire knowledge but, more important, to identify a representational format and control strategy of sufficient power to capture domain-specific problem-solving behavior and domain-specific knowledge. Once this is done, knowledge acquisition becomes constrained by the target architecture.
Diminishing the knowledge-acquisition bottleneck thus requires two interlocked solutions. First, a packaged problem-solving architecture which allows developers to focus on knowledge acquisition, rather than on knowledge-base design and problem-solving control, and issues of typically greater complexity. Second, a high-productivity workbench aimed at reducing the time it takes to build and maintain knowledge bases.
In this vein, TEST provides an application shell for troubleshooting systems, while TDE provides the high-productivity workbench domain experts use to build and maintain knowledge bases.
TDE continues a line of previous systems, including ROGET, MORE, SALT. TDE differs in two ways: first, it uses a problem-solving strategy that is more comprehensible to domain experts in the manufacturing and customers service domains. Second, TDE addresses the need for knowledge-acquisition systems to conform to a developer's desire to provide information as it comes to mind.

The problem-solving architecture.
Knowledge acquisition is largely a matter of mapping the knowledge which supports expert decision-making into the representations required by a problem-solving system. When there is a conceptual correspondence between these representational units and the terms with which experts understand their task and domain, mapping becomes a straightforward operation. In addition,

conceptual correspondence makes direct-manipulation techniques readily available, and permits the system to easily guide users engaged in knowledge-base development.
TEST uses semantic networks of schematic objects, or frames, to represent its key concepts.
The failure-mode represents a deviation of the unit under test from its standard of correct performance. Within TEST, the failure-mode prototype describes the characteristics which must be provided for each failure-mode instance, such as a broken cooling unit.
Other prototype objects within TEST include questions, tests, test-procedures, repair-procedures, rules, decision-nodes, and parts.
Since the failure-mode is the key concept in most trouble-shooting tasks, such aggregates provide an easily understood and readily accessible structure.
The search space can be dynamically altered by rules sensitive to information acquired during a diagnostic session.

By providing a mixed-initiative multi-task environment, TDE allows knowledge engineers and domain experts the option of directing or being directed by the underlying knowledge-acquisition system. By providing a graphic interpretation of the underlying knowledge base as it changes with respect to user input, users are provided with a context in which to understand the impact of replies to system prompts.
TDE supports knowledge-based enhancement by providing tools that support knowledge-based browsing, modification and debugging.
The TEST architecture greatly aided the development of the TDE workbench. In particular, the choice of a schematic as opposed to rule-based representation led to a knowledge base characterized by the use of domain-familiar concepts, and sufficient conceptual structure to facilitate several TDE features (Kahn et al. 1987).

## *3.30 TEIRESIAS*

TEIRESIAS is a program designed to function as an assistant in the task of building large knowledge-based systems. It embodies a particular model of interactive transfer of knowledge from a human expert to the system, and makes possible knowledge transfer in a high level dialog conducted in a restricted subset of natural language.
Of the major problems, the weakness of the natural-language-understanding techniques presents the largest barrier to better performance.
Knowledge acquisition in the context of a shortcoming in the knowledge base, for instance, has proved to be a useful technique for achieving transfer of expertise, offering advantages to both the expert and TEIRESIAS (Davis 1977).

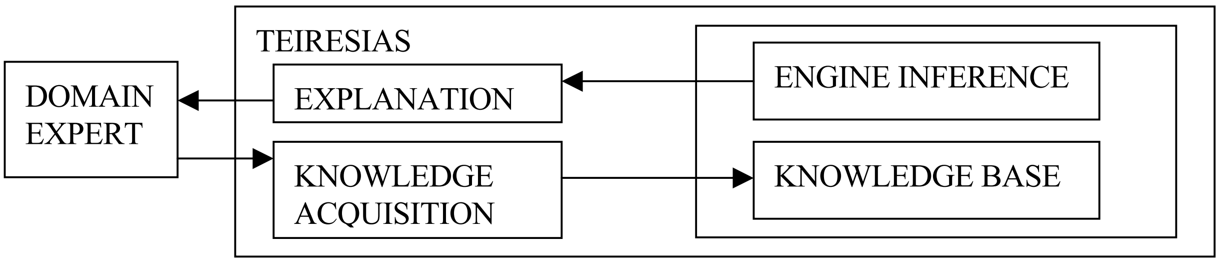
Two major goals were used as guidelines in creating a set of tools for the construction, maintenance, and use of large, domain specific knowledge bases. First, it should be possible for an expert in the domain of application to "educate" the performance program interactively, commenting on and correcting its behavior. Second, it should be possible for the expert to assemble and maintain a large body of knowledge.

The central theme of the development of TEIRESIAS is the exploration and use of what is labeled meta-level knowledge. This concept is about "knowing what you know".

MYCIN provided the context in which TEIRESIAS was developed. MYCIN was designed to provide consultative advice on diagnosis and therapy for infectious diseases.
TEIRESIAS is written in INTERLISP, an advanced dialect of LISP.

There are two major forms of knowledge representation in use in the performance program:
1. The attributes, objects, and values which form a vocabulary of domain-specific conceptual primitives.
2. The inference rules expressed in terms of these primitives.

*******************************************************************************



**************************The TEIRESIAS architecture **************************
figure 3.2

There are, correspondingly, two forms of knowledge acquisition: 1. The acquisition of new primitives - to expand the performance program's vocabulary of concepts.
2. The acquisition of new rules expressed in terms of existing primitives.

Knowledge levels in knowledge-acquisition routines.
The first level of the hierarchy contains the object-level knowledge - medical knowledge of cultures, organisms, drugs, etc. High performance on the task of diagnosis and therapy selection is supported by an extensive collection of knowledge about objects in the medical domain.
The next level is concerned with the conceptual building blocks of the knowledge representation - the predicate functions, attributes, values, rules, and so on.
Knowledge at the third level is concerned with the conceptual primitives behind representations in general. The second-order system can be used to acquire knowledge about a representation.

The most general technique involves having TEIRESIAS directly examine the rules in the knowledge base, as in the use of the templates to determine whether a premise clause has already been established or is still untested. In doing this, TEIRESIAS's explanation facility examines and interprets the same piece of code that the performance program is about to execute. The resulting explanation is thus constructed with reference to the content of the rule, and this referral is guided by information (the templates) contained in the rule components themselves.

The interaction between the domain expert and the performance program is viewed as "interactive transfer of expertise". This can be seen in terms of a teacher who continually challenges a student with new problems to solve and carefully observes the student's performance. The teacher may interrupt to request a justification of some particular step the student has taken in solving the

problem or may challenge the final result. This process may uncover a fault in the student's knowledge of the subject (the debugging phase) and result in the transfer of information to correct it (the knowledge acquisition phase).
Knowledge acquisition is emphasized in the context of shortcomings in the knowledge base.

