1992

Automated knowledge acquisition for knowledge-based systems: KE-KIT

John Parsons Scott

KE-KIT

AUTOMATED KNOWLEDGE ACQUISITION

FOR

KNOWLEDGE-BASED SYSTEMS

Rochester Institute of Technology
Department of Computer Science

Automated Knowledge Acquisition for Knowledge-Based Systems
>> KE-KIT <<

The Use of Kelly's Personal Construct Theory in the Automation of Knowledge Acquisition (Theory and Prototype)

by

John Scott Parsons

A thesis submitted to
The Faculty of the Department of Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.

Approved by:

Professor John A. Biles

Mr. William G. Thiers

Professor Walter A. Wolf

Dedicated to my mother and grandmother

Automated Knowledge Acquisition for Knowledge-Based Systems: KE-KIT

The Use of Kelly's Personal Construct Theory in the Automation of Knowledge Acquisition

(Theory and Prototype)

I, John Scott Parsons, prefer to be contacted each time a request for reproduction of this thesis is made. I can be reached at the following address:

248 Crittenden Way #5

Rochester, New York 14623-2215

Signed:

Date: 2-13-92

ABSTRACT

Despite recent progress, knowledge acquisition remains a central problem for the development of intelligent systems. There are many people throughout the world doing studies in this area. However, very few automated techniques have made it to the market place. In this light, the idea of automating the knowledge acquisition process is very appealing and may lead to a break through. Most (if not all) of the approaches and techniques concerning intelligent, expert systems and specifically knowledge-based systems can still be considered in their infancy and definitely do not subscribe to any kind of standards. Many things have yet to be learned and incorporated into the technology and combined with methods from traditional computer science and psychology.

KE-KIT is a prototype system which attempts to automate a portion of the knowledge engineering process. The emphasis is on the automation of knowledge acquisition activities. However, the transformation of knowledge from an intermediate form to a knowledge-base format is also addressed. The approach used to automate the knowledge acquisition process is based on the personal construct theory developed by George Kelly in the field of psychology.

This thesis gives and in-depth view of knowledge engineering with a concentration on the knowledge acquisition process. Several issues and approaches are described. Greater details surrounding the personal construct theory approach to knowledge acquisition and its use of a repertory grid are given. In addition, some existing knowledge acquisition tools are briefly explored. Details concerning the implementation of KE-KIT and reflections on its applicability round out the presented material.

Keywords:

Knowledge Acquisition
Automated Knowledge Acquisition
Personal Construct Theory
Construct (attribute, trait, condition)
Repertory Grid
Rating Grid
Knowledge Engineering
Expert Systems
Knowledge-Based Systems

CHAPTER 1 OVERVIEW of THESIS

INTRODUCTION

The following material introduces and documents the automated knowledge acquisition tool KE-KIT. The tool is an attempt to make the knowledge engineering process easier and to reduce the amount of time required for information gathering. In order to understand the tool better, this text is divided into several chapters. The first chapter provides some basic background material about knowledge-based systems, knowledge engineering and the knowledge acquisition process. Chapter two covers knowledge acquisition in more detail; the problems are discussed as well as current research areas and current manual techniques. Personal Construct Theory, the basis for KE-KIT, is discussed in chapter three. This chapter covers the basic theory as well as other application areas and how the methods of the theory are used. Chapter four, then, surveys several automated tools in existence in one form or another. KE-KIT and its implementation is discussed in chapter five. Finally, chapter six covers the results and conclusions of KE-KIT in detail. Other related information to this work is presented in the appendix.

BACKGROUND

KNOWLEDGE BASE SYSTEMS

The creation and management of Knowledge Bases has become a central research issue in the world of artificial intelligence. Many reasons have been given over the years for implementing Knowledge-Based Systems (KBS). Some of them include:

- Experts retire, taking their knowledge with them.
- There may be better ways to make use of an expert's time than answering others' routine questions.
- Expertise may be scarce.
- Expertise may be expensive to deliver.
- An expert system may help a product or service be delivered in a more timely manner since expertise is not always immediately available.
- Experts are not always consistent.

PROJECT ISSUES

Many things must be considered when undertaking a KBS project. The major issues involve knowledge acquisition. Some issues to consider during the creation of a KBS include: 1) handling of multiple experts, 2) validation and verification, 3) the user interface and other human factors, 4) use of an understandable knowledge representation, and 5) integration with other systems and technology. Human factors are often neglected. The human-computer issues are very important and can make or fail a project no matter what else happens. Most Knowledge-Based systems do not stand alone, thus integration issues of working with other manual or automated systems must be addressed. This may involve procedural changes and/or actual program changes and enhancements. Typically knowledge-based systems are integrated with simulation, database, communication, hypermedia, or traditional third generation systems.

KBS projects require the support and effort of many people. These may include a user representative, a supplier of knowledge (expert), management of the target activity, a

system maintainer, an auditor (responsible for validation), management of the knowledge engineering process, a primary interviewer, a note taker or back-up interviewer, a knowledge analyzer, and a system coder. The distinction between "expert", "knowledge engineer", and "clients" or "users" may often be fuzzy. This is especially true is a person is performing several activities on a project, which is common on smaller projects. Automated elicitation of knowledge blurs the previously mentioned titles even more when an "expert" is acting as a "knowledge engineer" and so on. The minimal set of people needed for a project include the management of the domain and the service provider, a user, a domain expert, and a knowledge engineer.

GENERAL PROBLEMS

Most AI-based projects involve some level of risk. A project may fail, among other reasons, due to a lack of cooperation or buy-in, a shortcoming in the knowledge that is captured and therefore restricted usefulness, the lack of acceptance and use of the system, technology restrictions, or due to complexity of use. Since a level of risk does exist, some guidelines should be followed, to minimize this risk, when selecting potential KBS applications. All of these are not required, but the more that exist, the better the chances of success.

1) The end user of the system should be familiar with domain terminology
2) Someone already knows how to solve the problem
3) A single expert can solve the problem; a team is not required
4) The expert can consult with users over the phone
5) There is justification (business, technical, and functionally)

6) The expert will be available for long periods of time
7) The expert has a collection of real cases readily available

APPLICATION SPECTRUM

There are two ends of a spectrum that describe the types of problems handled by Knowledge-Based Systems. *Analysis* or interpretation problems are on one end of the spectrum while *synthesis* or construction problems are on the other end. In between there are combinations of both. The solutions to analysis problems can be enumerated, while synthesis solutions are built using rules and constraints. The following table (also see figure 1.1) describes the realm of application problems.

ANALYSIS problems

> Identify (recognize) - recognize a system from its observed inputs and outputs

1) Monitor - detect discrepancies in (audit,check) behavior (or characterize the current state) of a system
2) Diagnose (debug) - explain monitored behavior in terms of discrepancies between the actual system and the intended design

> Predict (simulate) - given a known system, describe output behavior from its inputs

> Control - given a known system determine inputs to generate required outputs

SYNTHESIS problems

> Specify (constrain) - constrain a system description

> Design - describe a system in terms of spatial and temporal interactions of components

1) Configure - characterize a structure

2) Plan (process) - characterize a process

> Assemble(manufacture) - execute a plan for construction of a system

1) Modify (repair) - transform a system to enact a redesign (change of structure)

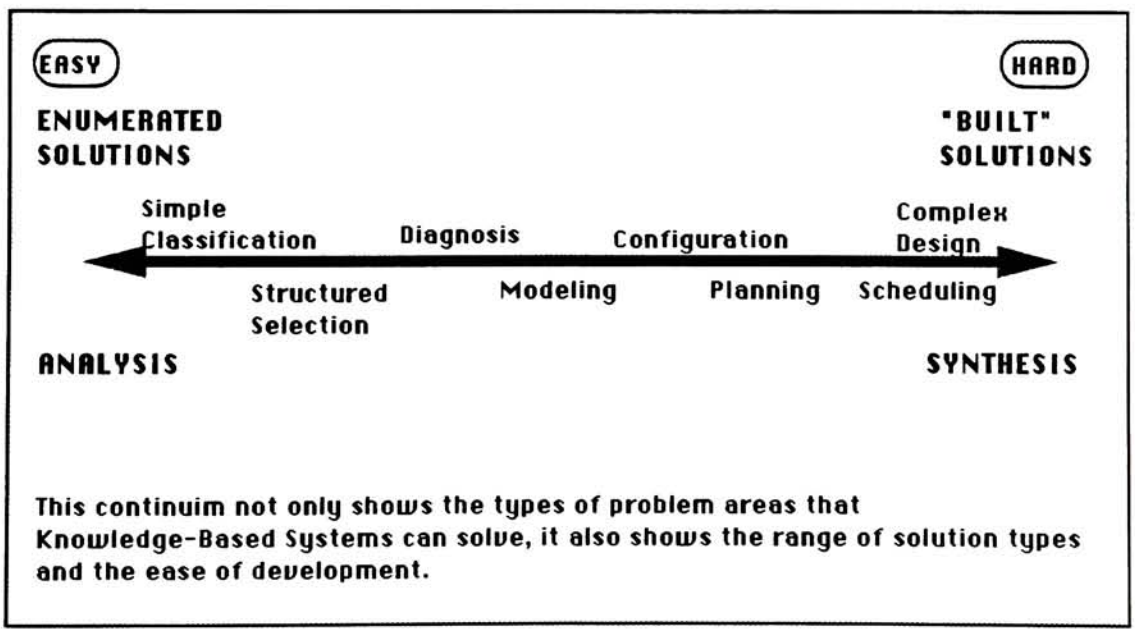


Figure 1.1

KNOWLEDGE REPRESENTATION

NEED FOR KNOWLEDGE REPRESENTATION

In a sense all programs possess knowledge on how to solve a particular problem. Traditional algorithmic programming stores problem solving knowledge with the code of a program. AI based programming stores the program, data, and knowledge separately. The AI based program thus acts as an interpreter. This allows the user to modify or change the stored knowledge and solve different problems without having to change the underlying program (control mechanism). Knowledge representation is the keystone to AI and systems using AI techniques. It follows then that the power and utility of an expert system depends on the quality of the underlying representation of expert knowledge.

OBJECTIVES

A task in the development of an AI-based program is to map the real problem into a representation that can provide a solution. This involves selecting an appropriate data structure and representing the problem in this structure. A good representation should have characteristics that:
1) highlight only the states of the problem that are important to problem-solving, 2) eliminate the features of the problem that do not contribute to the problem-solving effort,
3) produce a simple representation that is also elegant, and
4) represent the domain in a manageable fashion. This is really saying that the representation must give easy access to the knowledge and also provide an expressive power that sufficiently allows reasoning efficiency. A knowledge representation system must do more than just represent; it must be able to respond to queries about what it represents.

KNOWLEDGE ENGINEERING

NEED FOR KNOWLEDGE ENGINEERING

The more *expert* one is at a task, the harder it is to access one's knowledge of the task. A Knowledge Engineer (KE) must be able to draw on that expert knowledge and put it into a usable form. The KE needs to be a psychologist, a diplomat, and researcher, as well as possess the necessary technical skills. Many delicate human relations situations may arise, which if not handled appropriately, may jeopardize the entire project or corrupt the knowledge gathered. For instance, experts and end-users may feel that their jobs are threatened by such systems or knowledge may not be translated properly due to misunderstandings. The KE process can be thought of as just another activity related to human inquiry.

All traditional KE activities are not replaced by automated tools. To some degree manual methods will never be completely replaced. There are some tasks that a KE performs that can not and probably should not be automated. As tools improve, some control will remain in the hands of a trained professional who can handle problems and keep things in order. A KE will still be needed to advise the expert on the process of the interactive knowledge acquisition tool, to manage the knowledge acquisition tools, to add un-coded or edit coded knowledge with the help of the expert, to help validate the knowledge base created, to work on integration issues, and to help train end-users on the use of the system. In other words, the KE may be used as a technical expert to help the domain expert build applications.

THE KE PROCESS

The KE process can best be described as a cycle of related activities (see figures 1.2 and 1.3).

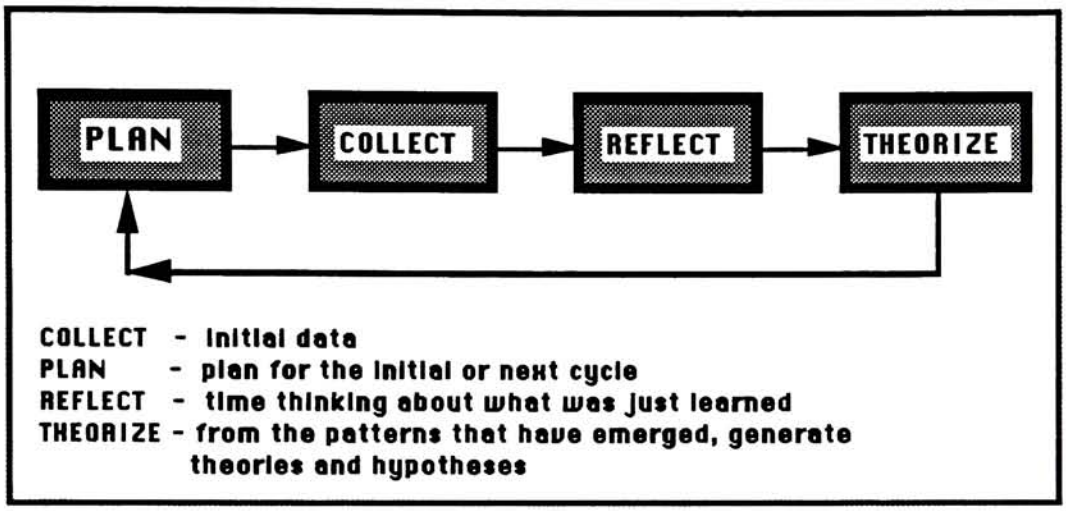


Figure 1.2

The process of manual knowledge acquisition is broken down into phases:

1) Conceptualization of the Domain

   Domain Organization - Identify functional goals and expectations; Identify sub-goals and tasks

   Preliminary Knowledge Acquisition
   - Identify knowledge acquisition needs and resources
   - Preliminary knowledge base design

2) Acquisition of Knowledge

   Identify General Heuristics
   Identify Procedures and Tasks
   Identify Concepts and Vocabulary
   Identify Decision-Making Heuristics
   Identify Analogies and Problem-Solving Techniques

3) Analysis of Results

4) Implementation

5) System delivery and operation

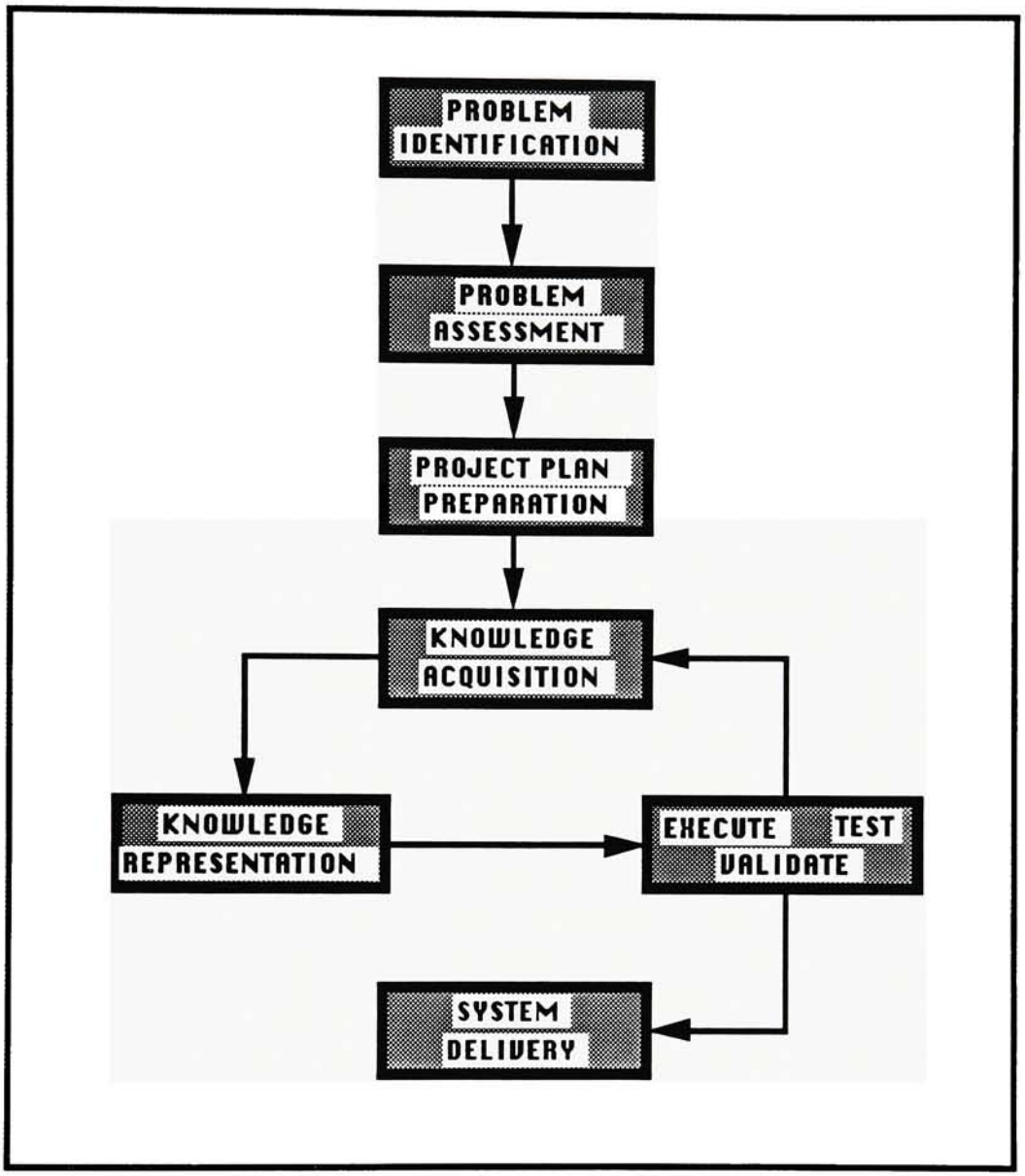


Figure 1.3

KE ACTIVITIES

The role of a knowledge engineer (KE) involves activities which require diverse needs and experience. A KE is the

primary intermediary between an expert and the system building tool. The expert's problem-solving behavior must be modelled in the system. The activities of the knowledge engineering process select knowledge representations to clarify the problem and its solution. Knowledge representations are used which can be understood, tested, and manipulated by anyone involved with the project, NOT just programmers. Appropriate knowledge representations also enable inference processes to be more transparent and documentable.

In all activities performed by a KE, it is very important to be as specific as possible. It helps to devise and use an appropriate method or technique for covering all relevant information. An aid in keeping track of information and activities is the use of "Domain Definition Log" book. This reference may contain information such as the general problem definition, a bibliography of reference documents, a glossary of terms, acronyms and symbols, a list of identified and recognized experts, a definition of appropriate and realistic performance measures, and a description of example reasoning scenarios. This log book is intended to be a living document and should be updated as the project progresses. The completed log book can serve as a useful reference during the maintenance phases as well as for other similar projects.