The schemata are the primary vehicles for describing representations. They were developed as a generalization of the concept of record structures and strongly resemble them in both organization and use.

Levels of knowledge and schemata.

0. The base of domain-specific knowledge consists of the collection of all instances of each representation.
1. The base of representation-specific knowledge consists of the schemata, which are, in effect, the declarations of the extended data types. These have a degree of domain independence since they describe what an attribute is, what a value is, etc., without requiring a priori knowledge of the domain in which those descriptions will be instantiated.
2. The base of representation-independent knowledge - the schema-schema - describes what a declaration looks like. At this level resides knowledge about representations in general and about the process of specifying them via declarations.

The inability to deal with more complex interrelationships of representations is currently the system's primary shortcoming.
The knowledge-acquisition capabilities of the schemata offer a very organized and thorough assistance that can:
- Attend to many routine details. Some of these are details of data structure management, and having the system attend to them means the expert need to know nothing about programming.
- Show how knowledge should be specified.
- Make sure that the user is reminded of all the items he has to supply.

Summary
Knowledge acquisition was described as a process of interactive transfer of expertise from an expert to a performance program, in which TEIRESIAS's task was to "listen" as attentively and intel-ligently as possible. The process was set in the context of a shortcoming in the knowledge base, as an aid for the expert. He is faced with a specific consultation whose results he finds incorrect and has available to him a set of tools that will allow him to uncover the extent of the system's knowledge and the rationale behind his performance. His task is then to specify the particular difference between the system's knowledge and his own that accounts for the discrepancy in results (Davis and Lenat 1982).

TEIRESIAS aids a human expert in monitoring the performance of a knowledge-based system. When the human expert spots an error in the program's performance, in either the program's conclusions or its line of reasoning, TEIRESIAS assists in finding the source of the error in the database by explaining the program's conclusions - retracing the reasoning steps until the faulty or missing rule is identified. At this point, TEIRESIAS assists in knowledge acquisition, modifying faulty rules or adding new rules to the database.
Meta-level knowledge about the kinds of rules and concepts in the database is applied to build expectations in TEIRESIAS's model-based understanding process.

Meta-level knowledge is also used to encode problem-solving strategies, in particular, to order the invocation of rules so that those that are most likely to be useful are tried first (Barr, Cohen and Feigenbaum 1981).

## 3.31 TIMM

The expert system TIMM/Tuner has been developed to tune VAX's[11].
Two features of the development of this system merit the attention of artificial intelligence application engineers. The first is a simple and rapid automated knowledge-acquisition strategy. The second is an atypical inference procedure which maintains the strength of standard forward chaining while providing for program sensitivity to gradation of meaning within rule clauses.

To approach the problem of automated knowledge acquisition, TIMM separates knowledge into declarative and procedural sections, much like frame-based systems which allow attached procedures.
The declarative foundation allows for an action-oriented approach to the development of procedural knowledge. The action-oriented approach is based on the belief that most experts are better at doing than at describing. TIMM uses the domain description in the declarative knowledge section to synthesize realistic example problems.
These are the basis of an interactive dialog with the expert, oriented toward making decisions (doing), rather than explaining how decisions are made (describing).

TIMM greatly reduced the need for knowledge engineering assistance in building TIMM/Tuner.
Some need for knowledge engineering principles remain.
TIMM has low cost for turning around or starting over in the early stages of development (Kornell 1984).
TIMM is commercially developed by General Research Corp. for use on an IBM-PC[12] or compatible.

## 3.32 OTHER SYSTEMS

It is not feasible in the scope of this paper to give a complete overview of all the systems that possibly exist. In the preceding sections only the most important and often the best documented systems are described. In the following list I enumerate several systems that are worth mentioning,

[11]TIMM and TIMM/Tuner are trademarks.

[12]IBM-PC is a trademark.

although little literature about them is available.

The Advice Taker/Inquirer (AT/I) is a domain-independent program that is used to construct, monitor, and improve an expert system.
In the learning phase, the Advisor teaches a strategy to the AT/I by providing it with general principles, specific examples, and assertions. Advice consists of a description of a situation and a recommended set of actions (possibly tagged with a certainty factor) to be performed. The system requests the expert for clarification whenever his advice is inconsistent, incomplete, or vague. During execution of the expert system, the AT/I enters the operational phase, in which it monitors and suggests improvements to the expert's strategy by analyzing its performance with respect to the current environmental situation and previous experience. Weaknesses in the strategy are detected and corrected automatically by the AT/I whenever possible (Cromp 1985).

The CONSUL system is an expert system that supports interactions between users and online services. CONSUL is designed as a general framework in which a wide variety of services can be embedded. One component of the CONSUL system is made to acquire the necessary knowledge in a semiautomatic way (Wilczynski 1981).

EXPERT is a system for designing and building models for consultation. EXPERT is an expert-system shell rather than a knowledge-acquisition tool. The EXPERT system allows for efficient testing of changes due to particular knowledge-base revisions.
Part of the program evaluates the current rules according to how many correct and incorrect problem solutions they are involved in over a large set of test cases. This program finds weak rules in the knowledge base (Weiss and Kulikowski 1979, Hayes-Roth et al. 1986).

EXPERT-EASE is a commercial version of ID3, which is an algorithm that can make decision trees with positive and negative instances of a concept (Michie 1987).

HERACLES is an expert-system shell for solving heuristic-classification problems and ODYSSEUS is an apprenticeship learning program. Both are used in combination with the expert systems of the MYCIN family (Wilkins et al. 1987).

MUM (Manage Uncertainty in Medicine) is an expert system that combines knowledge in order to avoid uncertain conclusions, given uncertain knowledge. It is also able to make decisions in diagnosis, based on control rules (Gruber and Cohen 1987).

SEEK is an interactive system that provides a unified framework for designing and testing expert models, and is applied to the development of a diagnostic consultation system in rheumatology (Politakis 1985).

# CHAPTER 4

# AUTOMATED KNOWLEDGE-ACQUISITION SYSTEMS II: EVALUATION

In this chapter the automated knowledge-acquisition systems are evaluated in terms of their formal structure. It is impossible to assess these systems for their quality only using their descriptions. Comparing these systems in action would give a substantial evaluation of their performances, but that is not within the scope of this paper. Moreover, evaluations of systems are rarely found in the literature.

The main conclusion that can be drawn from the previous chapter is that automated knowledge acquisition is not a univocal choir of singers, but it is easy to distinguish the various systems.

## *4.1 DEDICATED SYSTEMS*

The most obvious distinction that can be found is between automated knowledge-acquisition systems that are specially designed for a certain expert system and the general-purpose systems. This distinction can also be called dependent versus independent automated knowledge-acquisition systems.

The specially dedicated systems are Meta-DENDRAL, INFORM, KAE, KAS, KNACK, LEAP, the automated knowledge-acquisition system for MOLGEN, OPAL, the automated knowledge-acquisition system for REX, ROGET, SALT, SEAR, TEIRESIAS, and TIMM.
The essence of these systems is that they use the same structure(s) as the expert system for which they are constructed. The knowledge-acquisition system can consist of a model of the domain; it can use the same sort of representation as the knowledge base uses; it can also use the inference system of the expert system; or a combination of these. We can say that the knowledge-acquisition system operates analogously to its expert system.
Thus, the dedicated systems have a great advantage over the independent systems. They do not need to acquire the domain structure which is more difficult to obtain than the data and rules that fill the slots.

Both in ROGET and TEIRESIAS the technique of the MYCIN expert systems is very explicitly present.
TEIRESIAS exploits meta-level knowledge, i.e., knowledge about the knowledge base of the expert systems. With this knowledge TEIRESIAS is able to trace shortcomings in the knowledge base, but also to explain the conclusions of the program. In its capacity as a knowledge-acquisition tool, TEIRESIAS is able to communicate between the knowledge base and the domain expert.
ROGET is also able to acquire knowledge for expert systems of the MYCIN family. In the initial stage of knowledge acquisition ROGET method is to provide the domain expert examples from other existing expert systems. Thus the conceptual structure of the new expert system is chosen from an existing system, according to the task for which it will be used. ROGET operates with analogy.

TEIRESIAS and ROGET are dedicated to MYCIN-like expert systems but not only to one system.

The following knowledge-acquisition systems are specially built to facilitate the expansion of one particular expert system:

Meta-DENDRAL works only for DENDRAL. It uses in its knowledge-acquisition process a semantic model of the DENDRAL domain.

KAS, the knowledge-acquisition system for PROSPECTOR, knows the representation structure of the main system. A semantic network matcher is used to compare the input by the expert and the system. KAS also contains the PROSPECTOR inference engine.

KNACK uses the problem solving methods and knowledge roles of the expert systems it can build.

LEAP has an initial domain theory that enables it to derive generalizations from examples.

OPAL uses a domain model of ONCOCIN for which it acquires knowledge. The domain model is constructed in such way that it matches the course of thinking that oncologists follow in their field.

The knowledge-acquisition system for REX contains a conceptual framework of the prototyped expert system. This enables the expert to formulate his knowledge in terms of the knowledge base of the expert system.

SALT is the knowledge-acquisition system for VT which is an expert system that can configure elevator systems.
SEAR is the knowledge-acquisition system for R1, a computer system configurer. Both use the problem-solving strategies of their expert system to acquire domain knowledge.

TIMM uses the domain description to conduct a dialog with the domain expert.

TDE is not dedicated to a particular expert system, but is meant for building expert systems that utilize trouble shooting knowledge. TDE uses semantic networks of schematic objects or frames. INFORM operates in the same way.

KAE is typical for acquiring knowledge about interpreting aerial imagery. In this list of automated knowledge-acquisition systems KAE is the only system that is built for solving visually related problems.

The knowledge-acquisition system for MOLGEN is somewhat different since it is not a separate system. It provides a direct access to the knowledge base by ways of a stylized form of natural language. MOLGEN's knowledge base is built by domain experts themselves using declarative and procedural knowledge. Since they know the structure of the system there is no need for a separate construction of the knowledge-acquisition system.

## *4.2 STAND-ALONE SYSTEMS*

The automated knowledge-acquisition systems that are not dedicated to certain expert systems have their own methods of acquiring knowledge.

**Repertory grid method**
Most common is the repertory grid method. This is used in AQUINAS, its predecessor ETS, KITTEN, KRITON (and its predecessor PLANET).
In the repertory grid method the expert is asked to list, compare and rate series of items. The system is able to build hierarchical and relational structures from these ratings. This method serves well in areas with classification and diagnosis problems. Design and planning areas that are built up from components rather than broken down, are not suited for the repertory grid method.
Repertory grid is only one of the components of KRITON's method. The combination with other methods makes KRITON more powerful than AQUINAS and the others.
Goodall (1985) points out that repertory grid analysis is time consuming if there are more than 6 domain objects.

**Structured knowledge acquisition**
Structured knowledge acquisition is the knowledge-acquisition approach used in KADS. It serves as an example in the European versus American controversy about whether expert-system building should be done by careful planning or by rapid prototyping.
KADS uses a library of standard models of prototypical domains. This choice gives the advantages of the dedicated systems. Analysis of the elicited data from the expert knowledge leads to a conceptual model of the system to be built. This interpretation model can collect and interpret data from the expert.
The drawbacks of this system is the time-consuming preparation before the actual expert system building process. The risk that the structured plan does not serve well enough the purpose is too high to implement in an early stage. Often this shows in a much later phase of the expert system development and thus causes waste of time and effort. On the contrary, in rapid prototyping wrong directions can easily be corrected in an early stage. On the other hand a well structured plan in advance leads towards a more consistent and complete built knowledge base.
The more examples of KADS are developed the more structures are available to choose from. Thus it becomes similar to a multi-dedicated automated knowledge-acquisition system.
A difficulty with KADS seems to be that the interpretation and structuring process start working after the verbal data have been submitted. The interpretation model for interpreting verbal protocols gives room for misinterpretation. It is an extra burden to the system.  While on the other hand structured interviewing by the system, using the structured outline of the knowledge base, avoids an extra phase in the knowledge-acquisition process.

## *4.3 MACHINE-LEARNING METHODS*

Several automated knowledge-acquisition principles are not related to the fact that the system is independent or dependent of an existing expert system or family of systems.

A promising technique in automated knowledge acquisition is the method of machine learning. The essence of machine learning is that the system can generate rules from given samples. (For more details and possibilities see section 2.11.)

KLAUS and its derivatives use the machine-learning technique that is known as "learning by being told".
It has the capability to process natural language in various ways (including recognizing patterns and analogy). Its pilot system, NANOKLAUS contains seed concepts such as classes of things and relations. The input consists of concepts and vocabulary of the domain. KLAUS is able to transfer these concepts into different knowledge representation systems.

LEAP is a dedicated automated knowledge-acquisition system. It collects training examples to make generalizations, but it uses also the domain theory of the expert system to validate the examples.

PROTOS learns by asking the expert about facts and rules and the explanation of them. Then it is able to classify them and to detect missing information.

RuleMaster and EXPERT-EASE are systems that use algorithms to induce rules from given examples.

BLIP learns from the expert to specify a "sloppy" domain model by asking about definitions of the concepts. There is explicitly no structure involved. This allows the expert and the system to be as flexible as possible.