KNOWLEDGE ENGINEERING versus SOFTWARE ENGINEERING

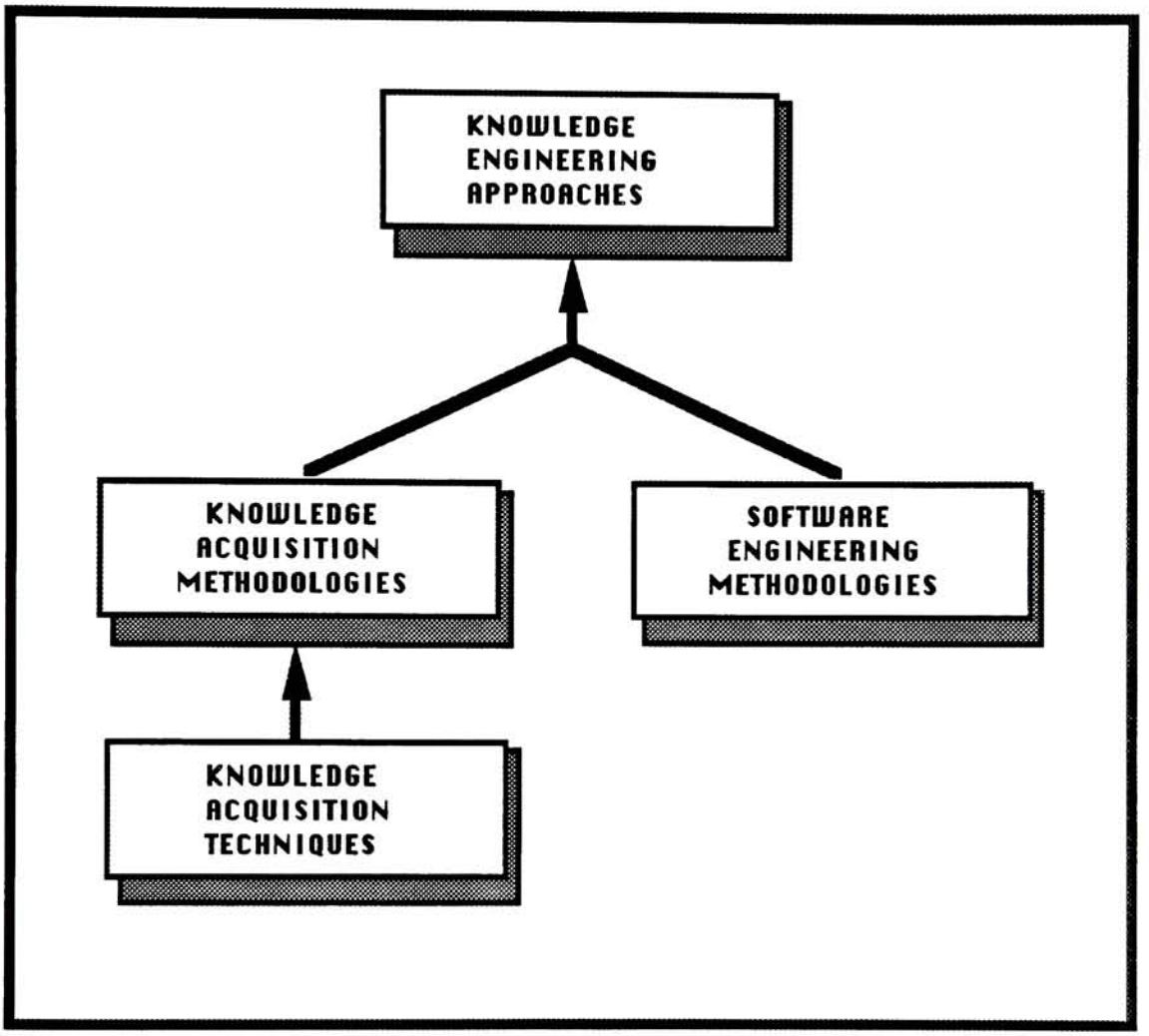


Figure 1.4

Knowledge engineering is similar in many respects to traditional software engineering. In fact, they share many common activities and steps. Each has its own variation of the activities involved as well as the required steps (see figures 1.5 and 1.6). For example, a good systems analyst needs similar interviewing skills as a knowledge engineer. Common steps included problem definition, system specification, design, implementation, testing, delivery, and maintenance. Classic systems development is seen as a set of

linear steps which is heavy on up-front planning. On the other hand, knowledge-based system development divides the tasks fairly evenly across the steps involved. However, the steps of development in knowledge-based systems may be executed several times in an iterative fashion. In this way, a KBS grows by incrementally improving the representation and organization of the knowledge. Another difference is the fact that most AI-based systems code knowledge separate from the control strategy - this is opposite of what occurs for traditional systems. Many of the differences seem to disappear in systems where traditional application solutions are integrated with artificial intelligence technology. In this case, both approaches must work together.

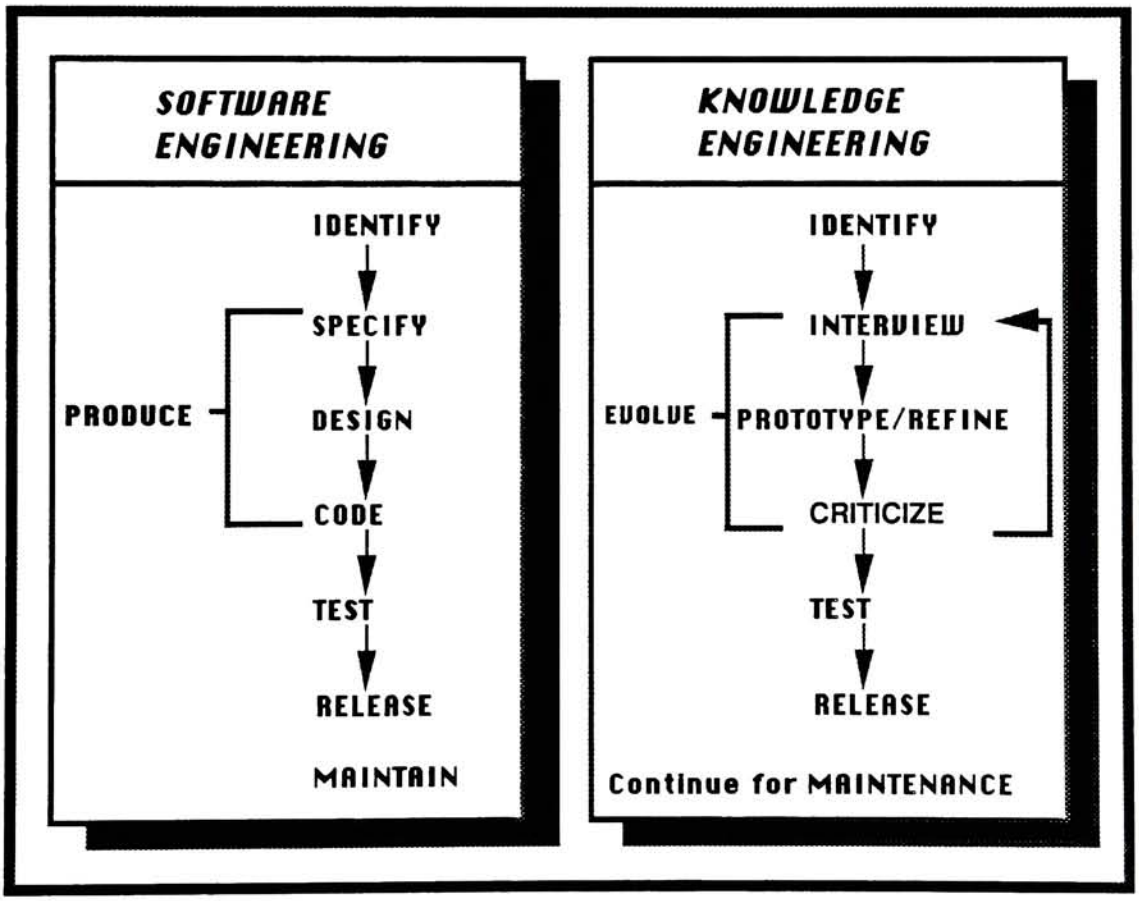


Figure 1.5

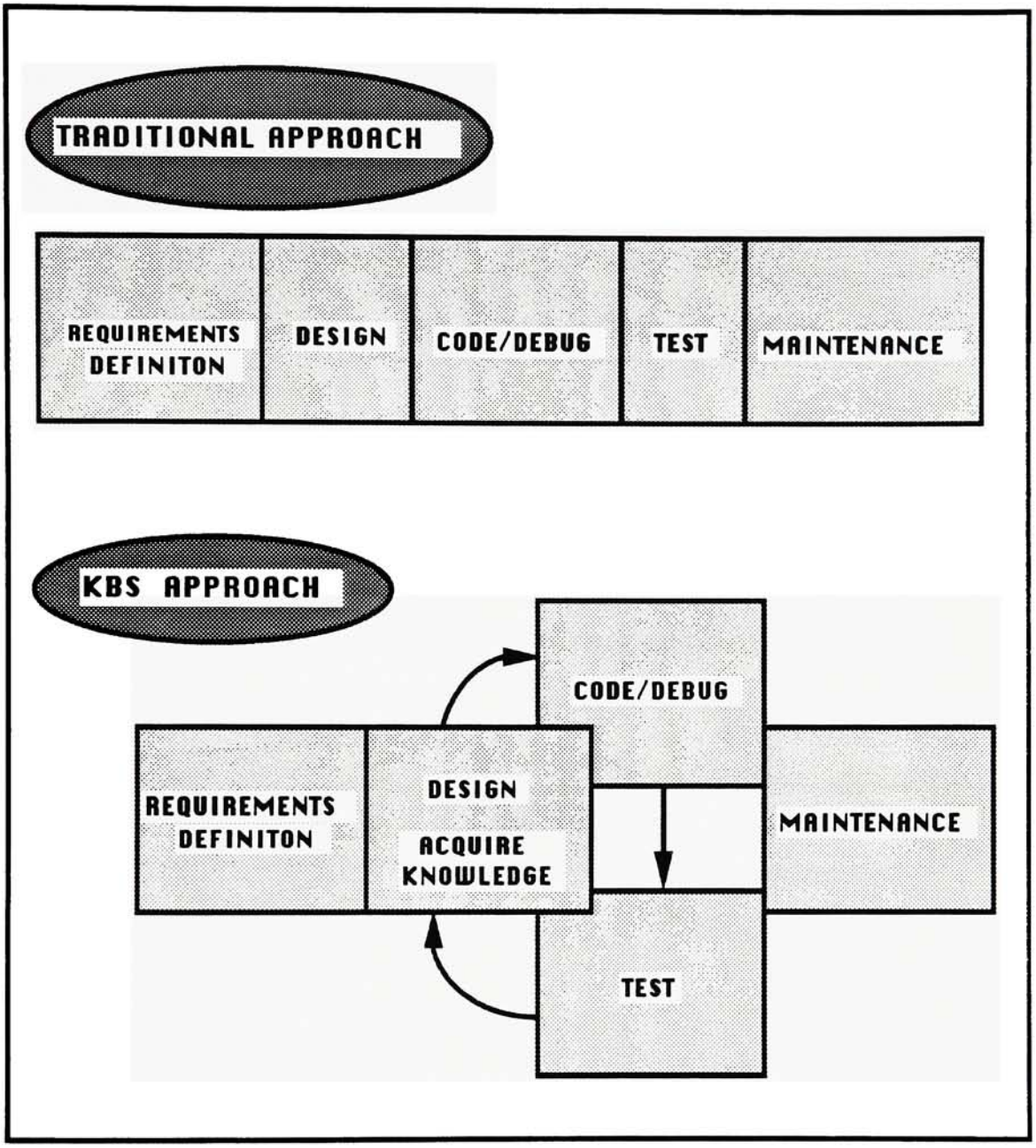


Figure 1.6

The fundamental principal of knowledge-based systems is "knowledge" development rather than "code" development. KBS development involves the acquisition of knowledge and its control mechanism. However, the two components are usually stored separately. This facilitates rapid prototyping. Traditional software development codes knowledge and its

control structure together as one component. KBS development uses knowledge at a higher level of abstraction. This leads to advantages which result from using higher modularity, transparency, and greater knowledge separation from reasoning. Data flow diagrams developed for knowledge-based systems portray a flow of data between "decisions" rather than processes. The following table (also see figure 1.7) summarizes some other differences and similarities between software engineering and knowledge engineering.

Software Engineering	Knowledge Engineering
- activities revolve around encoding program code	- activities revolve around designing and encoding knowledge (beyond code)
- prototyping tools emerging	- rapid prototyping tools exist
- information coded detail level	- information is stored in more abstract form

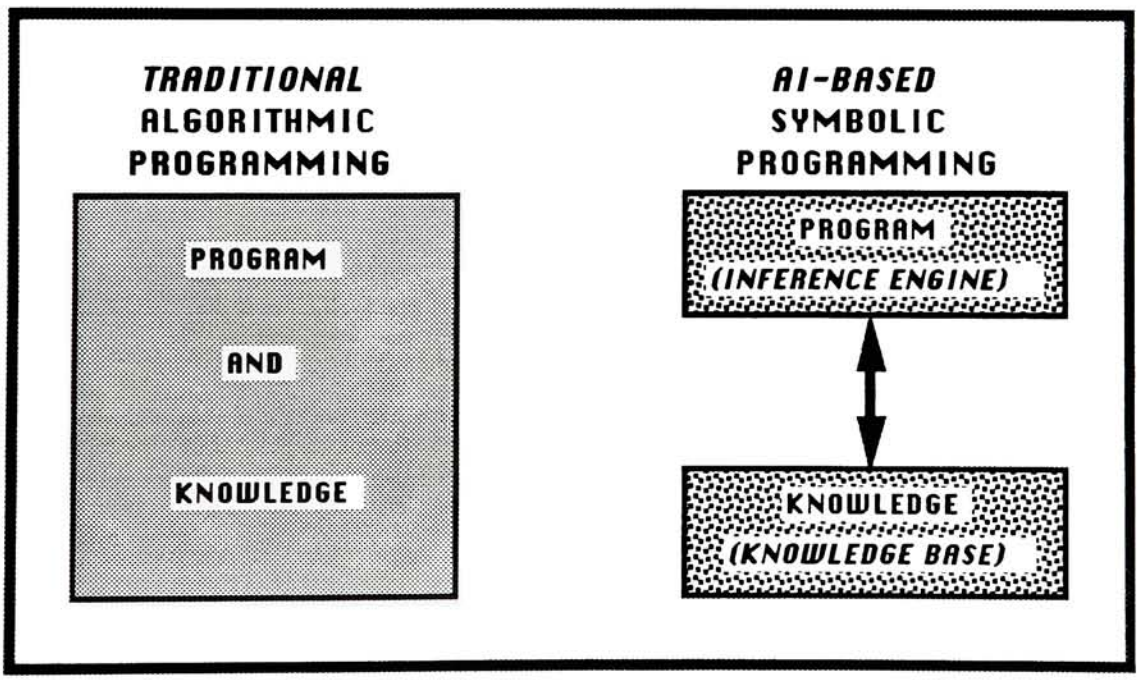


Figure 1.7

EXPERTS and EXPERTISE

Knowledge can be thought of as a natural resource; it is valuable, but not easy to extract. A KE must not only *find* the knowledge, but also process it into a form that is understandable by the expert so that it can be verified. The most common source for domain knowledge is the recognized **expert** on the subject matter. However, knowledge may come from non-experts (end-users, skilled practitioners, administrators, novices, others, etc...) or non-humans as well.

When dealing with experts, it is beneficial to remember the pitfalls of human decision-making. People can give radically different answers to the same problem when it is represented in various ways. If the same answer is received from more than one method, this serves as a validation. People tend to rely on an intuitive *representative* match between the evidence and a model of the situation. They draw conclusions while ignoring probabilistic consideration in uncertain situations and when it is considered, predicted outcomes are not determined consistently or in a necessarily logical manner. People tend to think that they know more than they do and that what they don't know must be unimportant. They also tend to avoid risks when seeking gains, but choose risks to avoid sure losses [BOOSE89]. The overriding issue is that more must be understood about the nature of expertise itself in order to successfully tackle the task of knowledge elicitation.

Many opinions and theories exist as to why problems exist between experts and knowledge engineers. Experts may feel insecure about the knowledge acquisition process. They may feel job insecurity; they may not want computers encroaching on their "private domain;" they may not want to expose their problem-solving methods to the scrutiny of others; they may be willing, but unable to articulate their problem-solving

methods and so on. All of these and other issues make the process more difficult than it already is. Whatever the reason, a feeling of comfort must be achieved in order for the expert to be truly beneficial in project development.

KNOWLEDGE ACQUISITION PROCESS

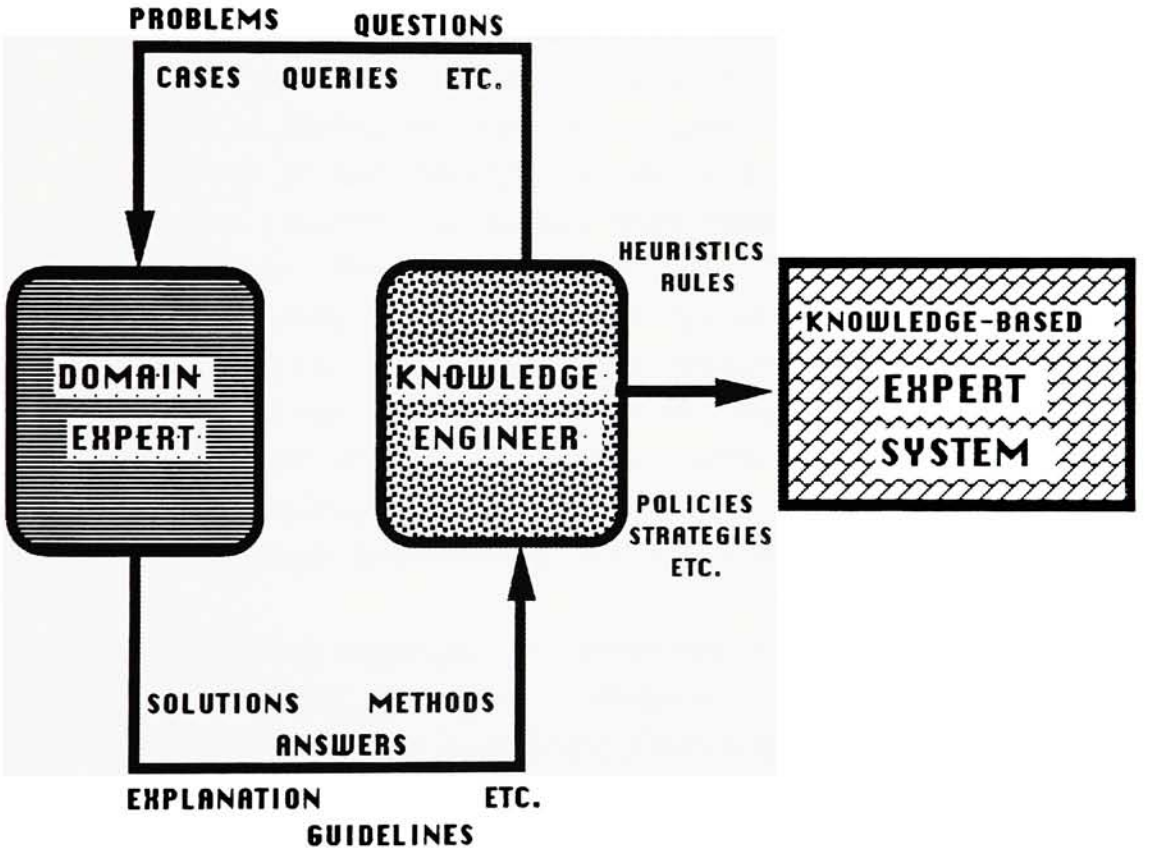


Figure 1.8

Knowledge Acquisition (KA) is the process of extracting, organizing, and structuring knowledge for use in a KBS. KA

usually takes two forms: transferring problem-solving expertise from humans to computers, and inducing problem-solving knowledge from examples. The process of knowledge acquisition has two major interfaces. The first one deals with the interaction between the domain experts and the knowledge engineer. This is the aspect which automated techniques address. The other interface deals with the knowledge engineer and the computer system. The emphasis on this aspect is getting the right knowledge representation for the knowledge acquired from the expert.