The knowledge acquisition system Meta-DENDRAL is an example of learning by example. It uses a huge quantity of examples of molecular structures (DENDRAL determines molecular structures) to generate rules. These rules are added to the domain model. Meta-DENDRAL has proven to be able to discover new rules.

When domains get bigger and more complex, experts become unable to explain how they operate. However, they can still supply the knowledge engineer with suitable examples of problems and solutions. Using rule-induction will allow expert systems to be used in these more complicated fields. On the other hand, rule-induction programs do not help select the attribute, or do not help discover that two attributes are functionally or causally related (Goodall 1985).

## *4.4 NATURAL-LANGUAGE PROCESSING*

An approach that is used in several systems is natural-language processing. The reason why research is done in this direction is that domain experts and end-users are not accustomed to working with formalisms. To ease this problem the use of natural language in the dialog with the domain expert is proposed.

Natural-language processing is only part of the knowledge-acquisition process. However, in some European publications on knowledge acquisition, the main focus is on natural language, assuming that the problem of knowledge acquisition would be solved as long as the system could deduct rules from sentences in natural language.

Some, like ALICE (Fum 1985), plan to be a full automated knowledge-acquisition system that models the cognitive processes that occur in humans when they learn descriptive texts and are able to reason about it.

Automated language understanding is still a difficult subject. In this stage of automated knowledge-acquisition research, it seems only to narrow the knowledge-acquisition bottleneck when extra problems such as language processing are part of the system. In TEIRESIAS natural language is also a barrier in its performance.

The systems that use natural language as part of their system are doing this to ease the dialog with the domain expert.

ASTEK, like OPAL, uses natural language in the knowledge-acquisition process. The automated knowledge acquisition is regarded as an extension of the dialog between a human expert and the knowledge representation of the expert system. A fundamental part of this approach is that ASTEK is guided by a model of the domain knowledge by specifying the types of knowledge structures.

In KITTEN the analysis of natural language is integrated with the repertory grid method. It can construct a prototype with statements in natural language. This has been proven to stimulate experts to submit knowledge from different perspectives.

KLAUS is able to process natural language in a machine-learning system. The latest literature on KLAUS dates from 1983 (Grosz and Stickel) in which it was mentioned that more emphasis on natural language was stressed.

## *4.5 OTHER TYPES OF KNOWLEDGE ACQUISITION*

**Data analysis**
As mentioned before, KADS needs data analysis to convert the verbal data of protocols into data and rules for the knowledge base. Actually it is much more than conversion. The verbal data should map into the knowledge structure. For this purpose KADS uses a hierarchy of levels of knowledge.

KRITON employs a set of different methods for different aspects of the knowledge-acquisition process. For the interpretation of verbal data it also uses data analysis. The authors admit that data analysis is still a difficult task.

KEATS is a tool that helps the knowledge engineer to organize the raw data of the protocols. (KEATS is not a genuine automated knowledge-acquisition system.) KEATS has a frame-based knowledge representation and uses a graphic interface that functions as a blackboard and controls the knowledge base. On this basis it is able to analyze the protocols.

**Mixed approaches**
Several systems are combinations of automated knowledge-acquisition subsystems.

KRITON is a typical mixed approach automated knowledge-acquisition system. Expert knowledge is obtained by automated interview methods (such as repertory grid) and protocol analysis. Textbook knowledge is gathered by text-analysis techniques. Then the acquired knowledge is represented in an intermediate knowledge representation. This serves as a blackboard to generate the desired representation of the knowledge base and check the knowledge for consistency and completeness.

**Other systems**
MORE is a stand alone system, but only used for diagnostic problem-solving expert systems. Its knowledge-acquisition process is based on a domain model. Using the pattern of relations between hypotheses and symptoms MORE is able to build a domain model from which it generates rules. MORE is able to look for inconsistencies and deficiencies. MOLE, the successor of MORE, uses a heuristic problem solver in interaction with the problem-solving method of the expert system. MOLE works with domain independent (although limited to diagnosis problems) heuristics about the knowledge-acquisition process and the context of diagnosis.

$K^n{}_{A}c$ is a knowledge matcher. It is able to anticipate modifications, like incomplete knowledge, in the knowledge base. It works with heuristics about the knowledge-acquisition process like a knowledge engineer would do.

KREME, a system for knowledge acquisition and knowledge editing is able to use different kinds of knowledge representation. The goal is to develop various knowledge-acquisition approaches.

KLAUS is able to transfer its acquired knowledge into various knowledge representations.

**Knowledge acquisition from multiple experts**
KITTEN and KNACK gather knowledge from multiple experts. ETS can do the same, but the end user can evaluate the input from the various experts and can make a choice.

## *4.6 KNOWLEDGE LEVELS*

The more complicated systems need to supervise the flow of knowledge. Therefore "knowledge of knowledge" or meta-knowledge is introduced. With meta-knowledge the system knows what is going on. This metaknowledge may vary from sorts of bookkeeping or using blackboard methods. But also levels of knowledge are distinguished and used to serve the various kinds of knowledge.

INFORM distinguishes 4 conceptual levels of knowledge:
1. The diagram filling level.
2. Knowledge about the knowledge-engineering activities.
3. The heuristic approach for the encoding process.
4. The level of providing explanation.

KADS distinguishes four layers of expert knowledge, corresponding to the different roles that knowledge plays in reasoning processes:
1. The domain level contains the static knowledge of the domain.
2. The inference level applies this knowledge into rules.
3. The task level describes the goals and tasks.
4. The strategic level plans, controls, and debugs.

KADS uses 5 levels for mapping verbal data into the knowledge base:
1. Knowledge identification; the recording of verbal data.
2. Knowledge conceptualization; the formalization of the data into concepts, conceptual relations, etc.
3. Epistemological analysis; the structural properties of the conceptual knowledge are uncovered by an epistemological framework.
4. Logical analysis; this applies to the formalisms of the knowledge in the higher levels.
5. Implementational analysis; the mechanisms are uncovered on which higher levels are based.

Metalevel knowledge in TEIRESIAS is about knowing what you know. This knowledge takes care of the process of understanding the data and rules. From here it can develop strategies in the inference process.
Other knowledge levels in TEIRESIAS in the knowledge-acquisition routine:
1. Object level knowledge.
2. Knowledge of the conceptual building blocks of the knowledge representation, such as attributes, values, rules.
3. The description of the conceptual primitives behind the representations.

## *4.7 CONCLUSION*

The many examples of automated knowledge-acquisition systems in various stages of sophistication give enough reason to predict a prosperous future for this sort of system.

Dedicated knowledge acquisition systems for one particular expert system (or group of expert systems) seem to work very well. Also knowledge-acquisition systems that can handle not too complex tasks are satisfying.
But all round systems that are usable during the whole process of expert-system building and for all kinds of expert systems are not feasible. The development of such a system will take a considerable amount of manpower over a long period of time (Bennett 1987).

Many aspects of the knowledge-acquisition process are particularly apt for automation. Or, more precisely, several tasks in the knowledge-acquisition process will be performed better when they are automated than when they are done by a human knowledge engineer.
The knowledge-engineering tasks that are specially difficult for human performance should anyhow be automated. For example knowledge-base refinement, consistency checking, machine learning are specially suitable. Also all the tasks that need constant tracking and overviewing are particularly appropriate for automation.

Preliminary interviewing by the knowledge engineer seems a must to get himself acquainted with the domain. This is not so much knowledge acquisition for the expert system, but for the knowledge engineer to be able to direct the project. It does not seem relevant to automate this part of knowledge acquisition. But from here on it is possible to use automated techniques.
Concepts, their hierarchies and their relations can be acquired by scaling and repertory grid methods. Experts are good in giving advice when confronted with cases. Using the concepts learned with the

previous methods, the system can guide the dialog with the expert to elicit production rules from problem cases.
Automated refinement and debugging are already mentioned.

An automated knowledge-acquisition system that can handle the entire process is not only able to do more, but also the different stages can benefit from each other.
It is also worthwhile to generalize well-working dedicated systems or to expand multifunctional systems, such as KRITON. There is no point in reinventing the wheel.
Systems that can handle various tasks seem to be better than one task systems. Real life expert problems are seldom of one kind, but mainly mixed.
It is not unthinkable that systems that can handle different tasks are able to solve more creatively tasks than the knowledge engineer would do.

Some serious problems remain to be solved. Several knowledge-acquisition tasks are particularly difficult for human knowledge engineering. I chose the ones that are mentioned in chapter 2, the ones that I find particularly interesting.
The man-machine mismatch problem and the knowledge-representation versus knowledge-acquisition problem are the crucial ones. So far not much has been done about these problems.
Another problem is the recognition of structures in the various stages of the knowledge-acquisition process. The guidance of the knowledge-acquisition process on the system level seems solved by blackboard systems, but the guidance on a metalevel throughout the whole process is more complicated.

In a forthcoming paper I shall go more extensively into the relations between knowledge representation, knowledge acquisition and mental models in the expert-system-building process.

## BIBLIOGRAPHY

**Abrett and Burstein 1987**
Glenn Abrett and Mark H.Burstein. The KREME knowledge editing environment. International Journal of Man-Machine Studies 27 (1987)

**ACL 1985**
Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics. Geneva, Switzerland. March 1985.

**Antonelli 1983**
David R. Antonelli. The application of Artificial Intelligence to a maintenance and diagnostic information system. In: J.J. Richardson, ed. Artificial Intelligence in maintenance. In: Proceedings of the Joint Services Workshop on Artificial Intelligence in Maintenance. Boulder, CO 1983. (Noyes 1985)

**Avignon 1987**
Les systèmes experts et leurs applications. 7th International workshop. Avignon, France. May 1987. (Nanterre 1987)

**Bareiss et al. 1988**
E.Ray Bareiss, Bruce W.Porter, Craig C.Wier. Protos: an exemplar-based learning apprentice. International Journal of Man-Machine Studies 29 (1988)

**Barr, Cohen, and Feigenbaum 1981**
Avron Barr and Edward A.Feigenbaum, eds. The Handbook of artificial intelligence. Volumes I and II. Paul R.Cohen and Edward A.Feigenbaum, eds. Volume III. (William Kaufmann 1981)

**Becker and Selman 1986**
Sue Becker and Bart Selman. An overview of knowledge acquisition methods for expert systems. Tech Rep CSRI-184 June 1986. Computer Systems Research Institute, University of Toronto.

**Bennett 1985**
J.S.Bennett. ROGET: A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system. Journal of Automated Reasoning 1 (1985)

**Bennett 1987**
J.S.Bennett. Personal communication. Stanford, CA. August 1987.

**Berry 1987**
Dianne C.Berry. The problem of implicit knowledge. Expert Systems 4 (1987)

**Bobrow and Winograd 1977**
D.Bobrow and T.Winograd. An overview of KRL, a knowledge representation language. Cognitive Science 1 (1977)

**Boose 1984**
John H.Boose. Personal Construct Theory and the transfer of human expertise. In: Proceedings of the 4th conference of the AAAI. Austin, TX, 1984

**Boose 1985**
John H.Boose. A knowledge acquisition program for expert systems based on personal construct psychology. International Journal of Man-Machine Studies 23 (1985)

**Boose 1986a**
John H.Boose. Rapid acquisition and combination of knowledge from multiple experts in the same domain. Future Computing Systems 1 (1986)

**Boose 1986b**
John H.Boose. Expertise transfer for expert system design. (Elsevier Science Publ. 1986)

**Boose 1988**
John H.Boose. Uses of repertory grid-centred knowledge acquisition tools for knowledge-based systems. International Journal of Man-Machine Studies 29 (1988)

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

**Boose and Gaines 1988**
J.H.Bosse and B.R.Gaines, eds. Knowledge acquisition tools for expert systems. (Academic Press 1988)

**Brachman and Levesque 1985**
Ronald G.Brachman and Hector J.Levesque, eds. Readings in knowledge representation. (Morgan Kaufmann Publishers Inc. 1985)

**Brachman et al. 1985**
Ronald G.Brachman, Richard E.Fikes, and Hector J.Levesque. KRYPTON: A functional approach to knowledge representation. In: **Brachman and Levesque 1985**

**Bramer 1985**
M.A.Bramer, ed. Research and development in expert systems. (Cambridge University Press 1985)

**Breuker and Wielinga 1985**
Joost Breuker and Bob Wielinga. KADS: Structured knowledge acquisition for expert systems. In: Expert systems and their applications. 5èmes Journées internationales. Avignon, France. May 1985. (Agence de l'informatique 1985)

**Breuker and Wielinga 1987**
Joost Breuker and Bob Wielinga. Use of models in the interpretation of verbal data. In: **Kidd 1987**

**Buchanan and Feigenbaum 1978**

Bruce G.Buchanan and Edward A.Feigenbaum. DENDRAL and Meta-DENDRAL: their applications dimension. Artificial Intelligence 11 (1978).