Material gathered may be from verbal communication, coded information, text, diagrams, formulae, tables, images, sounds or electrical signals. It may be stored in a computer, on disk, magnetic tapes, or printed. Since there are so many varied sources of information, it follows that there are many techniques to collect the needed knowledge. In the course of developing a KBS, KEs should use as many techniques from their repertoire and tool set as seem appropriate; sometimes using only one technique, sometimes using several. Getting the same knowledge using multiple techniques may serve as a consistency check among other things. Different techniques are good for different situations. When uncertainty is involved, more complex techniques and methods are required.

The issues around uncertainty concerning acquired knowledge is beyond the scope of this text. However, this is an important issue to consider while acquiring knowledge. Consideration of uncertain information does add a level of complexity to not only the knowledge acquisition process, but the entire knowledge-based system development. Implementation and verification is affected just as much as knowledge acquisition. A brief overview of this topic is included in the appendix.

Several things should be kept in mind during the knowledge acquisition process. Be aware that learning will take place on both sides. There may be situations where the *expert* is using the elicitation session to try out new ideas. There should be planned time and energy after each elicitation session to review and supplement the record made during the session with notes that bring out things that were not said, but contributed information (ie. actions, expressions, etc...). Also, a mechanism should exist to provide feedback and an opportunity to check the accuracy of the material elicited.

CHAPTER 2 KNOWLEDGE ACQUISITION

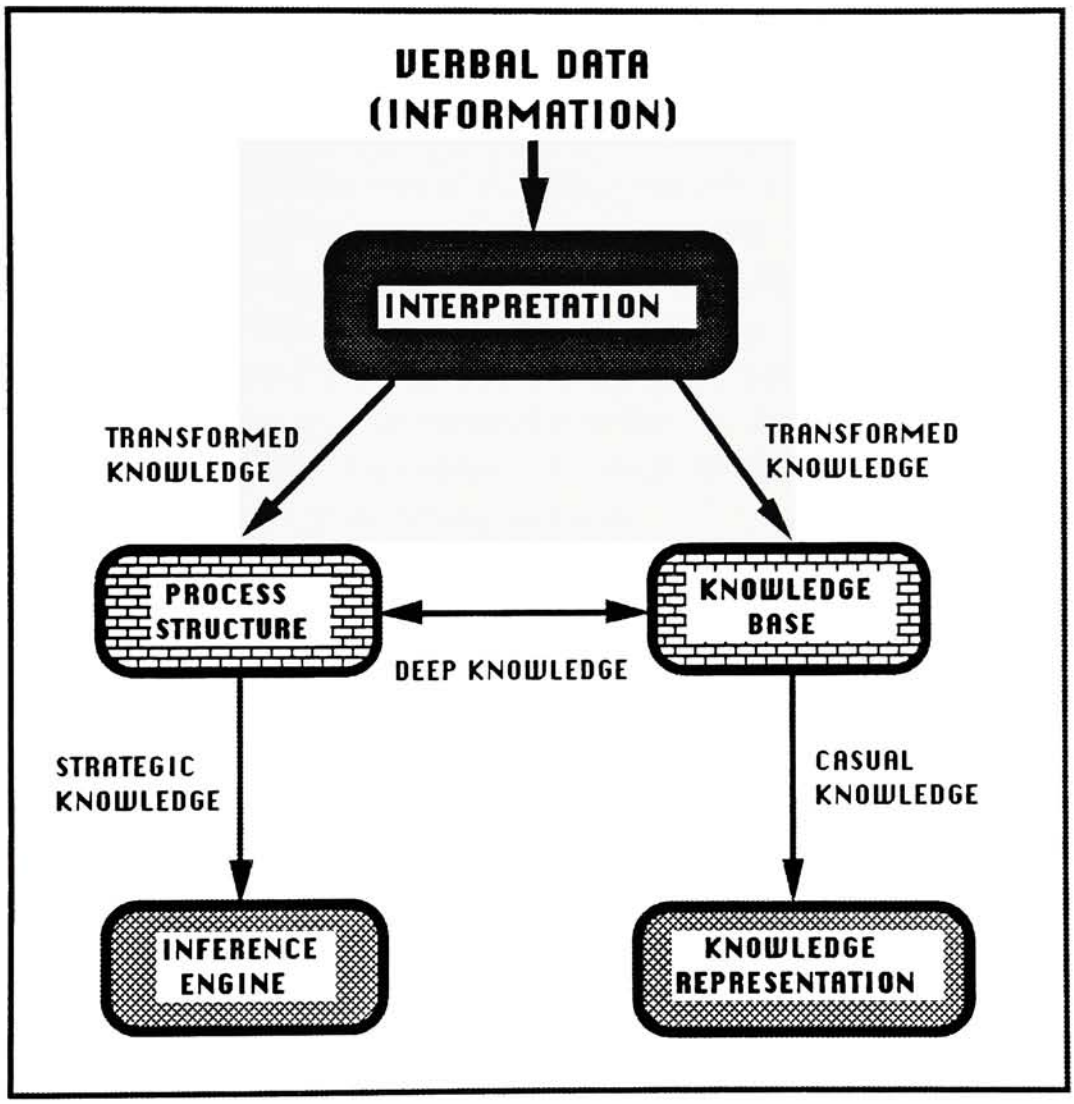
KNOWLEDGE ACQUISITION ISSUES

PROBLEM STATEMENT

Knowledge acquisition is the major hurdle when building knowledge-based systems. The process is time consuming, labor intensive, and often very difficult. Many reasons contribute to this problem. No generally accepted methodology for effectively transferring knowledge from humans to computers exists. Known manual methods are very long and have a tendency to be error-prone. Many techniques are in existence in one form or another. Often they are combined to complement each other and compensate for existing inadequacies. A question of when to use what technique often arises. The "revise and review" cycles of gathering knowledge, modelling, and design may go on for months and even years in order to reach "expert" behavior.

The usual approach to knowledge acquisition seems to follow "common sense" in an undirected manner. Most KEs do not have sufficient interview training to properly "extract" information. Domain experts have difficulty expressing their problem-solving methods. Explanations may seem to come directly from text books. It can be very difficult for an expert to talk about heuristics and knowledge compiled from experience that has been accumulated. Additionally KEs may introduce misunderstandings when transferring the acquired knowledge into a computer representation. Much of the communication and interpretation problems can be eliminated

by having the expert enter knowledge directly into a system and thus relieving a knowledge engineer from many of the activities.



modified from [BREUKER87]

Figure 2.1

The BOTTLENECK

DEFINITION

The knowledge acquisition process is time consuming and as such has become the major obstacle to the development and deployment of effective knowledge-based systems. The idea of a "knowledge acquisition" bottleneck constitutes more than just getting knowledge from a domain expert in order to achieve expert performance. It also includes gathering background knowledge and converting all collected information into a usable format that closely models the problem domain. One of the most difficult tasks is helping the expert to structure the domain knowledge in order to define and formalize the domain concepts. In many cases a model must be defined where none previously existed.

Knowledge acquisition has accounted for a major part of the overall development time and expense of most knowledge-based systems. The nature of this problem is amplified in the maintenance, refinement and validation stages of the system life cycle. Consistency becomes a problem as an existing knowledge base is changed. Unforeseen concerns may arise from incomplete knowledge which may also involve "inconsistent" knowledge. The bottom line is that knowledge elicitation affects the cost-effective development of knowledge-based systems. Also, performance in terms of reliability, validity, and utility depend upon the reliability, validity, and accuracy of the elicited knowledge. Not only is the knowledge acquisition process time and labor intensive, the need to validate the information gathered for completeness and accuracy compounds the problem.

ISSUES

The issues involved with the knowledge bottleneck mostly revolve around multiple knowledge sources and communication or translation problems. Knowledge sources can be categorized as process (strategies or procedures) or content (facts). Both may be needed, but one may be dominant in a given application. Knowledge sources may also be classified as human or static (manuals, documents, instructions, etc.). The problem with multiple knowledge sources emerges as conflicting knowledge. There must be a way to combine knowledge from multiple sources and to resolve any conflicts. Problems concerning communication and translation may relate to the background knowledge that a knowledge engineer possesses concerning the domain. A knowledge engineer may not understand the domain and the terminology used. The proper vocabulary must be defined and refined to a useful state. This makes the act of eliciting the "right" knowledge very difficult. In cases where there is some familiarity with the domain, the knowledge engineer may make assumptions, draw conclusions, or filter out knowledge thought to be irrelevant. Other problems may arise because of the personality traits, opinions, or beliefs of the people involved. All of these problems have a direct impact on the system construction, its maintenance, and the testing of the acquired knowledge.

Eliciting problem-solving knowledge from an expert is one of the critical problems in building expert or knowledge-based systems. A long series of interview, build, and test cycles are necessary before such systems can emerge. The pressing needs of business and industry for practical expert systems, coupled with the difficulty of labor-intensive approaches to

knowledge acquisition, have dictated the need for automatic knowledge acquisition systems. A speed-up in acquiring knowledge also mandates the development of tools to measure quality of the knowledge acquired. These tools should also be able to generate knowledge-based systems from the acquired knowledge. These will essentially eliminate any additional human translation process (acquired knowledge conversion to the underlying system representation).

KNOWLEDGE and KNOWLEDGE MODELING

A knowledge base is not the same as human expertise. Expert behavior is not necessarily a designed sequence of activity. Each behavior is inductive, derived from the effect of all previous experiences. The knowledge acquisition problem is not only to "figure out the expert's model" (transfer of expertise) but rather to construct one where none, in principle, existed before [BOOSE89]. It is unclear how well manipulation of representations can approximate human reasoning. By definition, all models are imperfect approximations of reality. The act of modelling human expertise is very difficult and thus is another major contributor to the problems of the knowledge acquisition process. Modelling as used in this context refers to a transformation of knowledge from a human source to a usable computer repository. Currently little methodology is known about modelling and acquiring expertise. Plausible lines of reasoning can have little to do with actual problem solving. Problems can be amplified if experts are insecure, for one reason or another, or knowledge engineers are not trained properly in interviewing techniques.

The following is a list of possible problems when modelling and acquiring knowledge [BOOSE89].

- Different answers may arise for the same problem if it is presented in various ways
- Reliance on an intuitive representative match between evidence and the mental model of situation
- The likelihood of something happening may be judged only by how easily examples of it come to mind
- An overestimation of probability of future scenarios that are constructed from a series of individually probable events (mentally summing predicted outcomes rather than multiplying them together) may occur
- Background data is often ignored when predicting the probability of an event
- Uncertainty is often suppressed; people tend to think they know more than they do and that what they don't know must be unimportant
- People tend to avoid risks when seeking gains, but choose risks to avoid sure losses ("Losses loom larger than gains")
- People tend to anchor on items that fall early in a sequence when reasoning about the entire sequence
- Easily available information is often over-valued
- The tendency is to use relative rather than absolute estimates of probability
- There is some belief that chance occurrences tend to be causally linked to events
- A tendency exists to be overly cautious
- The manner in which data are presented may affect the ability to retrieve related information
- After reaching a saturation point people may refuse to absorb or contribute anything else
- People see what they expect to see
- People believe in facts because they are thought to be important, or because a fact is deemed important, it is believed
- The tendency is to believe that future random events are affected by past random events
- Habits lead to the use of past successful strategies whether or not they fit new situations
- People change their memories of events in retrospect
- People tend to over generalize from small sample sets
- The tendency is not to explore distant regions in the problem space
- The first and last items in a sequence are remembered better than others

NEED FOR TOOLS

The automation of the knowledge acquisition process may eliminate many of the problems that exist. This is not to say that problems will no longer exist with the process. The idea of knowledge acquisition tools is to outline a conceptual framework and develop and validate the acquired knowledge. The users of these tools include knowledge engineers (or AI programmers), programmers, and more importantly the domain experts. The use of knowledge acquisition tools will not completely eliminate the need for a knowledge engineer, but it will greatly change the role of one and put more accuracy into the acquired knowledge. A user of knowledge acquisition tools may see the structure of the representation, the problem-solving strategy, or the domain model where previously this could not be accomplished. An expert using a knowledge acquisition tool will eliminate most of the "noise" that otherwise may have been introduced. In addition the resultant system will be more representative of the expert's view of the domain. This may be partly attributed to the fact that the expert is more involved and committed as a result of a more responsible role. All of these reasons lead to time savings, improved efficiency, well defined methods, and at the same can put a wealth of expressive power in the hands of an expert. Automated knowledge acquisition tools can make development of knowledge systems much easier and give the expert a sense of ownership and thus greater motivation to build a system.

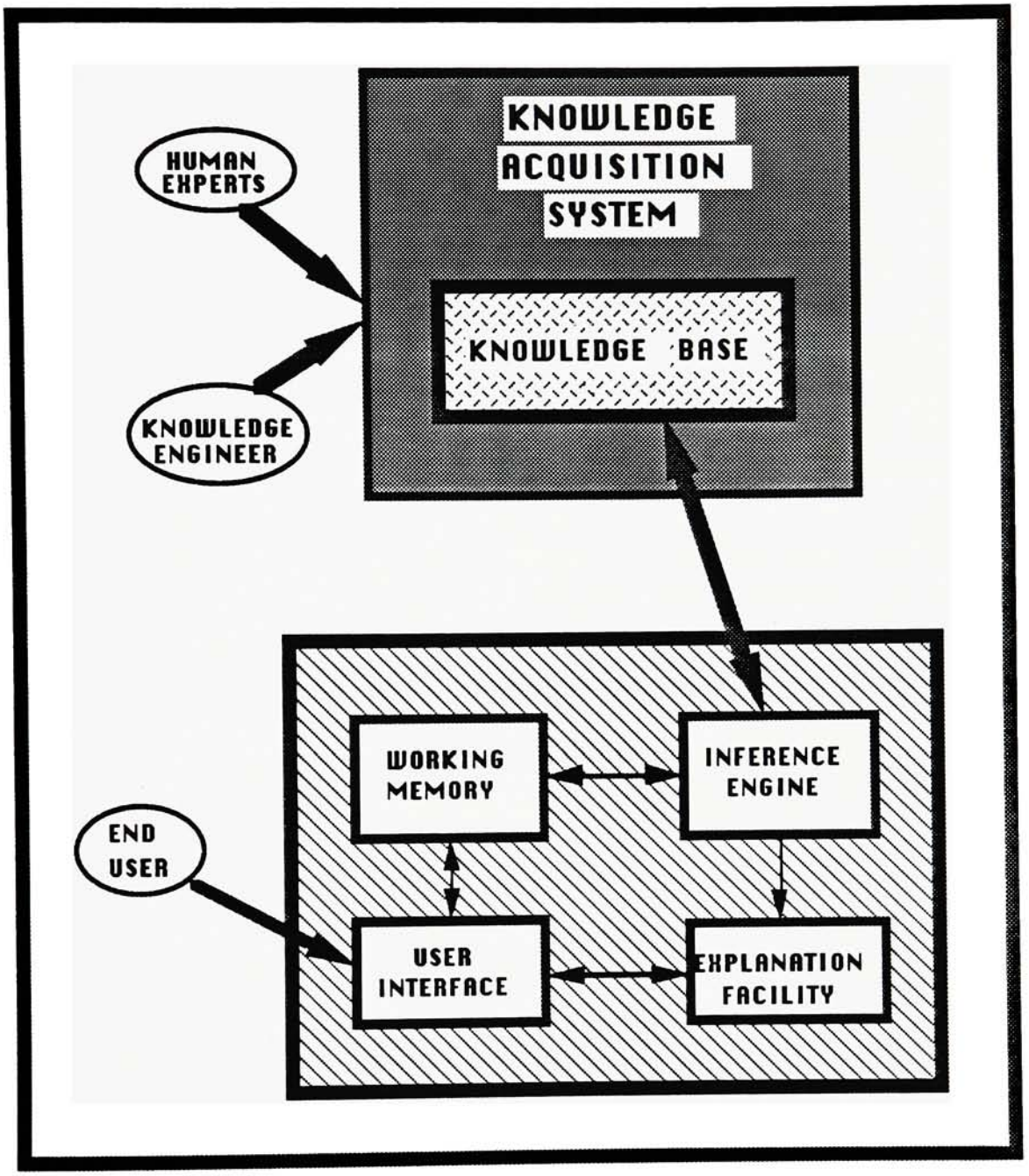


Figure 2.2

Knowledge acquisition tools should be developed that help experts capture their knowledge, allow knowledge engineers to capture knowledge more effectively, provide methods of automated induction, provide facilities for managing a data

base of test cases and evaluating them to measure the quality of the captured knowledge, and also be able to handle other issues such as uncertainty, ambiguities, conflicts, and multiple knowledge sources. The basic components of a knowledge acquisition tool are those for editing, displaying, and validating the knowledge base. While keeping all of this in mind, the tools and techniques used need to be integrated with the overall system and not treated as separate components. Automated knowledge acquisition can be accomplished through various interviewing techniques; learning by interaction, example, or case study; or learning via induction. Other fields within artificial intelligence may come into play during these techniques. For example speech recognition may be used to translate and interpret audio recordings; natural language translation may be used for written or typed input; or vision/pattern recognition may be used for video recordings. This is in addition to any translations done by a knowledge engineer.

The use of these tools may and should lead to many benefits and side-affects. In addition to cutting overall costs and development time, other important issues are addressed via automated knowledge acquisition. Verification and validation becomes easier and thus results in a higher quality system. Knowledge editing and maintenance also become easier, more accurate, and consistent. Overall completeness becomes a benefit of using consistency checking and induction techniques. Various forms of learning are also promoted. Another issue that is not often considered deals with the consideration of human factors. More attention is placed upon the user interface and other related components such as the help and explanation facilities.

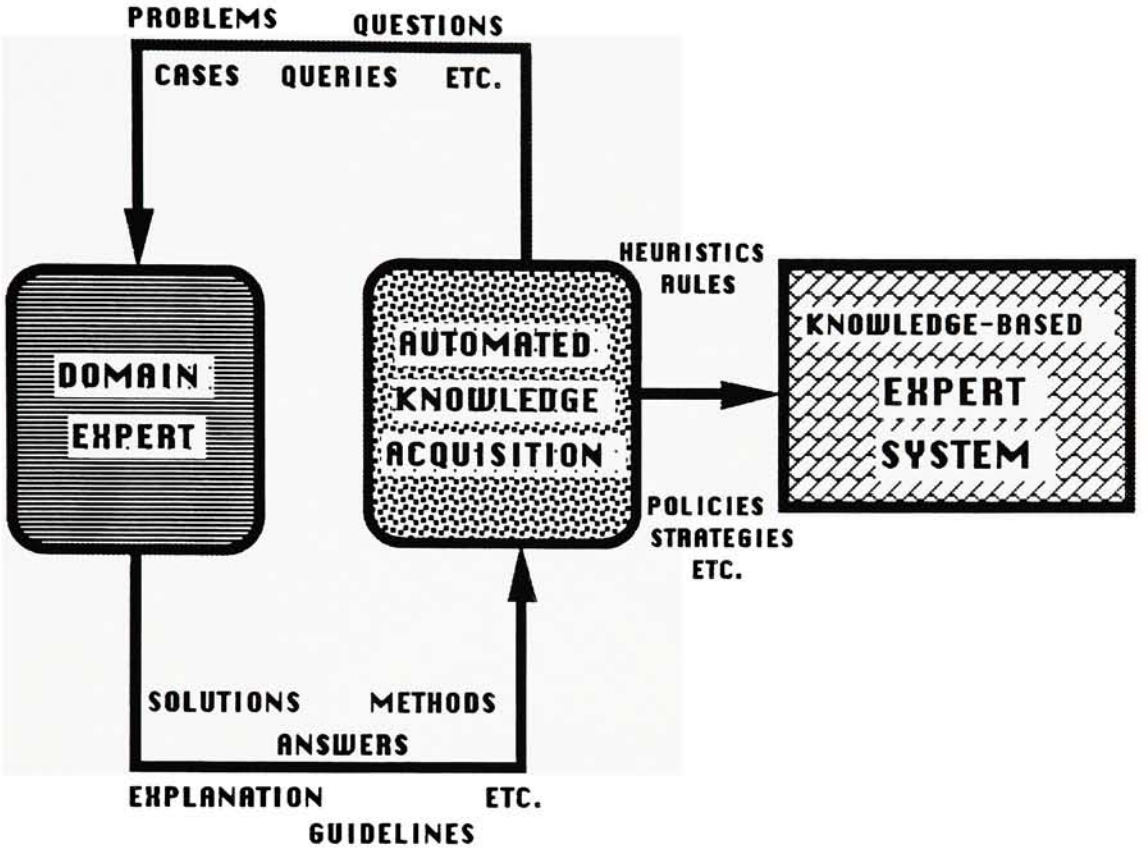


Figure 2.3

KNOWLEDGE ACQUISITION TRENDS and CURRENT RESEARCH

OBJECTIVES

There are many research areas involving knowledge-based systems. This research has the goal of making knowledge-based systems perform the tasks of human experts. The issues being addressed involve creating systems that would be capable of 1) explaining results in a manner suited to the audience, 2) learning and improving behavior,

3) restructuring stored knowledge, 4) stretching the current rule set, 5) determining the relevance of knowledge, 6) graceful degradation at domain boundaries, 7) reasoning about space and time, 8) reasoning from underlying problem principles, and 9) reasoning by analogy. These issues all contribute to the major limitations of knowledge-based systems. These limitations include: 1) the long and expensive knowledge acquisition process, 2) a lack of people availability from the domain and technical areas, and 3) the "Brittleness Factor". The "Brittleness Factor" concerns the issue that applications fall apart outside their immediate domain.

STRATEGIES

There are several research strategies concerning knowledge acquisition (KA). One strategy is to find and clarify KA strategies for a problem-to-method relationship. This usually involves a domain specific problem which uses a specialized method having much domain knowledge or a general problem employing a general method with little domain knowledge. Another strategy picks a problem, finds and develops KA strategies for an applicable method and then tries to apply the method and strategies to other problems. A third approach is to develop languages for defining and describing problems and methods. Yet, another approach takes a look at building intelligent editors to help AI programmers construct large knowledge bases. Much of the work within these approaches are looking at computer system solutions that help acquire knowledge. All of these strategies either develop new techniques or enhance and expand on current technology.

AREAS of RESEARCH

Future developments and trends in knowledge acquisition involve several, highly related areas.

> Automation - interactive support of knowledge elicitation methods

> Integration with Shells - integration of knowledge acquisition tools with Knowledge-Based System shells

> Elicitation from Multiple Experts - comparison of conceptual structures from several experts for consensus, conflict, correspondence & contrast analysis

> Improved Validation - recognition that the quality control of knowledge bases is part of the knowledge acquisition process and the development of validation methodologies and tools

> Integration of Machine Learning - integration of knowledge transfer tools for interviewing Experts with empirical induction tools for modeling cases

> Integration of Text Analysis - recognition that much knowledge is encoded in text and the integration of text analysis and clustering tools

> Convergence with Hypermedia - recognition that Hypermedia tools support Knowledge Acquisition and the integration of Hypermedia with more structured knowledge acquisition tools

> Convergence with CASE - recognition that knowledge acquisition tools are Computer-Aided Software Engineering for Knowledge-Based Systems and subsequent integration with existing CASE tools

AUTOMATION

Neither manual nor tool-based approaches have yet made a significant impact on the knowledge acquisition bottleneck which continues to impede the development and effective application of knowledge-based systems. Current research attempts to establish a new paradigm for knowledge acquisition, one that recognizes the wide diversity of forms and sources of knowledge, its essential dynamics making it subject to contradiction and change, and the constructive nature of knowledge acquisition whereby it is often the knowledge acquisition process itself that creates knowledge. The presence of Knowledge-Based Systems have been solidified and issues of problems and bottlenecks with the technology must be addressed. Given that knowledge acquisition is inevitably the bottleneck of KBS projects, methods and procedures must be developed that will improve this process. It must become easier and faster to gather the information required to build systems.

INTEGRATION

As research advances and KBS tools advance, knowledge acquisition tools and techniques will become more acceptable and usable. Through experience, the results will become more effective. Also, the techniques, whether they be manual or automated, will be capable of acquiring more complete knowledge. The discussion here has attempted to give a brief overview of some of the research work going on in this arena. Many techniques and procedures have been suggested and tried. As these techniques mature, more and more will become available as an automated tool. In some cases knowledge engineers will no longer be needed, but in others they will play different roles. As with all other advances,

The 80/20 rule often applies - the last 20% of information often costs 80% of the entire project. The scope of the application must be clearly defined with hard boundaries so that it is easier to determine if the required knowledge has been captured. Getting into issues like common-sense knowledge can cloud the picture really fast.

MACHINE LEARNING

Machine learning, which focuses on "learning" of knowledge from experience, is an important subject on its own. Learning and transfer of knowledge have much in common. Learning is the process of updating known knowledge. In this sense, it is an extension of the knowledge acquisition process. In the future these two aspects of knowledge-based systems will become increasingly more integrated and practical. The techniques of machine learning will be a major factor in keeping knowledge-based systems of the future useful and beneficial.

TEXT ANALYSIS

A significant component of human knowledge has already been expressed in text, diagrams, and pictures and has been captured in many places. The extraction of knowledge from such material is a major research area. Breakthroughs in this area should have major relevance to knowledge acquisition and may be expected to produce methodologies and tools that will be integrated with other techniques. Simplistic forms of text analysis has already been incorporated into knowledge elicitation tools (KSSO, KITTEN, KRITON,...). As these techniques are developed, their use

and proliferation will greatly help the knowledge acquisition process. Once reliable text analysis methods are in place, pools of knowledge will be available and accessible that were not even known before.

HYPERMEDIA

Hypertext and hypermedia are computer-based extensions of existing media such as books and films. This approach provides a non-structured way to link information that is a more natural flow than sequential processing. This technology allows fast keyword-type searches and has several important uses. Besides the capability to provide improved help and explanation functions, it can also be used for knowledge acquisition. A number of knowledge acquisition tools have already been written in Apple's HyperCard and this trend may be expected to continue [BOOSE89]. Hypermedia may also play an important role in representation of "acquired" knowledge. The integration of knowledge-based systems, knowledge acquisition systems, and hypermedia systems appears to be a major trend.

Another important issue is the support of knowledge acquisition methodologies through the use of hypermedia tools. Hypermedia is particularly important in knowledge acquisition because much human knowledge is implicit in action and not formally represented and processed. Hypermedia goes beyond knowledge bases in not requiring that the information system be able to represent and process the knowledge encoded in the media used. They subsume knowledge bases but extend knowledge representation to include any media where human knowledge can be transferred even if there is no means for analyzing that knowledge or its transfer.

For example, a film of a skilled person performing a task which enables another person to perform that task better contains knowledge which can be accessed and transferred through a hypermedia system (Vickers & Kingston 1987).

CONVERGENCE with SOFTWARE ENGINEERING TOOLS

Most AI systems can be thought of as just complex, sometimes abstract software systems that could have been done using conventional approaches. Software engineering for knowledge-based systems development has developed outside the framework of most existing software engineering methodologies. The "artificial intelligence" field has been perceived as different enough as to warrant its own approaches under such titles as "knowledge engineering" and "knowledge acquisition". Though there are differences, many past problems in software engineering are now being addressed in knowledge engineering. Many things happening with CASE (Computer-Aided Software Engineering) tools are also going on within artificial intelligence. One example can be seen by comparing the emergence of **automated knowledge acquisition tools** (which also code rules or other objects) with that of mainstream **code generators**. Both of these tools take as input some form of knowledge, requirements, or specifications. Another example is the relationship between data design tools using entity-attribute-relation models and the repertory grids (entity-attribute relations) used in various knowledge elicitation tools. Technologies used within the AI field are considered rather immature compared to software engineering methods and technology. AI methods can learn a lot from traditional systems techniques especially where AI systems interact with existing systems. It appears that AI can gain a lot in regards to maintenance

and validation by using some form of software engineering approaches. An important issue open for research concerns interfacing existing AI tools with traditional systems and databases. Certainly AI can learn a lot from traditional approaches and vice-versa. The traditional software engineering areas are taking advantage of AI approaches. For example, some CASE tools internally apply AI techniques.

KNOWLEDGE ACQUISITION METHODS

A Survey of Knowledge Acquisition Techniques, Methods, and Procedures

Many knowledge acquisition methods and techniques exist. All of them have their place and often must be used in combination. The bottom-line is that no one method can work in all situations. There are three factors that affect the choice of knowledge elicitation techniques: 1) the nature of the source of knowledge, 2) what form the knowledge takes, and 3) what is allowed to drive the selection of relevant information and the elicitation process itself. Manual knowledge extraction techniques may categorized as on-site observation, problem discussion, problem description, goal decomposition, pure reclassification, problem analysis, system refinement, system examination, and system validation. These methods of knowledge acquisition are briefly described below.

> On-site observation - watch the expert solving real problems on the job; the expert may verbally describe the problem-solving behavior

> Problem discussion	- explore the kinds of data, knowledge, and procedures needed to solve specific problems
> Problem description	- have the expert describe a prototypical problem for each category of answer in the domain
> Goal decomposition	- a problem reduction approach is taken in which the expert enumerates goals and categories of goals
> Pure reclassification frame analysis)	- rules are formulated for classifying observables into more specific objects and activities
> Problem analysis	- present the expert (or let the expert choose) with a series of realistic problems to solve aloud, while probing for the rationale behind the reasoning steps
> System refinement	- have the expert give a series of problems to solve using the knowledge acquired during previous interviews
> System examination	- have the expert examine and critique the prototype system's rules and control structure
> System validation	- present the cases solved by the expert and prototype system to other outside experts for review

Several manual KA techniques and procedures have emerged over the years. Many of these are summarized by the following.

<u>Brainstorming</u>

> Crawford Slip Method	- rapidly generate a large number of ideas
> Q-Methodology	

Interviewing

> Unstructured Interview - ask general questions and hope for the best, recording as much as possible

> Semi-Structured Interview - interview with open questions and a list of topics to cover

> Structured Interview - interview with a strict agenda and list of specific questions relating to features of the system

> Teachback Interview - a knowledge engineer demonstrates understanding of expertise by paraphrasing or solving a problem

> Tutorial Interview - the expert delivers a lecture

Knowledge Organization Techniques

> Card Sorting - sort objects on cards to help structure knowledge

> Overcoming Bias - recognize and correct bias from knowledge sources

> Psychological Scaling - use scaling techniques to help structure knowledge

> Uncertain Information - expert encodes uncertainty about Elicitation the problem

Protocol Analysis Techniques

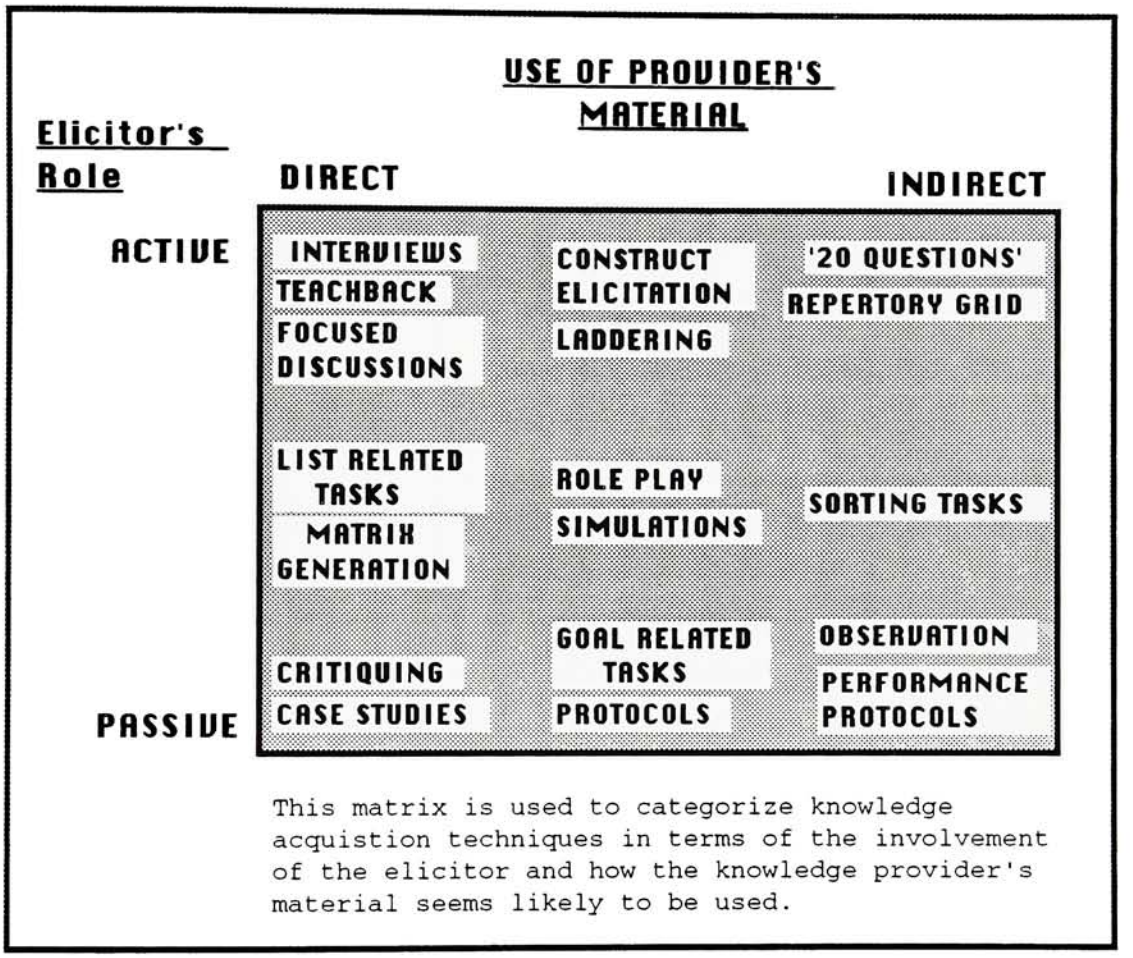
> Participant Observation - the knowledge engineer becomes an apprentice or otherwise participates in the expert's problem-solving process

> Protocol Analysis(Case Walk-Through/Observation/Process Tracing) - record and analyze transcripts from experts thinking aloud during tasks

User Interface Techniques

> Wizard of Oz Technique - an expert simulates the behavior of a future system

Figures 2.4 and 2.5 summarize many of the methods and techniques available as well as their associated use.



modified from [CORDINGLEY89]

Figure 2.4

METHOD \ KNOWLEDGE TYPE	FACTS	CONCEPTUAL STRUCTURE	CAUSAL KNOWLEDGE	RULES	WEIGHT OF EVIDENCE	PROCEDURES	EXPERT STRATEGY	SYSTEM STRATEGY	CONTEXT OF RULES	EXPLANATION	JUSTIFICATION
INTERVIEW	Y	Y	Y	*	*		N		*		Y
TALK-THROUGH CASE STUDIES	Y	N	Y	Y	N	Y	N			Y	
OBSERVING							Y			Y	
PROTOCOL ANALYSIS						Y	Y				
CARD-SORTING		Y									
SCALING TECHNIQUES		Y									
REPERTORY GRID		Y	N	Y	Y	N	N				
INDUCTION			N	Y							
TASK ANALYSIS								Y			
USER INTERVIEWS								Y			
EXAMINING PROTOTYPE				Y		Y		Y		Y	

* - only partially derived

The acquisition method used will depend upon the intended use of the knowledge. More than one method will be necessary in order to get enough information for a complete system.

modified from [WILSON89]

Figure 2.5

The INTERVIEW

Most forms of gathering information take the form of an interview. There are several types (structured, unstructured, focused, unfocused, etc.) of interviews that can be used. Each type of interview can use various techniques to gather the needed information. Interviewing relies on verbal data and the act of putting knowledge into words. This can be a difficult activity. Other knowledge collection activities may consist of reading manuals and other documents, observing current users and experts at work, analyzing past cases, and a host of similar activities.

The interview process is the most commonly used technique for knowledge acquisition. The interview process is used to identify tasks and major concepts and to structure and refine already-acquired information. A knowledge engineer conducting an interview must decide not only on the type of interview to be used and the line of questioning, but also, what note taking approach to use. Paper and pencil, video taping, or audio taping are available options.