**Buchanan et al. 1983**
Bruce G.Buchanan, David Barstow, Robert Bechtal, James Bennett, William Clancey, Casimir Kulikowski, Tom Mitchell, and Donald A. Waterman. Constructing an expert system. In: **Hayes-Roth et al. 1983**

**Burton et al. 1988**
A.M.Burton, N.R.Shadbolt, A.P.Hedgecock, and G.Rugg. A formal evaluation of knowledge elicitation techniques for expert systems: domain 1. In: D.S.Moralee. Research and development in expert systems IV. Proceedings of Expert Systems '87. Brighton, England. December 1987. (Cambridge University Press 1988)

**Bylander and Chandrasekaran 1987**
Tom Bylander and B.Chandrasekaran. Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition. International Journal of Man-Machine Studies 26 (1987)

**Clancey 1983**
W.J.Clancey. The epistemology of a rule-based expert system - a framework for explanation. Artificial Intelligence 20 (1983)

**Clancey 1985**
William J.Clancey. Heuristic Classification. Artificial Intelligence 27 (1985)

**Cooke and McDonald 1987**
Nancy M.Cooke and James E. McDonald. The application of psychological scaling techniques to knowledge elicitation for knowledge-based systems. International Journal of Man-Machine Studies 26 (1987)

**Cromp 1985**
Robert T.Cromp. The task, design and approach of the Advice Taker/Inquirer System. TR-85-014. Arizona State University, Tempe, Arizona.

**Davis 1977**
Randall Davis. Interactive transfer of expertise: acquisition of new inference rules. In: Proceedings of the 5th IJCAI. Cambridge, MASS, 1977

**Davis and Lenat 1982**
Randall Davis and Douglas B.Lenat. Knowledge-based systems in artificial intelligence. (McGraw-Hill 1982)

**De Greef and Breuker 1985**
Paul de Greef and Joost Breuker. A case study in structured knowledge acquisition. In: Proceedings of the 9th IJCAI, Los Angeles, CA, 1985

**De Greef, Schreiber, and Wielemaker 1988**
Paul de Greef, Guus Schreiber, and Jan Wielemaker. StatCons: een case-study in gestructureerde kennisacquisitie. In: Proceedings of the NAIC-88. Amsterdam, April 1988.

**De Groot 1965**
Adriaan de Groot. Thought and choice in chess. (Mouton, The Hague 1965)

**Diederich et al. 1987**
Joachim Diedrich, Ingo Ruhmann and Mark May. KRITON: a knowledge acquisition tool for expert systems. International Journal of Man-Machine Studies 26 (1987)

**Duda and Reboh 1984**
Richard O.Duda and René Reboh. AI and decision making: The PROSPECTOR experience. In: W.Reitman, ed. Artificial Intelligence applications for business. (Ablex Publ.Comp. 1984)

**Ericsson and Simon 1984**
K.A.Ericsson and H.A.Simon. Protocol analysis. Verbal reports as data. (MIT Press 1984)

**Eshelman 1988**
Larry Eshelman. MOLE: a knowledge acquisition tool that buries certainty factors. International Journal of Man-Machine Studies 29 (1988)

**Eshelman et al. 1987**
Larry Eshelman, D.Ehret, J.McDermott, and M.Tan. MOLE: a tenancious knowledge acquisition tool. International Journal of Man-Machine Studies 26 (1987)

**Feigenbaum 1977**
Edward A.Feigenbaum. The art of artificial intelligence: themes and case studies of knowledge engineering. In: Proceedings of the 5th IJCAI, Cambridge, MASS, 1977

**Feigenbaum and McCorduck 1983**
Edward Feigenbaum and Pamela McCorduck. The fifth generation: artificial intelligence and Japan's computer challenge to the world. (Addison-Wesley 1983)

**Findler 1979**
N.V.Findler, ed. Associative networks: Representation and use of knowledge by computer. (Academic Press 1979)

**Friedland 1981**
Peter Friedland. Acquisition of procedural knowledge from domain experts. In: Proceedings of the 7th IJCAI, Vancouver, B.C., 1981

**Fum 1985**
Danilo Fum. Natural language processing and the automatic acquisition of knowledge. In: Proceedings of the Second Conference of the European Chapter of the Association for Computational Linguistics. Geneva, Switzerland. March 1985

**Gaines 1987**
Brian Gaines. An overview of knowledge acquisition and transfer. International Journal of Man-Machine Studies 26 (1987)

**Gaines and Boose 1988**
B.R.Gaines and J.H.Boose, eds. Knowledge acquisition for knowledge-based systems. (Academic Press 1988)

**Gale 1987**
William A.Gale. Knowledge-based knowledge acquisition for a statistical consulting system. International Journal of Man-Machine Studies 26 (1987)

**Gammack and Young 1985**
John G.Gammack and Richard M.Young. Psychological techiques for eliciting expert knowledge. In: **Bramer 1985**

**Gentner and Stevens 1983**
D.Gentner and A.L.Stevens, eds. Mental models. (Erlbaum 1983)

**Ginsberg et al. 1985**
A.Ginsberg, S.Weiss, and P.Politakis. SEEK2: A generalized approach to automatic knowledge base refinement. In: Proceedings of the 9th IJCAI, Los Angeles, CA, 1985

**Goodall 1985**
Alex Goodall. The guide to expert systems. (Learned Information Ltd. 1985)

**Greiner and Lenat 1980**
R.Greiner and D.B.Lenat. A representation language language. In: Proceedings of the 1st conference of the AAAI, Stanford, CA, 1980

**Greiner et al. 1988**
Russell Greiner, Bernhard Silver, Sue Becker and Michael Grüninger. A review of machine learning at AAAI-87. Machine Learning 3 (1988).

**Grosz and Stickel 1983**
B.J.Grosz and M.E.Stickel. Research on interactive acquisition and use of knowledge. Final rep. 3 Jul 80 - 30 Nov 83 SRI International, Menlo Park, CA. 1983

**Grover 1983**
Mark D.Grover. A pragmatic knowledge acquisition methodology. In: Proceedings of the 8th IJCAI, Karlsruhe, West Germany, 1983

**Gruber and Cohen 1987**
Thomas R.Gruber and Paul R.Cohen. Design for acquisition: principles of knowledge-system design to facilitate knowledge acquisition. International Journal of Man-Machine Studies 26 (1987)

**Haas and Hendrix 1980**
Norman Haas and Gary G.Hendrix. An approach to acquiring and applying knowledge. In:

Proceedings of the 1st conference of the AAAI, Stanford, CA, 1980

**Haas and Hendrix 1983**
Norman Haas and Gary G.Hendrix. Learning by being told: acquiring knowledge for information management. In: **Michalski et al. 1983**

**Hart 1986**
Anna Hart. Knowledge acquisition for expert systems. (McGraw-Hill 1986)

**Haugeland 1981**
John Haugeland, ed. Mind design. (MIT Press 1981)

**Hawkins 1983**
David Hawkins. An analysis of expert thinking. International Journal of Man-Machine Studies 18 (1983)

**Hayes-Roth et al. 1983**
Frederick Hayes-Roth, Donald A.Waterman, and Douglas B.Lenat, eds. Building expert systems. (Addison-Wesley 1983)