Structured Interview

Advantages:

> Forces organization on the interview
> Very goal-directed
> Attempts to remove distortion from experts subjectively
> Allows better integration of material after the interview
> Forces the expert to be systematic
> Knowledge engineer identifies gaps in the knowledge which acts as a basis for questions
> Purpose of the session is clear to the expert

Disadvantages:

> Needs more preparation by the knowledge engineer

> Knowledge engineer needs to study background material extensively

Unstructured Interview

Advantages:

> Appropriate when the knowledge engineer wants to explore an issue
> Facilitates description of domain in a way that is easy for the expert
> Goal is to establish rapport and to get a broad view

Disadvantages:

> Data acquired is often unrelated and difficult to integrate
> Often exhibits lack of structure
> Does not allow gathering of specific knowledge
> Takes time and training to do well
> Similar questions asked in future sessions may annoy expert

STRUCTURED TECHNIQUES

Interviewing should be a planned event in order to be productive. The direction and kinds of questions should be designed before a session begins. One approach is to use LaFrance's Acquisition Grid [BOOSE89],[CORDINGLEY89]. This grid is composed of six question types and five "forms of knowledge". The question types consist of "Grand tour", "Cataloguing Categories", "Ascertaining Attributes", "Determining interconnections", "Seeking Advice", and "Cross-checking". The "forms of knowledge" model the ways in which experts can represent their know-how. The theoretical foundations of this dimension of the grid include Minsky's society-of-the-mind theory which proposes that intelligent action emerges from the interactions of many small systems operating within an evolving overall administrative structure [CORDINGLEY89]. Knowledge is categorized under this idea as

layouts, stories, scripts, metaphors, or rules-of-thumb. See figure 2.6 for an example of LaFrance's Acquisition Grid.

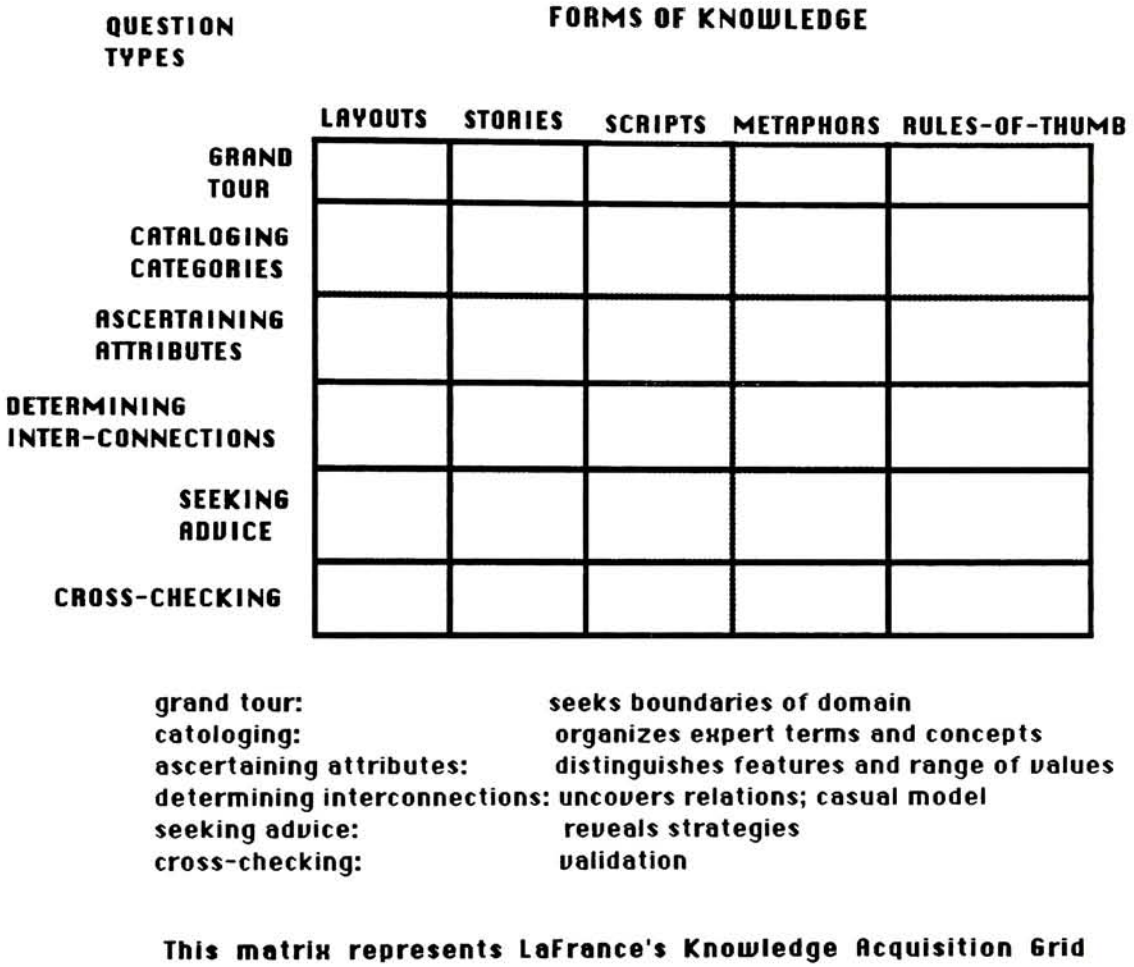


Figure 2.6

A *focused talk* is primarily used to gather information about tasks. This information may be from verbal reports, hands-on experience, or any other appropriate form. Case studies are often analyzed through forward scenario simulation (an example case is used and its handling is described step by step) or retrospective case description. Usually the more

"interesting" cases or ones with "critical" incidents make for better analysis. List-related methods require a list of all possible decisions and for each decision a list of the possible consequences. Other approaches are used to determine goal-related information. These approaches help divide the domain into subgoals by distinguishing goals, goal decomposition, and reclassification of these goals. *Critiquing* is another form of a focused talk. It requires that the person or system performing the critique be aware of alternatives. This technique is useful in domains where there is no single 'right' way of doing things. The technique provides a chance to identify and correct errors. It is a way to focus on positive aspects of a solution. Also, it provides an opportunity to indicate missing knowledge.

A couple of role reversal techniques also exist. One, the *teachback* procedure is primarily a feedback and verification technique. The idea is to teach back to the expert, the knowledge acquired in the interview. Another technique termed the "*20 questions*" approach is context-focusing. A Knowledge Engineer (KE) selects a situation, diagnosis, fault, problem, solution, state, or outcome. The expert then asks questions that are answered by "yes" or "no." This is a role reversal where the expert probes the KE and it is the KE who responds. The KE must know the domain well enough to select meaningful targets and provide correct answers.

NON-STRUCTURED TECHNIQUES

The analysis of structured material can make use of many existing methods. For example cluster analysis, factor analysis, principal component analysis, multidimensional

scaling, and machine learning techniques (eg. induction) may be used. However, there is no knowledge acquisition method currently for analyzing unstructured interview data. Data in a natural state is normally long and complex and arranged in a non-standard format. In addition, expert testimony usually relies on unspecified assumptions, background meanings and implied knowledge. It therefore seems contrary to not have well-defined methods to analyze unstructured information. Those that exist are still developing.

Some of the non-traditional structured methods for knowledge acquisition include case analysis, protocol analysis, critical incident analysis, commentaries, repertory grids, prototypes, and conceptual graphs. A case analysis collects comments on how past cases were handled. The use of protocol analysis involves reading through transcripts and categorizing and analyzing what is said or performing analysis of specific behaviors. Critical incident analysis is handled through discussions of interesting, difficult, memorable, and/or funny cases. Commentaries are a running commentary of current work. The use of Repertory Grids is used to explore one's thinking around a subject. The use of conceptual graphs is just like any other picture. "A picture is worth a thousand words." Work with an initial prototype system is a powerful way to get people to talk. The following will briefly cover these and other methods and procedures of elicitation.

The use of *protocols* provide much information about the performance of the task. A standard description of the task is acquired. A protocol is basically an externally made record (audio/video) of a task being performed. It may capture words and actions. A byproduct of the session may be artefacts such as completed forms that are made while

performing the task. Protocols can be of several types. These include "think aloud", "talk aloud", "eidetic reduction (relating observations about one's behavior - critique), retrospective reporting (relating remembered aspects of the activity), behavioral descriptions, or playback (accounts of behavior) [CORDINGLEY89].

Activities such as *role play* can also be beneficial. An expert assumes a role and enacts a situation in which the expertise is needed. The KE in this situation is only an observer. Another form, *simulations*, does involve the KE. The disadvantage with this technique is creating a real-world situation. Some bias and misinterpretations can be included in information acquired as a result.

The use of *construct elicitation* (which includes Personal Construct Theory and the Repertory Grid technique) is another technique which is rapidly being accepted. The Personal Construct Theory was originally developed in the field of clinical psychology. Although data collected under this theory is primarily associated with the use of repertory grids, the data can be used in other analysis such as multidimensional scaling. There is no one correct format for grids nor only one way to use them. Construct elicitation may be solely a focus for discussion. In other cases, the constructs and data about how elements are placed on the constructs is used. Other analysis is limited to looking at the defined grid. The grid is best looked on as a particular form of structured interview. For good or bad, the grid is often used without regard for its underlying theory. The theory and detail of this technique will be covered in greater detail in a later chapter.

Laddering started in the context of personal construct studies and was made widely known by Fransella and Bannister [CORDINGLEY89]. The use of laddering has a history of use as part of the technique involving repertory grid tools. This technique has gained prominence as a KBS elicitation method valuable for generating various hierarchies of "concepts". Laddering is characterized as a *goal decomposition* technique since it can be used to generate many kinds of hierarchy (See figures 2.7a-2.7e [BOOSE86] for examples). To get "superordinate" concepts, the expert is asked WHY? questions. To get "subordinate" concepts, the expert is asked HOW? questions. Same level concepts are gathered by asking the expert for alternatives. It is recommended by many that only "safe" areas of inquiry be used and one should only ladder up one level.

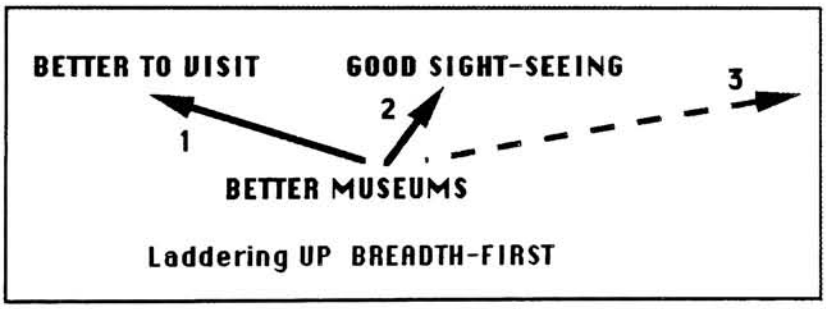


Figure 2.7a

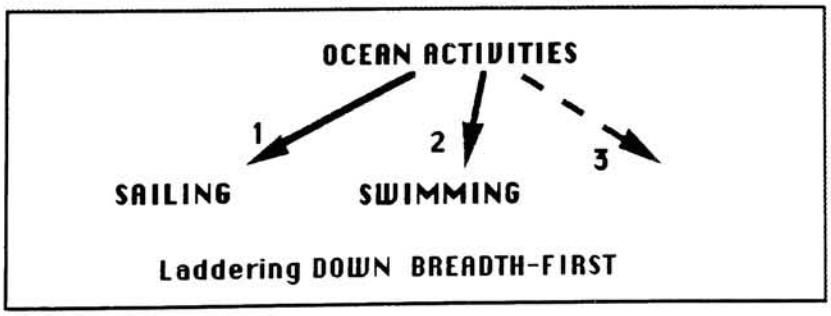


Figure 2.7b

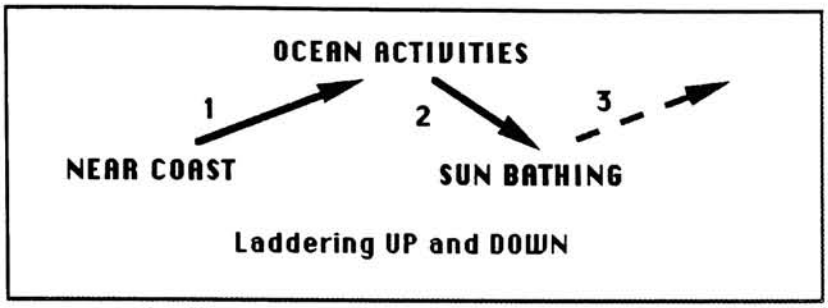


Figure 2.7c

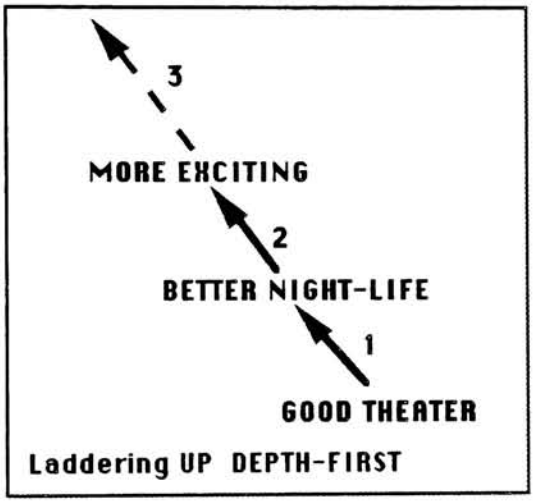


Figure 2.7d

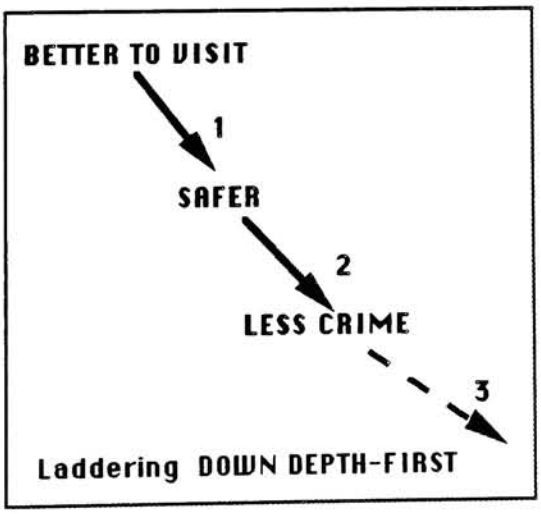


Figure 2.7e

HANDLING MULTIPLE SOURCES of KNOWLEDGE

The need to handle knowledge from multiple sources has lead to several specific techniques. One, The *Delphi Technique* has been used to elicit knowledge from multiple experts for classification problems. Information is derived from experts independently, the results are returned, experts re-evaluate their earlier decisions, and the process continues until a general degree of consensus is reached. Another group technique is the *Crawford Slip method*. Individuals in groups respond to target questions by writing answers on slips of paper in a certain format within a specified period of time. The format makes it easy to organize and classify responses. Repertory Grids also have been used in various ways to handle groups of users and multiple knowledge sources.

KNOWLEDGE ORGANIZATION

A technique utilizing elements of a domain to understand how the knowledge provider conceptualizes this world is "card sorting" (defined as a *sorting task*). The things to be sorted, the elements, are either elicited from the expert or taken from an analysis of the domain and are each written on a small card. If a binary tree knowledge representation structure is required, the expert is asked to divide the elements into two piles in a way which is meaningful to that person and to label each pile. Each of these piles are then divided and labelled repeatedly until they can not be meaningfully divided or until there is only one element left. An alternative is to give the person two cards and ask whether they are the same of different. They are then put

into appropriate piles (same or different). All remaining elements are looked at one a time and put into an appropriate pile or a new one. This is done until no elements are left. Another option is to give out all of the elements and ask that the expert divide them into the appropriate piles.

Matrix generation is useful when tabulating information. It is used to generate tables that are both two-way (row/column) and two-mode (2 sets of lists). Column and row headings can be defined in a variety of different ways (ie. symptoms & faults, etc....). Several traditional and "newer" methods have been used to capture data in a matrix or graphical form. These include repertory grids, conceptual graphs, decision/action trees or tables, flowcharts, flow diagrams, and logic diagrams.

COMMON ISSUES

No matter what method is being used, common issues exist. The kinds of knowledge must be identified. Developing glossaries or "knowledge dictionaries" is a big help in developing a vocabulary and keeping definitions consistent. The use of intermediate representations also help clarify knowledge and it often makes important things explicit. Proper representations can expose constraints, make the knowledge complete and concise, and also make it easier to handle the knowledge data while suppressing rarely used detail. Diagrams (just another form of intermediate representation) prepared to represent knowledge on the problem solution process make it easier for an expert to verify. Decision tables, decision trees, various grids,

flowcharts, and various classification hierarchies may be used.

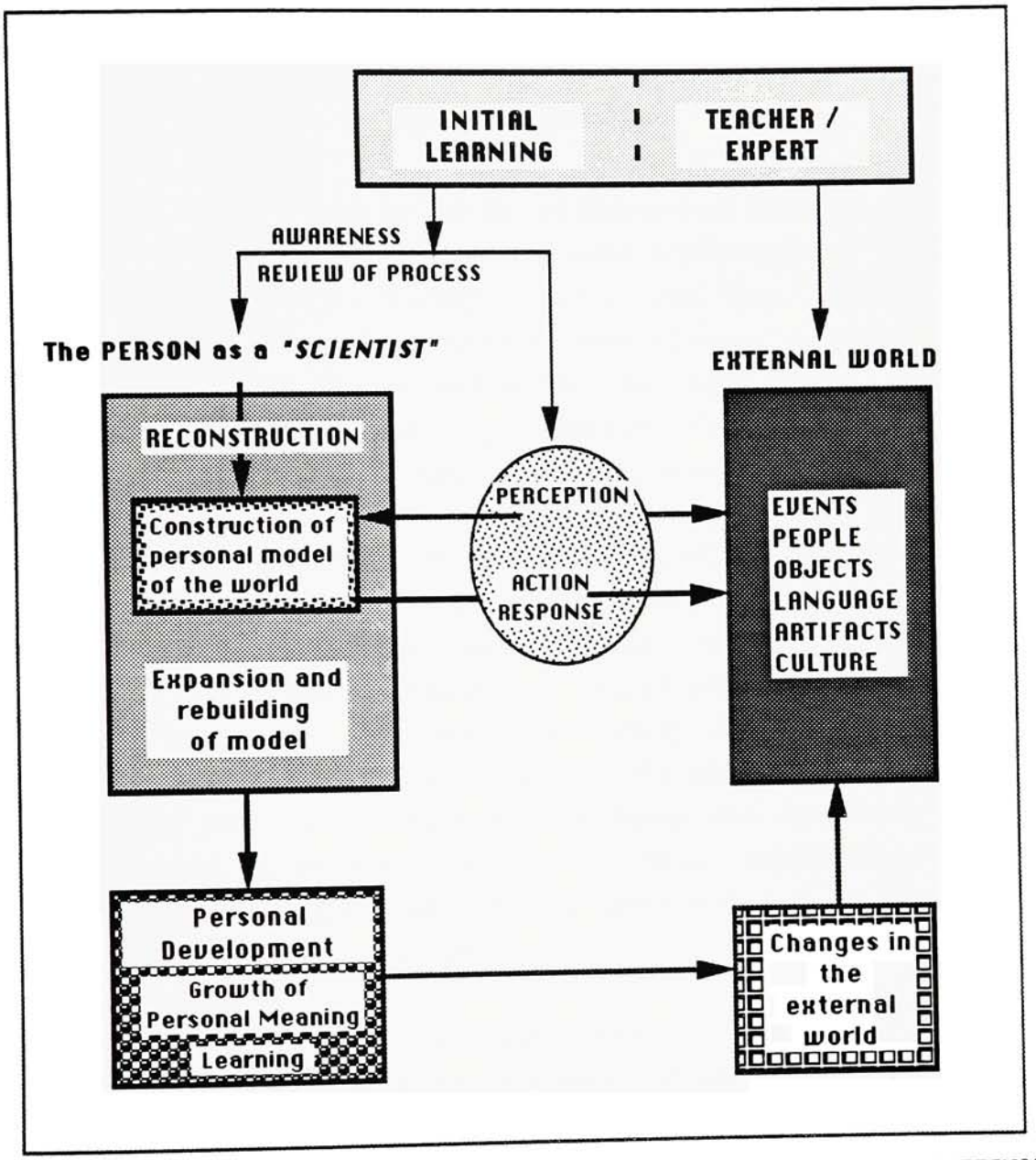
An important thing to consider when using any technique is that of objectivity. There is a natural tendency for people to bias information received. A KE must be careful not to influence the acquired knowledge based on his experience or understanding. The goal is to model the expert's knowledge and not the KE's view of it. In using many manual techniques, it is hard to eliminate bias completely. The use of an automated technique does remove the opportunity for subjectivity and bias, but it may not eliminate it. If KE bias and subjectivity is removed, the subjectivity of the expert may still remain. This may be an aspect of the knowledge which makes it "expert-level", but on the other hand it may be thought of as undesirable and detrimental to the intended meaning.