**Hayes-Roth et al. 1986**
F.Hayes-Roth, P.Klahr, D.J.Mostow. Knowledge acquisition, knowledge programming, and knowledge refinement. In: P.Klahr and D.A.Waterman. Expert systems; techniques, tools, and application (Addison Wesley 1986)

**Hayward, Wielinga, and Breuker 1987**
S.A.Hayward, B.J.Wielinga, and J.A.Breuker. Structured analysis of knowledge. International Journal of Man-Machine Studies 26 (1987)

**Hink and Woods 1987**
R.F.Hink and D.L.Woods. How humans process uncertain knowledge: an introduction for knowledge engineers. AI Magazine 8 (1987)

**International Journal of Policy Analysis 1980**
Special Issues on Knowledge and Machine Learning.International Journal of Policy Analysis and Information Systems 4 (1980)

**Jackson 1986**
Peter Jackson. Introduction to expert systems. (Addison Wesley 1986)

**Jacobson and Freiling 1988**
Chris Jacobson and Michael J.Freiling. ASTEK: A multi-paradigm knowledge acquisition tool for complex structured knowledge. International Journal of Man-Machine Studies 29 (1988)

**Johnson-Laird 1983**
P.N.Johnson-Laird. Mental models. (Cambridge University Press, 1983)

**Kahn et al. 1984**
Gary Kahn, Steve Nowlan, and John McDermott. A foundation for knowledge acquisition. In: Proceedings of the IEEE workshop on principles of knowledge-based systems. Denver, CO, December 1984

**Kahn et al. 1985a**
Gary Kahn, Steve Nowlan, and John McDermott. MORE: An intelligent knowledge acquisition tool. In: Proceedings of the 9th IJCAI, Los Angeles, CA, 1985

**Kahn et al. 1985b**
Gary Kahn, Steve Nowlan, and John McDermott. Strategies for knowledge acquisition. In: IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-7 (1985)

**Kahn et al. 1987**
Gary S.Kahn, Edwin H.Breaux, Peter DeKlerk and Robert L.Joseph. A mixed-initiative workbench for knowledge acquisition. International Journal of Man-Machine Studies 27 (1987)

**Kelly 1955**
G.A.Kelly. The psychology of personal constructs. (New York 1955)

**Kidd 1987**
Alison L.Kidd. Knowledge elicitation for expert systems. A practical handbook. (Plenum Press 1987)

**Kitto and Boose 1987**
Catherine M.Kitto and John H.Boose. Heuristics for expertise transfer: an implementation of a dialog manager for knowledge acquisition. International Journal of Man-Machine Studies 26 (1987)

**Klahr 1976**
David Klahr, ed. Cognition and instruction. (Erlbaum 1976)

**Klinker et al. 1987**
Georg Klinker, Joel Bentolila, Serge Genetet, Michael Grimes and John McDermott. KNACK - Report-driven knowledge acquisition. International Journal of Man-Machine Studies 26 (1987)

**Klinker et al. 1988**
Georg Klinker, Serge Genetet and John McDermott. Knowledge acquisition for evaluation systems. International Journal of Man-Machine Studies 29 (1988)

**Kornell 1984**
Jim Kornell. A VAX tuning expert built using automated knowledge acquisition. In: Proceedings of the first conference on AI applications. IEEE - 84 CH 2107-1. Denver, CO, 1984

**Kornell 1987**
Jim Kornell. Formal thought and narrative thought in knowledge acquistition. International Journal of Man-Machine Studies 26 (1987)

**LaFrance 1987**

Marianne LaFrance. The knowledge acquisition grid: a method for training knowledge engineers. International Journal of Man-Machine Studies 26 (1987)

**Lefkowitz and Lesser 1988**
Lawrence S.Lefkowitz and Victor R.Lesser. Knowledge acquisition as knowledge assimilation. International Journal of Man-Machine Studies 29 (1988)

**Lenat et al. 1986**
D.Lenat, M.Prakash, and M.Shepherd. CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. AI Magazine 6 (1986)

**Marcus 1987**
Sandra Marcus. Taking backtracking with a grain of SALT. International Journal of Man-Machine Studies 26 (1987)

**Marcus 1988**
Sandra Marcus, ed. Automating knowledge acquisition for expert systems. (Kluwer 1988)

**Marcus et al. 1985**
Sandra Marcus, John McDermott, and Tianran Wang. Knowledge acquisition for constructive systems. In: Proceedings of the 9th IJCAI, Los Angeles, CA, 1985

**Michalski 1987**
Ryszard S.Michalski. Learning strategies and automated knowledge acqusition. An overview. In: Leonard Bolc, ed. Computational models of learning. (Springer Verlag 1987)

**Michalski et al. 1983**
Ryszard S.Michalski, Jaime G.Carbonell, Tom M.Mitchell. Machine Learning. An artificial intelligence approach. (Tioga Publishing Company 1983)

**Michie 1979**
Donald Michie, ed. Expert systems in the micro electronic age. (Edinburgh University Press 1979)

**Michie 1987**
Donald Michie. Current developments in expert systems. In: J.Ross Quinlan, ed. Applications of expert systems. (Addison Wesley 1987)

**Mitchell et al. 1985**
Tom Mitchell, Sridhar Mahadevan and Louis I.Steinberg. LEAP: A learning apprentice for VLSI design. In: Proceedings of the 9th IJCAI, Los Angeles, CA, 1985

**Mitchell et al. 1986**
Tom.M.Mitchell, Jaime G.Carbonell, and Ryszard S.Michalski, eds. Machine Learning. A guide to current research. (Kluwer Academic Publishers 1986)

**Mittal and Dym 1985**
S.Mittal and C.L.Dym. Knowledge acquisition from multiple experts. AI magazine 6 (1985)

**Moore and Agogino 1987**
Eric A.Moore and Alice M.Agogino. INFORM: an architecture for expert-directed knowledge acquisition. International Journal of Man-Machine Studies 26 (1987)

**Morik 1987**
Katharina Morik. Acquiring domain models. International Journal of Man-Machine Studies 26 (1987)

**Motta et al. 1988**
Enrico Motta, Marc Eisenstadt, Kent Pitman, Malcom West. Support for knowledge acquisition in the Knowledge Engineer's Assistant (KEATS). Expert Systems 5 (1988)

**Musen et al. 1987**
Mark A.Musen, Lawrence M.Fagan, David M.Combs and Edward H.Shortliffe. Use of a domain model to drive an interactive knowledge-editing tool. International Journal of Man-Machine Studies 26 (1987)

**Politakis 1985**
P.G.Politakis. Empirical analysis for expert systems. (Pitman 1985)

**Porter and Bareiss 1986**
Bruce W.Porter and Ray E.Bareiss. PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. Report AI TR86-35 September 1986. University of Texas, Austin, TX.

**Porter et al. 1986**
Bruce W.Porter, Ray Bareiss, and Adam Farquhar. Acquiring domain knowledge from fragments of advice. In: **Mitchell et al. 1986**

**Prerau 1987**
D.S.Prerau. Knowledge acquisition in the development of a large expert system. AI Magazine 8 (1987)

**Quinlan 1979**
J.Ross Quinlan. Discovering rules by induction from large collections of examples. In: **Michie 1979**

**Rosenbloom et al. 1984**
Paul Rosenbloom, John E.Laird, John McDermott, Allen Newell, and Edmund Orciuch. R1-Soar. An experiment in knowledge-intensive programming in a problem-solving architecture. In: Proceedings of the IEEE workshop on principles of knowledge-based systems. December 1984. Denver, CO.

**Rosenbloom et al. 1986**
Paul S.Rosenbloom, John E.Laird, Allen Newell, Andrew Golding, and Amy Unruh. Current research on learning in SOAR. In: **T.M.Mitchell et al. 1986**

**RuleMaster2 1987**
RuleMaster2. A software tool for building expert systems. A technical introduction. Radian Corporation, Austin, TX 1987

**Schweickert et al. 1987**
R.Schweickert et al. Comparing knowledge elicitation techniques: a case study. Artificial Intelligence Review 1 (1987)

**Shachter and Heckerman 1987**
R.D.Shachter and D.E.Heckerman. Thinking backward for knowledge acquisition. AI Magazine 8 (1987)

**Shalin et al. 1988**
Valerie L.Shalin, Edward J.Wisniewski, and Keith R.Levi. A formal analysis of machine learning systems for knowledge acquisition. International Journal of Man-Machine Studies 29 (1988)

**Shaw 1982**
Mildred L.G.Shaw. PLANET: some experience in creating an integrated system for repertory grid applications on a microcomputer. International Journal of Man-Machine Studies 17 (1982)

**Shaw and Gaines 1987**
Mildred L.G.Shaw and Brian R.Gaines. KITTEN: Knowledge initiation and transfer tools for experts and novices. International Journal of Man-Machine Studies 27 (1987)

**Shaw and Woodward 1988**
Mildred L.G.Shaw and J.Brian Woodward. Validation in a knowledge support system: construing and consistency with multiple experts. International Journal of Man-Machine Studies 29 (1988)

**Shema and Boose 1988**
David B.Shema and John H.Boose. Refining problem-solving knowledge in repertory grids using a consultation mechanism. International Journal of Man-Machine Studies 29 (1988)

**Simon 1978**
H.A.Simon. On the forms of mental representation. In: C.W.Savage. Perception and cognition. (Minnesota Studies in the philosophy of science, 1978)

**Slatter 1987**
P.E. Slatter. Building expert systems: cognitive emulation. (Ellis Horwood 1987)

**Stout et al. 1988**
Jeffrey Stout, Gilbert Caplain, Sandra Marcus, and John McDermott. Toward automating recognition of differing problem-solving demands. International Journal of Man-Machine Studies 29 (1988)

**Thiemann 1989**
Janine Thiemann. Wielinga over KADS. Kennissystemen 3 (1989)

**Tranowski 1988**
Deborah Tranowski. A knowledge acquisition environment for scene analysis. International Journal of Man-Machine Studies 29 (1988)

**Van de Brug et al. 1984**
Arnold van de Brug, Judith Bachant, and John McDermott. The taming of R1. IEEE Expert 1 (1986)

**Van den Herik 1986**
H.J.van den Herik. Het gebruik van kennis in expertsystemen.
In: Anton Nijholt & Luc Steels, eds. Ontwikkelingen in expertsystemen. (Academic Service, Den Haag 1986)

**Van Dijk et al. 1988**
J.E.M.van Dijk, F.G. Hilgevoord, M.T.Jacques, G.A.M.Otten, A.C. Sitting, and J.L.Talmon. Drie methoden voor kennisverwerving. Kennissystemen 2 (1988)

**Waterman 1986**
D.Waterman. A guide to expert systems. (Addison Wesley 1986)

**Weiss 1987**
Ray Weiss. System automates knowledge acquisition. Electronic Engineering Times. Issue 428, April 6, 1987

**Weiss and Kulikowski 1979**
Sholom M.Weiss and Casimir A.Kulikowski. EXPERT: A system for developing consultation models. Proceedings of the 6th IJCAI, Tokyo, 1979.

**Weiss and Kulikowski 1984**
Sholom M.Weiss and Casimir A.Kulikowski. A practical guide to designing expert systems. (Chapman and Hall, London 1984)

**Weizenbaum 1966**
J.Weizenbaum. ELIZA - A computer program for the study of natural language communication between man and machine. Communications of the Association for Computing Machinery 9 (1966)

**Welbank 1983**
Margaret Welbank. A review of knowledge acquisition techniques for expert systems. Techn.Rep. British Telecom Research Laboratories, Martlesham Heath, Ipswich, 1983

**Wielinga and Breuker 1985**
B.J.Wielinga and J.A.Breuker. Interpretation of verbal data for knowledge acquisition. In: T.O'Shea, ed. Advances in artificial intelligence. ECCAI-85. (Elsevier Science Publ. 1985)

**Wielinga and Breuker 1987**
B.J.Wielinga and J.A.Breuker. Models of expertise. In: B. Du Boulay, D.Hogg, and L.Steels, eds. Advances in artificial intelligence - II. (Elsevier Science Publ. 1987)

**Wilczynski 1981**
David Wilczynski. Knowledge acquisition in the CONSUL system. In: Proceedings of the 7th IJCAI, Vancouver, B.C. 1981

**Wilkins et al. 1986**
David C.Wilkins, William J.Clancey, and Bruce G.Buchanan. Overview of the ODYSSEUS learning apprentice. In: **Mitchell et al. 1986**

**Wilkins et al. 1987**
David C.Wilkins, William J.Clancey, and Bruce G.Buchanan. Knowledge base refinement by monitoring abstract control knowledge. International Journal of Man-Machine Studies 27 (1987)

**Winograd and Flores 1987**
Terry Winograd and Fernando Flores. Understanding computers and cognition; a new foundation for design. (Ablex Publ. Comp. 1987)

**Wright and Ayton 1987**
George Wright and Peter Ayton. Eliciting and modelling expert knowledge. Decision Support Systems 3 (1987)

**Wrobel 1988**
Stefan Wrobel. Design goals for sloppy modeling systems. International Journal of Man-Machine Studies 29 (1988).