CHAPTER 3 PERSONAL CONSTRUCT THEORY

INTRODUCTION: What is Personal Construct Theory About?

Some psychologists are interested in how people categorize experiences and classify their environment. In this context, psychology is the study of the structure of experience and judgement. Experience in its raw form, according to personal construct theory, can't be used to make judgements about the world. With this point of view, experience is the data for perception. To understand how a person makes judgements and acts effectively, one needs to know how the person categorizes their experiences and classifies the environment. All human perception involves "categorizing". When people realize how this categorization is achieved, they can use that awareness to predict future events more accurately and act more effectively. The individual is also in a position to change his system and to adapt to specific needs of himself and others.

George Kelly's Personal Construct Theory (PCT), also known to some as the theory of a "personal scientist," was that each individual seeks to predict and to control events by forming theories, testing hypotheses, and weighing existing evidence [BOOSE86]. Anticipated outcomes then depend on the individual's interpretation of past similar events and personal experiences. "...a scientist is one who observes, construes relationships, articulates theories, generates hypotheses, ventures predictions, experiments under controlled conditions, and takes candid account of outcomes..." Kelly's attempt to explain how people get to be the people they are relies on the basic assumption that *Man is a scientist* (see figure 3.1). Everybody builds a system of hypotheses known as a 'construct' system.



[HARRI-AUGSTEIN85]

Figure 3.1

PERSONAL CONSTRUCT THEORY

BACKGROUND

The theory proposed by Kelly by no means has been accepted lightly. In fact, it has sparked some controversy. Some have regarded PCT as a meta-theory. Also, some have classified the theory as cognitive, while others have classed it as existential. Many have given other classifications. These include: an emotional theory. a learning theory, a psychoanalytic theory, a typically American theory, a Marxist theory, a humanistic theory. a logical positivistic theory, a Zen Buddhistic theory, a Thomisitic theory, a behavioristic theory, an Apollonian theory, a pragmatistic theory, a reflective theory, and no theory at all. It also has been classified as nonsense [KELLY70]. Still others have considered PCT as a *methodological theory [MAIR70]*. Much of this controversy in the psychology field may have something to do with the way the theory was introduced and defended. The theory seemed to be presented as something revolutionary rather than a formalism of various ideas about learning, perception, and anticipation.

Prior to 1955, personal construct theory (PCT) was in its infancy. Supported by a few students, Kelly worked in virtual isolation at Ohio State University to establish the outlines of his approach to clinical psychology and personality theory. From 1955-1966, the theory of personal constructs grew tremendously, primarily through a large British following, especially at the University of London. In the 1966-1972 timeframe a "publication explosion" began. A great many of articles were published which employed adaptations of the repertory grid technique. From 1972 to the present day, the British community relinquished some of its dominance and the United States began to increase its interest. At the same

time, however, other countries have moved into the arena [NEIMEYER85]. It is interesting to note that much of the more interesting work done relating to Personal Construct theory recently has involved non-psychologists.

A major revision of the theory and the primary technique used was proposed by Hinkle in 1965. He saw constructs as being defined by their implications and thus created the Implications Grid or Impgrid to quantify these relationships. These grids have no elements to construe. Each construct is paired with every other construct to see which implies the other. Hinkle also described *laddering* as a procedure for finding out the position of any construct or implication in the person's hierarchical construct system (a pole preference is sought to help the laddering process) [BANNISTER71].

PERSONAL CONSTRUCT THEORY DEFINED

The theory of Personal Constructs is *anticipatory* rather than *reactive*. Psychological response is initially and basically the outcome of a construing act. Experience is no guarantee of the validity of personal constructs, neither does the duration of experience give any such warranty. Personal constructs are the tools of experience rather than its products. People's behavior is taken as validating evidence for a variety of personal constructs. Everyone uses this approach. It should be apparent that a "role" is being defined and tried. In fact, personal-construct theory was originally called "role theory" [KELLY55].

PCT has as its base the fields of psychology and psychotherapy. The main thrust of personal construct theory is its recognition that psychology is man's understanding of his own understanding [BANNISTER71]. Construct theory treats

men as scientists. It is a broad, abstract theory which is not tied to any particular concept or thing. Kelly's ideas were to define the structure of knowledge and its growth rather than define the learning process, which can be seen as a subset.

The Psychology of Personal Constructs, as defined by Kelly, is based upon one fundamental principle and eleven supporting corollaries.

FUNDAMENTAL POSTULATE:

A person's processes are psychologically channelized by the ways in which events are anticipated

CONSTRUCTION COROLLARY:

A person anticipates events by construing their replications.

A person develops a physiological construct system. An abstract structure is erected which has some meaning. Constructs are erected that are of similarity and contrast.

INDIVIDUALITY COROLLARY:

Persons differ from each other in their construction of events.

Though there are individual differences in the construction of events, persons can find common ground.

ORGANIZATION COROLLARY:

Each person characteristically evolves, for his convenience in anticipating events, a construction system embracing ordinal relationships between constructs.

Different constructs may lead to conflicts. Everybody handles these conflicts differently. Constructs are systematically arranged in hierarchies to abstract them further. Constructs can be *evaluative* or *descriptive*

DICHOTOMY COROLLARY:

A person's construction system is composed of a finite number of dichotomous constructs.

An aspect or abstraction both determines similarities and differences. All constructs follow the basic dichotomous form. Inside its particular range of convenience, a construct denotes an aspect of all the elements lying within. This system is entirely of constructs; its organizational structure is based upon constructs of constructs. Most everyday thinking is based upon likenesses and differences. This goes along with nonparametric mathematics.

CHOICE COROLLARY:

A person chooses for himself that alternative in a dichotomized construct through which he anticipates the system.

A choice made is elaborative. It can be constricted and based on certainty or broad in scope which may increase understanding. There is a continual movement toward the anticipation of events.

RANGE COROLLARY:

A construct is convenient for the anticipation of a finite range of events only.

Areas outside this range are not contrasts; they are irrelevant. A construct may be thought of as a concept by some, but it is more closely aligned with the term "percept." This term has carried the idea of its being a personal act.

EXPERIENCE COROLLARY:

A person's construction system varies as he successfully construes the replications of events.

New constructs are established when something unexpected happens. This is experience. This also gives an opportunity to re-visit previously established constructs. Learning is assumed to take place. It has been built into the assumptive structure of the system. To be effective, the construct system must have some regularity.

MODULATION COROLLARY:

The variation in a person's construction system is limited by the permeability of the constructs within whose ranges of convenience the variants lie.

The progressive variation must take place within the system. The permeability of a construct relates to the capacity to embrace new elements.

FRAGMENTATION COROLLARY:

A person may successively employ a variety of construction subsystems which are inferentially incompatible with each other.

A person's construction system is continually in a state of flux. New constructs are not necessarily direct derivatives of, or special cases within other constructs. It is sure only that the changes that take place from old to new constructs do so within a larger system. Some inconsistency is tolerated.

COMMONALITY COROLLARY:

To the extent that one person employs a construction of experience which is similar to that employed by another, his psychological processes are similar to those of the other person.

It is not the similarity of experience which provides the basis for similarity of action, but similarity of their present construction of that experience.

SOCIALITY COROLLARY:

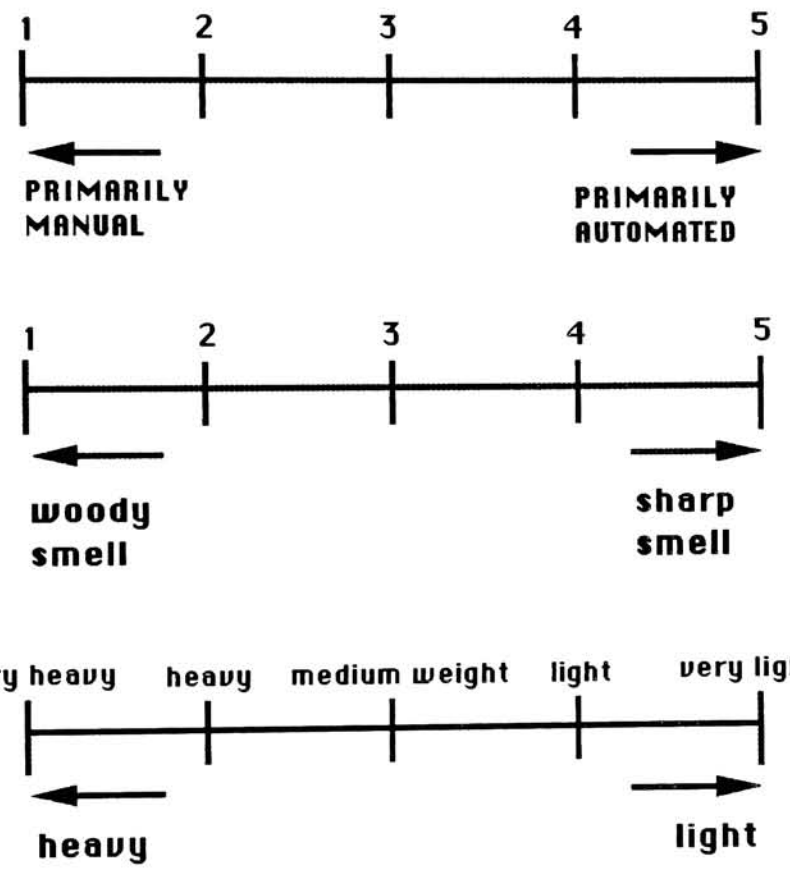
To the extent that one person construes the construction processes of another, he may play a role in a social process involving the other person.

CONSTRUCT DEFINITION

To construe events is to abstract them in order to make sense out of them. Each person goes at it in his own way. *Constructs* are the channels in which one's mental processes run. Constructs are also known as attributes or traits. They are used to reach conclusions and anticipate events. A 'construct' is a mental tool which allows a person to discriminate between elements of the world. Constructs are at different levels of awareness. Some can be clearly articulated and tied to words while others are more vaguely represented by action or emotion. The most important constructs are called *core role* constructs: "One's deepest understanding of being maintained as a social being is his concept of his core role"[KELLY55]. An evolving construct system responding to a requirement change often proceeds by propositions of the form "what if" or "let me look at it as if." They are a device for asking about the implications of construing an event in a particular way. Viewed in this way, a plan or a design can be treated as an indication of an evolving construct system [STRINGER77].

Constructs are given verbal labels and may be highly subjective (they are highly personal and are not fixed). Since constructs are personal, they may be factual, imaginary, emotional, or whatever is important to the person identifying them. Individuals perceive the world with different construct systems and therefore any two people may interpret the same event differently. However, individuals in a common culture may have many similar constructs. The notion of constructs is similar to the notion of *contrasts* used in Spradley's Contrast principle. This principle suggests that the meaning of a symbol can be discovered by finding out how it is different from other symbols [CORDINGLEY89]. The construction system may be considered a system of controls. Construct systems

control the role one plays in life. The things or events which are abstracted by a construct are called *elements*. The context of a construct is composed of all those elements to which the construct is ordinarily applied [KELLY55].



Bipolar CONSTRUCTS allow subjective ratings of elements; they facilitate comparison and ranking. Ratings are given which place elements appropriately in relation to the construct poles. Various rating scales can be used (ie. 1/0, yes/no, 1-5, 1-7, etc.

A direct comparison can be made between the rating scales used and FUZZY LOGIC. Each value on the rating scale can represent a degree of membership.

Figure 3.2

The range of *convenience* relates to relevance of a construct to a particular event or element. Each construct has a limited range of *convenience* because it will only be applicable to a limited set of objects. These constructs are also viewed as parts of a dynamic network, rather than in isolation, and this network of constructs may shift and be re-organized over time. Constructs that are more important or general than others are referred to as *superordinate* and those that are more specific are called *subordinate* (see figure 3.3). Kelly defined a superordinate construct as one which includes another as one of the elements in its context and he defined a subordinate construct as one which is included as an element in the context of another. General constructs, closer to the core constructs, are more stable and resistant to change than lower order, more specific constructs. Core constructs can not be changed without disturbing the roots of a person's existence. These core constructs are more comprehensive than first order constructs and possess a wider range of convenience. Core constructs are more abstract. They are very difficult to validate [STEFAN77].

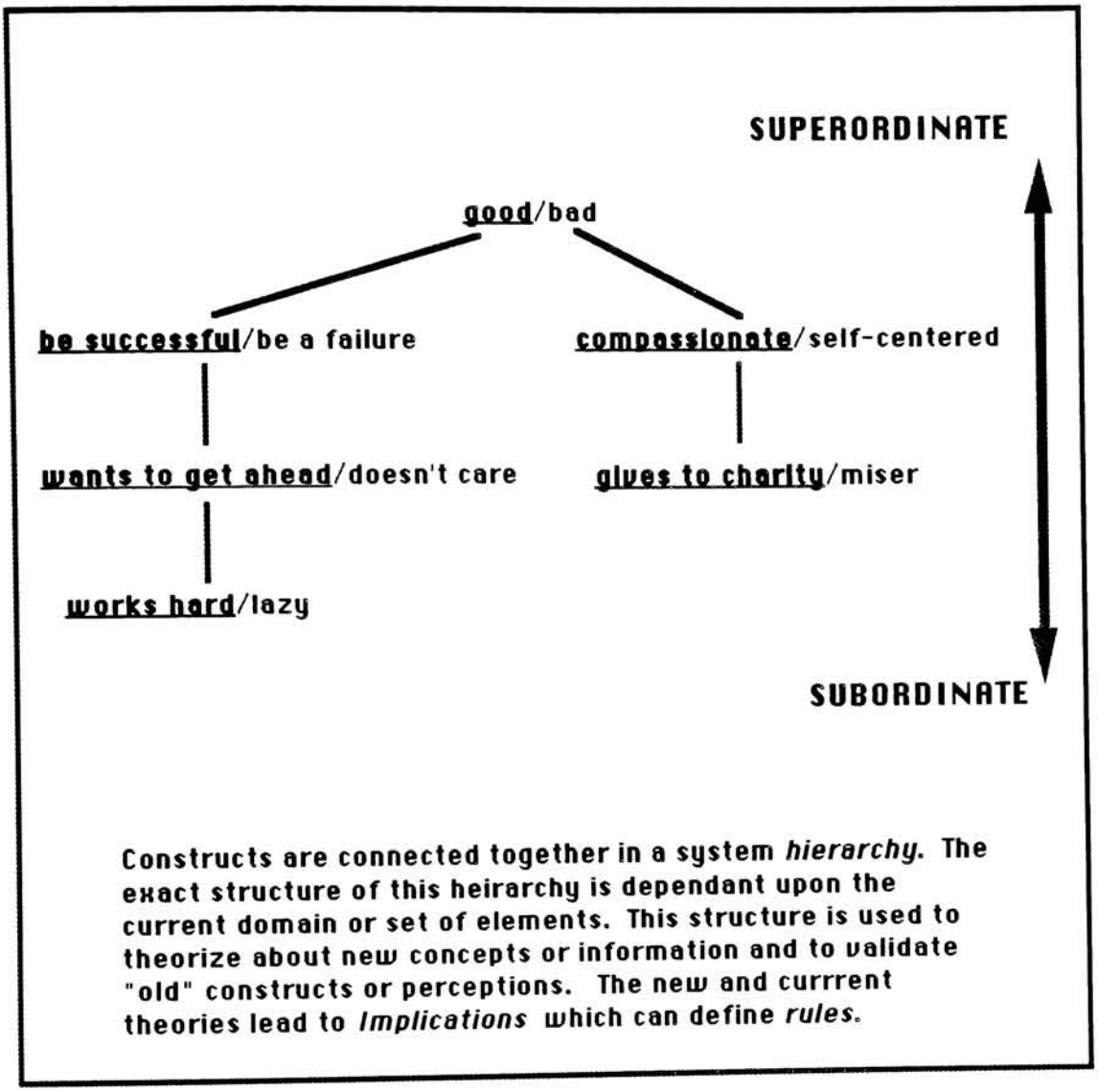


Figure 3.3

George Kelly originally stated six assumptions concerning constructs. First, constructs elicited should be permeable. This means that the person who has "defined" the construct is able to apply it to elements other than the ones used to elicit the construct. The second assumption states that pre-existing constructs should be elicited rather than develop new ones during the process. Another important assumption is that the verbal labels attached to the constructs are communicable. In other words, the meaning is understood by others. In

addition, constructs elicited should "represent the subject's understanding, right or wrong, of the way people look at things." Still another assumption states that the person giving the constructs should not disassociate himself entirely from the elements or from the constructs elicited. Finally, the constructs elicited should be explicitly bipolar. Either an element fits the label or it does not [FRANSELLA77], [KELLY55].

CONSTRUCT USE

The minimum context of three things and two relationships (likeness and difference) must exist in order to define a construct. Each construct involves two *poles* which a person finds useful in understanding the world. Each pole represents one end of a dichotomy. Every 'element' for which the construct is relevant can be characterized best by one pole of the construct or the other. The **emergent** pole of a construct is that which holds true for most of the immediate context. The **implicit** (contrast) pole of a construct is the one which contrasts the emergent pole.

Validation represents the compatibility between one's prediction and the outcome that is observed. If the prediction is disproved, new constructs may be formed or existing ones may be invalidated and thus necessitate change. Crucial to the theory is that people make sense of the world by anticipating events on the basis of their *personal construct system.* An experience which does not fit the anticipated pattern will cause the person to 'think again' about the constructs which lead to the violated anticipation. These 'surprising' experiences lead to a 'loosening' of the construct system. Experiences which confirm the expectation

support the existing system of constructs and 'tighten' the system. The "scientist" cycle within PCT can be viewed initially as loose and ultimately by tight construing. A tight construct leads to unvarying predictions whereas a loose construct is one which can lead to varying predictions, but can be identified as a continuing interpretation. Another way to think about the relationships between constructs is to consider them as inferences with varying degrees of probability [BANNISTER70]. New constructs can be built with the use of "fresh" elements, experimentation, and the availability of validating data ("knowledge of results facilitates learning"). The creation of new constructs are hampered under threat, preoccupation with old material (habits, etc...), or in situations where new things can not be tried and verified very easily.

Kelly developed one major method for eliciting constructs and several other analysis techniques. His original method was the Role Construct Repertory Test (known as the Rep Test). It was a method for exploring the way people make sense of the world. The Rep Test is an application of the concept-formation test procedure. It uses as "objects" those persons with whom the subject has had to deal in his daily living. The Rep Test is concerned with how details are dealt with, not merely the abstraction involved. It is also concerned with the subject's relations to particular people. Use and growth of the Rep Test has resulted in the use of the repertory grid technique. To some, including George Kelly, the introduction of the repertory grid with the original PCT may obscure the real meaning of the theory itself [HINKLE70].

The measurement for PCT is the construct. The repertory grid technique assesses the mathematical relationship between constructs. Self-characterization, another analysis technique started by Kelly, obtains a person's view of himself in a qualitative form. Self-characterization is the technique of

personal assessment. It is a format which invites the person to say something about himself. The aim is to find out how the person structures his immediate world and how he sees himself in relation to these structures and the strategies he has developed to handle his world.

ELICITATION: THE REPERTORY GRID

A repertory grid may be defined as any form of sorting task which allows for the assessment of relationships between constructs and yields its primary data in matrix form. The repertory grid preserves individual construct systems. It is a system of cross-references between personal observations or experiences of the world (elements) and personal constructs or classifications of that experience. The grid form of the Repertory Test is created by listing elements on one axis and constructs along the other. At the intersection of each row and column is a cell in which an indication can be placed to represent the applicability of the construct. Originally, the Grid form of the Rep Test included three parts (see the appendix A): 1) a sheet of instructions, 2) a list of titles, and 3) a sorting grid. The titles were those of various people (ie. family members) familiar to the person taking the test.

Several assumptions are made about the use of grid methods. It is assumed that elements elicited from a person would eventually form an adequate representation of the subject matter. The construct pairs developed on the grid are representative of those that the person must deal with in structuring their "roles." It is assumed that the person's internal conception of the construct does not change significantly while different elements are being rated. By

default, when a rating within the grid is left blank, it points to the "implicit half" (or contrast part) of the construct. The examiners must assume that the words used to describe the construct pairs mean much the same thing to the person involved as to themselves.

Basically the construing of persons (elements) is simply the creation and application of a system of grids, where the intersections between events may be plotted. The Rep Test Grid is composed of intersections between certain personal constructs of the person who takes the test and certain representative persons whom he knows. The basic output is a geometric or a mathematical structure of the person's psychological space. Once a grid was prepared, matching scores (measure of relationship between constructs) could be calculated. The higher the matching score between constructs, the greater the similarity exhibited in the use of them. Originally, Kelly used a nonparametric factor analytic approach (several newer techniques have been used since) to examine the results of the completed grid. Following construction and analysis of the grid, an interview session was held. This session would help expand on and verify the relationships between constructs pointed out by the grid analysis. *Laddering* is one method used to help connect the constructs in their superordinate and subordinate relationships. An important fact is that any construct can be an element in the range of convenience of a more superordinate construct.

Over the years the Grid technique has grown and become easier to use. Constructs as subject records on a grid are not *elicited* from some pre-existing repertoire, but are *created* in response to experimental demands. Two of the most frequently used methods of eliciting constructs are the use of triads and dyads. In using triads, a person is asked to consider three elements and to say which two are alike in some way and

different from the other. The person is then asked how the two (emergent pole) would be characterized and how the other (implicit pole) would be described. Elicitation handled in this manner identifies constructs in the minimal context. The elements used may be picked randomly or in some non-random pattern. The method elicitation by use of dyads are used when the elements are complex and the person finds considering three at a time difficult. In this case the person is asked to consider two elements and say whether they are the same or different and what makes them the same or different.

Since Kelly's original development of PCT, grid methodology and grid applications have been significantly extended. Grids are like people; they come in many shapes and sizes and are handled in many different ways. Different types of grids yield various scores other than "relationships between constructs." Every "grid" is an experiment in itself. The outcome varies depending on the technique used, the method of elicitation, and the type of analysis performed. Not all grids give the same results. Different grids may prove to be answering a wide range of questions. Some have extended the original binary rating method ("X" or blank) to include *rating scales*. This allows a degree of applicability to be given for each construct pole. Thus, a grid can be seen as a matrix of truth values. This is not to say that elements can be placed along a continuim represented by constructs. Ratings are a way of incorporating fuzzy logic concepts.

Over the years since PCT and the grid technique were introduced, several variations of the grid have been introduced. Kelly originally introduced the "repertory test" and used a role title list. He later presented the "dependency grid" (also known as the situational resources repertory test) which looked at the relationships between situations and people. The "rank order grid" was introduced by Salmon. For each construct, an element is chosen which

best represents the construct; as elements are chosen, they are removed from the "available" list. The same construct is used until all elements have been ordered. All other constructs are looked at in the same manner. In this way a matrix of rankings is built. A matrix with each element given a ranking is transformed into a matrix in which the elements are put in rank order. The "rating grid" assigns a rating based on a scale defined by the two construct poles. Hinkle proposed several variations of the grid. The "implication grid" or the Impgrid address only the implicit element of "self." The goal is to see what meaning each construct had for the individual in terms of other constructs. This may be done by analyzing the change in one's position on one construct and determining what others will change. In affect he tried to provide a schematic representation in matrix form of the superordinate and subordinate implications that interrelate a set of constructs. The "resistance-to- change grid" was originally used to verify that superordinate constructs were more resistant to change than subordinate constructs. Another Hinkle grid, the "bi-polar implications grid" identified and analyzed the different relationships between construct poles(parallel, orthogonal, reciprocal, or ambiguous) [FRANSELLA77]. The grid is only a technique and is limited only by the user's imagination. It can be adapted to many forms for the desired needs at hand.

The Rep Grid and the ImpGrid differ in their approaches. The Rep Grid approaches the construct hierarchy indirectly; determining the structure through analysis. On the other hand the Impgrid asks subjects directly to indicate certain links which they think exist between constructs. Kelly's "rep grid' approach allows for a degree of probability of association between constructs, while Hinkle's ImpGrid approach allows only for all-or-none decisions about construct links.

REPERTORY GRID

CONSTRUCTS			ELEMENTS						
	1	5	1	2	3	4	5	6	7
C1	WOODY	SHARP							
C2	YOUNG	MIDDLE-AGED							
C3	UNUSUAL	AVERAGE							
C4	POWERFUL	WEAK							
C5	FEELS GOOD	INDIFFERENT							
C6	SERIOUS	FUN-TIME							
C7	WELL PACKAGED	BADLY PACKAGED							
C8	SWEET	MUSKY							
C9	DAY-WEAR	NIGHT-WEAR							
C10	FEELS EXPENSIVE	FEELS CHEAP							

This particular grid is an example used for evaluating fragrances. The intersection of each element and construct would be given a rating according to the scale provided (in this case 1-5).

ELEMENTS represent people, events, things, or other tangible objects abstracted by the use of a construct. The **CONTEXT** of a construct comprises those elements which discriminate the meaning of the construct.

Figure 3.4

The usefulness of the repertory grid technique has been become more apparent and publicized since the advent of computer analysis of grid data. It is also thought that using a computer to interview patients, can improve the grid

elicitation process in several ways. A computer reduces the chances that the examiner influences the patient. The use of a computer allows the interviewing process to be more private. Many feel that they may be "wasting the time of a valuable professional" during the interpersonal interviewing process. The computer can show grid analysis results immediately. Developments in the technique have lead to several interactive computer grid programs. These programs include INGRID (by Slater) which shows elements and traits mapped onto principal axes, QARMS (by Atkin) which uses hierarchical clustering, FOCUS (by Shaw) which uses hierarchical clustering, DYAD (by Keen & Bell) which uses an incremental interviewing technique, and PEGASUS (by Shaw) which attempted to use feedback from analysis to further direct the interview [BOOSE86]. Some of these programs were not specifically developed for psychological analysis; areas of applicability range from the education process to various other interviewing domains.

Repertory grids are seductive; they promise accurate measurement of subtle perceptions, while being based on a technique which appears to be quite simple. They are also very easy to modify and adapt. However, the design is very important. If done incorrectly, inaccurate results may occur. There are several problems associated with the grid technique itself. There are questions around the soundness, reliability, and validity of definitions determined for specific constructs. The degree to which grids and constructs are predictive of other behavior are also questioned. In some cases there is the potential restriction of the subject's response gathered by grids requiring dichotomous classification or rank ordering of elements. There is some feeling that grids magnify those aspects of PCT that are least adequate and ultimately may give the wrong impression of the theory. Although construct theory stresses the importance of nonverbal constructs, grid methods are limited to verbal labels. Laddering is not guaranteed to acquire further

information. In addition, finding a good set of element samples that are significant may be difficult. Other problems may appear due to the fact that distinctions may not be publicly agreed upon.

Some advantages of the Grid technique have also been identified. Grid is applicable to developmental research. It has the ability to access multiple levels of construing, both verbal and nonverbal. Its usefulness in studying relationships between constructs (and between constructs and elements) is strong. The technique is very flexible and easy to learn. It has the ability to clarify data in an easy quantifiable form. The grid technique captures distinctions among closely related concepts. In many cases personal concepts can be gathered in the absence of a public vocabulary. The Repertory Grid technique allows the interviewer to get a mental map of how the interviewee views the world and to write this map with the minimum of observer bias. The process will help draw out and make explicit the expertise that everyone has and help define problems so that solutions become obvious. One of the most important benefits is its high degree of connection to Personal Construct Theory and its representation of the theory principles [NEIMEYER85], [STEWART81].

Given that many grid methods now exist, there are several common characteristics among the various grid techniques. They are concerned with eliciting the personal relationships, between sets of constructs, either in terms of construing elements or by directly comparing construct with construct. The central aim is to reveal the construct patterning for a person and not to relate this patterning to some established normative data. There is no fixed form or content. All forms are designed so that statistical tests of significance can be applied to the set of comparisons each individual has made. Many different applications can use the grid technique with

various analysis methods while still staying within the general guidelines of use.

PROBLEMS WITH PERSONAL CONSTRUCT THEORY

Problems of Personal Construct Theory concern several issues. Most of these issues can be grouped into two categories. The first can be identified as "Intellectual Isolationism." This problem grows from the tendency of the theory group members to disaffiliate themselves from other similar theories. This tendency has its origin in Kelly's dismissal of the formulations of his predecessors as if they were wholly irrelevant to his own theory-building efforts. Some feel that the theory could be enriched by recognizing and working with research done in related areas. The second major problem can be called the "Crisis of Methodology." There may be limitations on discovery imposed by the particular set of techniques available and endorsed. Such reliance tends to restrict both the type of questions addressed and the range of knowledge produced. An overwhelmingly large number of articles and reports published have employed some variant of the repertory grid technique as their primary or only means of analysis. Very little research has been done using other analysis methods.

PERSONAL CONSTRUCT THEORY SUMMARY

Key ideas to remember about PCT include: 1) perceptions influence expectations and expectations influence perceptions, 2) the medium through which this happens is known as the

construct system, and 3) construct systems are unique to the individual and develop throughout life. There are several points about PCT that people in almost any field of study or research may find useful. The degree of agreement between the construct systems of two people is a measure of the extent to which they are like each other, and the extent to which they are likely to understand each other without effort. The degree to which one person can understand the construct system of another is a measure of the extent to which he understands the other person. Construct systems change over time to digest new information. Construct systems have a survival value for their owners; they are the core to their being. These and other reasons make PCT useful in various industrial activities which also includes knowledge acquisition for Knowledge-Based Systems.

PCT was written as a comprehensive and practical theory of personality. Personal construct theory begins with the fundamental assumption that the person is striving to create an understanding of the world. The person then "tests" this understanding by predicting future events. If these events are consistent with prediction, the construct is said to be validated. On the other hand if the events are inconsistent with prediction, the construct is invalidated and the person experiences internal pressure to change his system. An assumption is made that the person interacts with the world in a manner similar to a scientist, hence the "Person as a Scientist" metaphor.

GRID USE

GRID APPLICATIONS

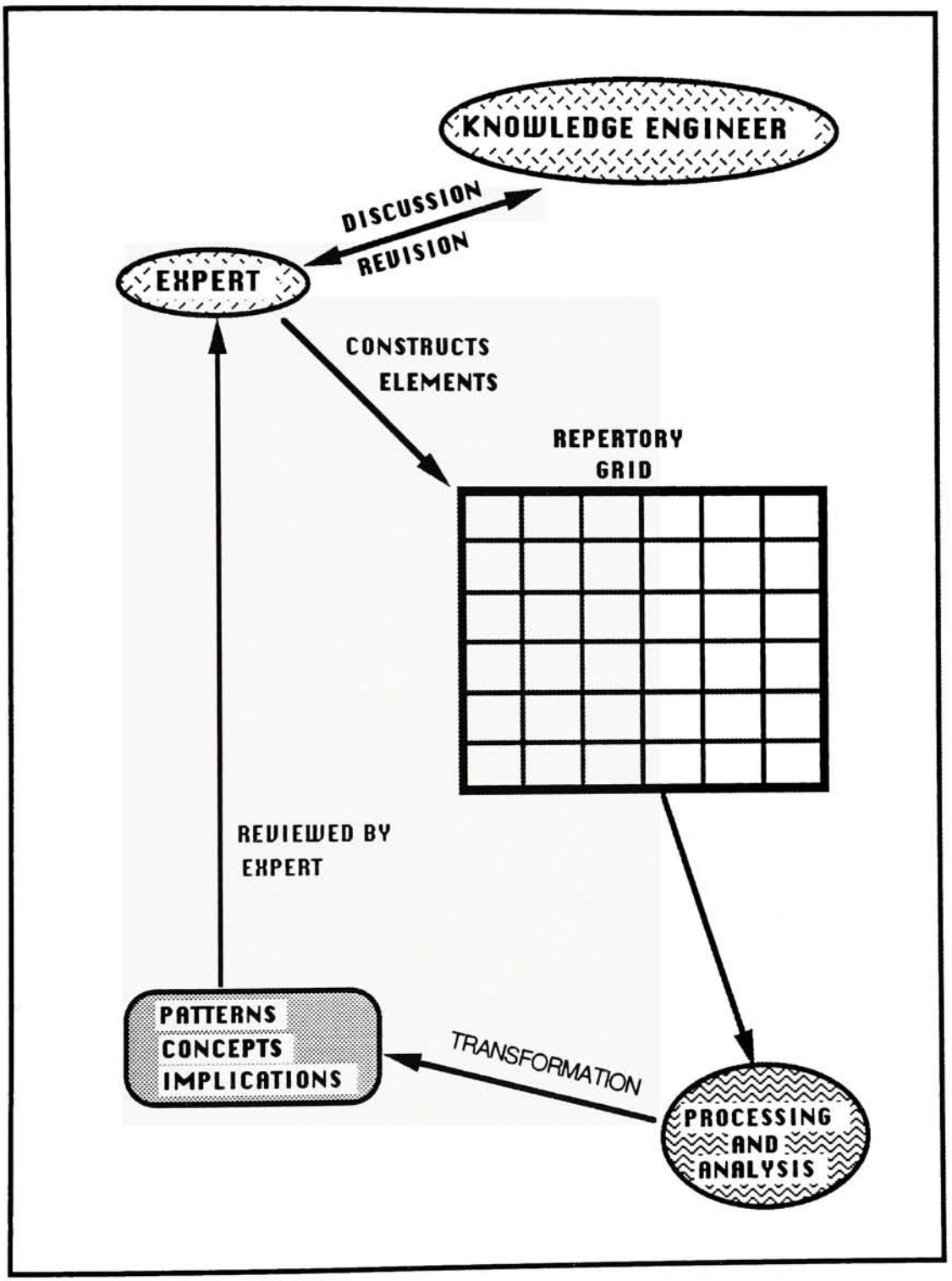
Repertory grids have been used widely in clinical psychology, educational research, and decision making studies. At its simplest, the grid methodology provides a way of doing research into problem areas - almost any problem area - in a more precise, less biased, way than any other research method. The grid is a very helpful tool for exploring subjective judgements. A byproduct of the grid technique is the determination of the domain vocabulary. Subtle uses of the grid have made it possible to evaluate training, facilitate negotiations, resolve conflicts, assemble teams, and counsel people with problems. Other application areas include designing questionnaires (to be sure that issues are addressed and that priorities are considered), creating surveys, creating questions for interviews, and impact analysis (ie. training: a grid is created before and after).

As can be seen, many applications exist in the business and education worlds as well as the psychology arena. The area of market research was the first known business application of Grid [STEWART81]. Other activities where Grid can be used include quality control, design, and attitude surveys. Any activity which requires asking questions is a potential use of the Grid technique. The Grid can be used for many purposes, but only one at a time can prevail. Interviews with an underlying purpose require that questions steer the construct elicitation. A qualified question is used to do this. The qualifying question nearly always takes the form "in terms of ..." or "from the point of view of ....". The Grid technique is a developing one which is constantly being applied to new applications or having modifications made to fit a need.

The "Grid Process"

The way that Grid records interview data is a great advantage. A Grid record is full of information. A line of thought can almost always be triggered by looking at the record. Also, several grid interviewers can work on a problem and everyone can understand other interview records without much explanation. The concepts used for comparison are *elements*. The bipolar distinctions between elements are called *constructs*. The process of getting constructs from elements is called *construct elicitation*. Elicitation can be done on an individual or on groups at a time. Constructs relating to the same elements will vary from person to person. For example, one person may have the construct "lot of experience -- very new," given the elements A, B, and C. On the other hand, another person may not identify with this same construct.

A construct is not necessarily composed of a phrase and its semantic opposite. Construct elicitation can be used to put into words, perceptions that have never been verbalized. Care should be taken so that verbal labels applied to constructs are communicable. The elicitor must be careful not to contribute parts of his own construct system nor to distort in any way the constructs which are offered by the subject. The examiner should listen and not impose constructs. Construct poles should be specified by the subject in order to get their view of the contrast poles. The elicitation process may identify new constructs and old ones may be re-defined. It is important to identify pre-existing constructs as much as possible instead of building new ones. A person regards his own constructs more highly than those selected from a pool of constructs. Constructs supplied by an interviewer do not necessarily represent poles of the same construct in all cases. Thus, elicited constructs are usually more meaningful.



[HART86]

Figure 3.5

Constructs can be generated in several ways. They may be supplied, elicited through triads or dyads, determined through laddering techniques as well as several other psychological based methods. Supplied constructs may not be meaningful or may introduce some level of distortion. Triads may be defined through various means including minimum context card forms, fill context forms, sequential forms, self-identification forms, personal role forms, the full context form with the personal role feature, or some other random draw method. It is equally reasonable to use two elements or more than three elements for elicitation [FRANSELLA77]. In some cases it may be easier to work with two elements at a time rather than three. The elicitation and laddering of constructs may be considered an art rather than a science. There are varying beliefs about the number of constructs that need to be elicited. In many cases a large number of triads and dyads may be available in order to generate constructs.

The selection of triads and dyads affects the construct elicitation process more than one might think. The elements used and the order presented may affect what constructs are acquired. Triads or dyads may be chosen on a truly random basis, chosen to bring out the greatest contrast, or they may be chosen based upon some pattern. Regardless of what method is used, the elements in triads should be changed frequently to avoid the tendency to dwell on a point. There is no correct way of generating triads or dyads. The process is part of the "art" of using the grid technique of elicitation.

The construct system is a hierarchy. Constructs have relationships with other constructs which should also be defined. These include parallel, orthogonal, reciprocal, and various ambiguous relationships (see figures 3.6a-3.6d). A good Grid interviewer needs to know what to do in the interview to expose as much of this construct system as is necessary for the purpose of the interview. Laddering is one

way to solicit constructs from other constructs, but it is not guaranteed to elicit superordinates of subordinates as desired. A *laddering* affect can be achieved by asking successive "WHY" questions, after determining a construct, until no further explanation can be given. Along the way new constructs can arise. The "HOW" question is used to come down the ladder and thus break a construct into its component constructs (How are they different..?). There are various laddering techniques. A laddering technique is used to get some depth and perspective and to get from the general to the specific and back again. It may become dangerous when *core constructs* are reached. These core constructs are the fundamental constructs that one uses to interpret the world. They are very personal for most people.

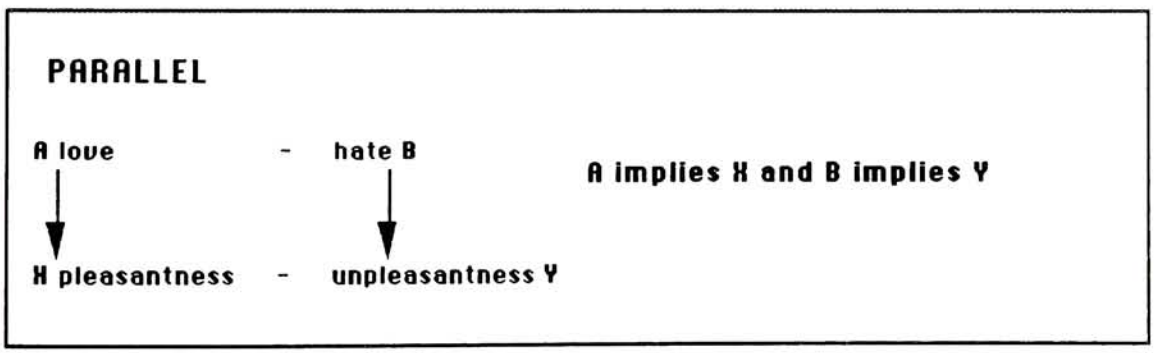


Figure 3.6a

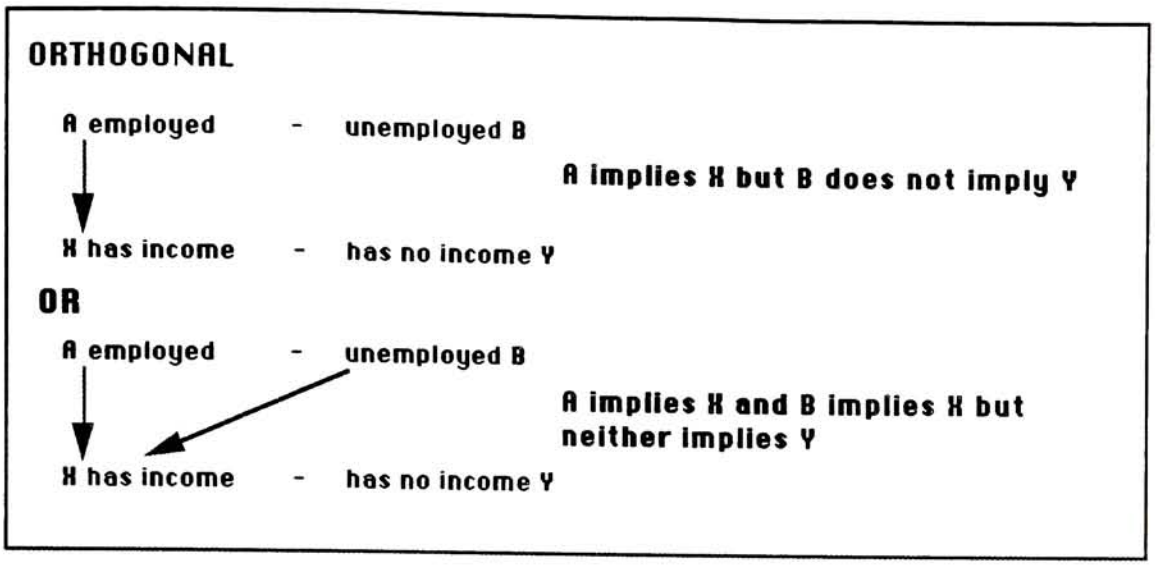


Figure 3.6b

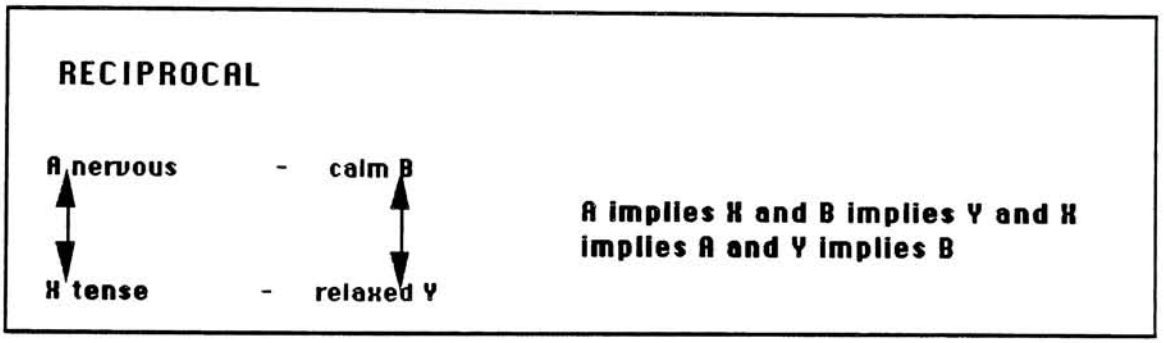


Figure 3.6c

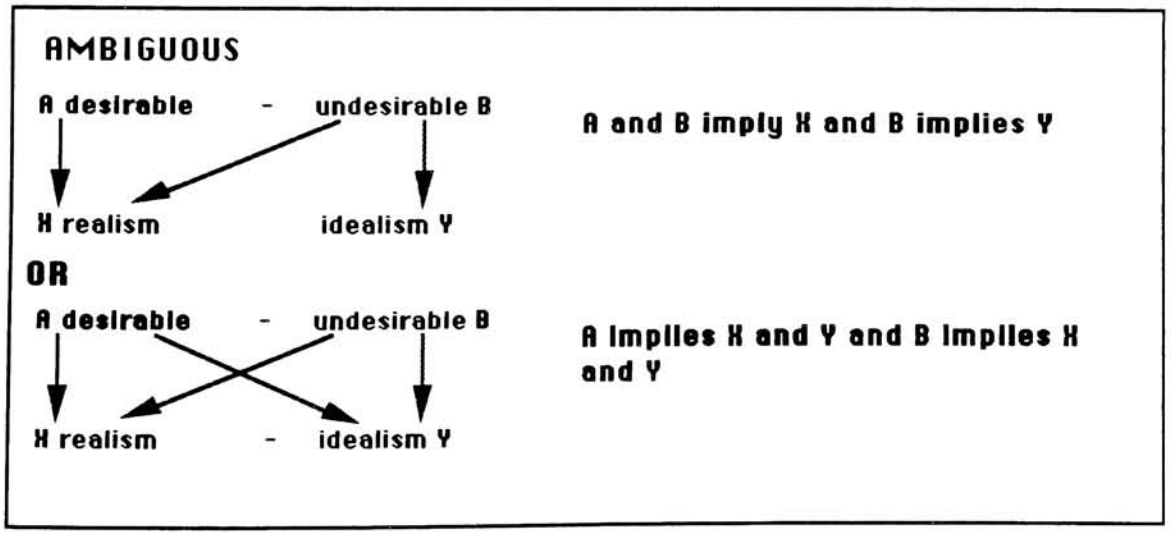


Figure 3.6d

It is important to avoid having some elements dominate the grid by using them much more than others to build triads and dyads. Eliciting constructs is time consuming and if not handled well can become tedious or unnerving. Constructs should be avoided which are based on role titles, exist as vague or superficial concepts, or depend upon situational circumstances. It is suggested that the number of constructs elicited be as few as possible (maybe 8) - especially if additional data will be elicited at another date [EASTERBY-SMITH81]. On the other hand it is often difficult to verify that a sufficient set of constructs have been elicited.

ELEMENTS

Since a grid is derived from the elements, appropriate selection is critical. The elements in any form of grid must be relevant to the constructs used. It is important that the elements chosen be specific as possible since they will define the focus of the grid. Elements should be homogeneous; drawn from the same category or type and level of complexity. They should also provide representative coverage of the domain. The number of elements used (for computerization) should range from six to twelve (many clinical applications have used 15-25) [EASTERBY-SMITH81]. The elements used help define the kind of conversation that develops. When choosing elements, the following points should be remembered: 1) The more specific and precise, the better, 2) A rough scatter over the element area is acceptable, and 3) if interest lies in the border between one kind of element and another, then some elements from the other side of the border must be included.

Elements may be generated in several ways. They may be supplied by the interviewer, listed by the interviewee, elicited through discussions, or given based upon general

descriptions provided by the interviewer. Many different methods exist to generate elements based upon the approach taken. There are reasons for using and not using each approach. Elements provided by the interviewer may not be familiar and thus the interviewee may feel less participation. A list generated by the interviewee once told what class of elements are needed can be biased toward only those elements that are familiar, liked, or have a good reputation. In order to solicit elements through a natural discussion, a list of questions to generate the elements be prepared before hand. Extra preparation may be necessary if it will be possible to receive answers multiple times. In this case more time is required, but more information can be gained. When using this approach questions should come in pairs, to hopefully produce contrasting elements.

Any element used should follow a few simple guidelines. Elements should be discrete. For example, elements are most often people, objects, events, ideas, or activities (in other words, nouns and verbs). Selected elements must be homogeneous; that is classes of elements should not be mixed. Elements should not be subsets of other elements. An element that is a subset of another will contain many similarities and it will be difficult to compare and contrast. Finally, elements should not be evaluative. It should be noted that elements are more easily shared than constructs. As a result, increasing the number of elements elicited may reduce the amount of structure that a subject may impose on the given grid.

RATINGS

The "full grid" procedure allows one to exhibit not only the constructs themselves, but in detail how they are used. Instead of regarding each construct as a pair of words, a scale is used to add meaning - most often a five-point scale (scales from two to nine have been used). This scale is often thought of as applying fuzzy logic principles or degrees of membership. The ratings are generally viewed as assignments of truth-values to logical predicates. A grid then may be seen as a matrix of truth values. In many cases linguistic variables are assigned to the ratings of the scale. An alternative to rating is ranking. The ranking method assigns the elements in a rank-order from the emergent (left-hand) pole to the implicit (right-hand) pole. In using the rank-order method, each element will have a unique value assignment, whereas the the rating approach may give the same value to multiple elements.

Each element for each construct is rated so that a "matrix of elements by constructs" is built. Each rating identifies how applicable a construct pole is to the element at hand. In a "full grid" interview, one is testing the *range of convenience* of each construct. For constructs that can not be rated, values of N/A or 0 may be given. A construct is irrelevant to an element if the element is assigned to neither of the poles (ie. the degree of membership is 0). The "across method" (rate each element of the construct before moving on to the next construct) or the "down method" (elicit all constructs before rating the elements) may be used, though each will have a slightly different flavor.

"GRID" ANALYSIS METHODS

OVERVIEW and BACKGROUND

Once the purpose of using a grid is met, the elicitation process can cease. This can happen at almost any point. At this stage some kind of analysis usually takes place. However, one should be careful not to become too far removed from the raw data when doing analysis. What analysis to use is partly a matter of personal preference and partly a matter of purpose. Analysis of the grid is dependent upon general methods of analyzing statistical data; the computation of the "similarity matrices" or "correlation matrices." The interpretation of grid data is considered by some to be an art and not a technology. It is important to obey the rules of statistics when interpreting grid results since the relationships are relatively stable. Most analysis on grids is done to establish meaning with others. Therefore the analysis is more comparative and creates correlations and other such measures. For analysis purposes, two grids with the same elements, but different constructs can be combined into one.

Traditional approaches of grid analysis have used a non-metric method of factor analysis, other methods of factor analysis both metric and non-metric, principal component analysis, and multidimensional scaling. Non-metric methods assume monotonicity between the final solution and the matrix being analyzed. The principal ways known to analyze grids include frequency counts, content analysis, visual focusing, cluster analysis, and principal-components analysis [STEWART81]. However, analysis may also address measures of construct relationships, intensity, cognitive complexity, functionally independent construction scores, saturation scores, extremity ratings, ordination, superordinacy, inter-construct

correlations, conflict assessment, and so on. The lower the intensity score, the more disordered(loose) is one's thinking. The more agreement between elements of each pair of rows, the higher the score and the lower the degree of cognitive complexity [FRANSELLA77].

Two major approaches have been made concerning grid analysis. One has been cluster analysis and the other is a singular-value decomposition or "principal components" approach. Many grid analysis techniques are based on distance measures between vectors based on either rows or columns of the grid. Correlation matrices help create difference scores or distance measures. The correlation matrices of similarities between elements and between constructs which are then used to form clusters are usually calculated using either similarity or distance measures. Grids using rankings are useful for multidimensional scaling analysis which generates conceptual maps or cluster analysis. The "focus technique" rearranges rows and columns so that similar constructs are positioned close together.

Analysis techniques are handled in several ways. Dendrograms or clustering diagrams graphically explain the relationships and groupings. Elements and constructs may be graphically compared for similarities and differences. The area of concentration may lie with either or both. Pattern analysis is based on evaluating the variation of an element from the mean point in terms of a multiple of the standard variation (or the goal alternative). Linguistic labels which label the nearest point (of the scale) to the unknown value may be used to represent a difference measure [ESHRAGH81].

Computers can be used to assist in collecting and analyzing repertory grid data in two ways: 1) analyze the matrix of numbers, 2) facilitate the elicitation process. The use of computers in the research of Construct theory and the use of

grids adds a new dimension to the work. Grids collected for computer analysis may contain much material redundant for the purpose at hand. If it is a big grid study, then a "prototype" interview and analysis will be very useful. There should be some sensitivity to the "Not Applicable" rating. Some computer programs turn a "Not Applicable" into a mid-point scale judgement and in so doing, do some damage to the structure of the grid in the process. This can change the meaning of the analysis and the construct system dramatically. Most of the interactive computer systems use a simple interview technique in which the subject simply lists elements and produces constructs and the results are analyzed [BOOSE86].

In using *frequency counts*, a count is made of the number of times particular elements or particular constructs are mentioned. This technique is most often used when several people are interviewed and a common thread is being sought. Frequency counts work best when the elements or constructs being counted are discrete, well defined, and have consistent public meaning. *Content analysis* is similar to frequency counts in that elements and/or constructs are assigned to series of categories and then a frequency count is performed for each category. The first task is to develop a category system.

A series of comparisons are needed for *visual focusing*. Rather than using the scale technique, a tick/cross system is used. The pattern of ticks and crosses for each element is compared against each other element to determine the number of matches between any two elements. The maximum number is the number of constructs in the grid and the minimum is zero. The next step is to draw a matrix showing the agreement scores between every possible pair of elements. This symmetric matrix is then inspected for 'high' numbers. Where there is a high agreement score between two elements, it can be assumed

that those two elements share the same meaning. The differences between these two highly similar elements should be gathered and new constructs added to the Grid as a result. The Grid is next re-sorted so that similar elements are placed close together. Exactly the same procedure can be done for constructs. In using constructs, the minimum acceptable score is four and the highest is eight, given that nine elements are used. If a score is found to below four, the construct is reversed (ie. the emergent pole becomes the implicit pole and vice-versa).

CLUSTER ANALYSIS

Cluster analysis is one of the more recent techniques used to analyze grids. There are several forms of cluster analysis, but most of them start from a matrix of similarities or distances between elements of the data. Cluster analysis is similar to visual focusing. Using cluster analysis yields a simple view of the structure created. Cluster analysis takes many forms. Some of the techniques include various hierarchical methods, partitioning techniques, density methods, or clumping techniques. The main idea is to move constructs together that are similar.

The idea of re-sorting the Grid so that like elements are placed together and like constructs are together is carried further by the use of various computer programs. The technique may best be exemplified by the FOCUS program, developed by Mildred Shaw and Laurie Thomas at Brunel University. The FOCUS program, instead of using simple differences, uses correlations. This enables it to be much more sensitive and to cope with many various scales. FOCUS, first, examines the Grid looking at the correlation between elements. It searches for two that are most highly

correlated. The program then melds them into a 'new' element. This process continues for all similar elements until they all combine into one final 'new' element and a 'tree' structure is formed. PEGASUS and SOCIOGRID, by the same authors, use the same principles. In PEGASUS the Grid is built up interactively with the computer; the interviewer is removed from the procedure. The computer begins by asking for the purpose of the Grid and asks for six elements. It then asks for constructs and assigns ratings to each element applicable to that construct. The Grid is built up by looking at degrees of correlation. SOCIOGRID allows Grids of two or more people to be mapped onto the same space. Q-Analysis used in the QARMS program uses a form of hierarchical cluster analysis based on a distance measure. In Q-analysis, the reversal of poles becomes an important operation. George Kelly used such a process in his analysis, however he called it "reflection" [GAINES81].

FOCUS uses a distance based clustering technique which sorts constructs into a linear order such that constructs close together in the space are also closest together in order. The cluster analysis method used within FOCUS is based on the "city block metric." The distances between elements or constructs calculated from the "city block metric" are functions of the number of constructs or elements respectively in the grid, together with the rating scale used. These are therefore scaled to give "percentage matching scores." The focus algorithm is similar to the "single linkage" or "nearest neighbor" hierarchical method. The matrices of element and construct matching scores are produced from the "city block metric." The major criterion for forming clusters is that linear re-orderings of the constructs and elements respectively will result in the final grid displaying a minimum total difference between all adjacent pairs of rows and columns. This leaves the patterning in blocks of like

responses, often but not necessarily diagonally across the grid [SHAW80].

DISTANCE-BASED ANALYSIS

Analysis of grids based on distance measures takes the view that constructs can be thought of as vectors in space. Each construct can be represented as a point in a multi-dimensional space whose dimension is the number of elements involved. It is equally possible to consider vectors of elements in space (a knowledge-based system, however, will concentrate on comparing constructs). Numbers in the grid are thus referred to as *vector values*. The distance between vectors describe the difference in which the elements are rated for that construct. Constructs with a distance of zero seem to be equivalent. Those constructs of distance zero imply that all elements are construed the same way and that the constructs are used the same way and therefore they are equivalent. For measuring the distances between elements in a ranked grid, the standard technique is to calculate Euclidean distances. For measuring correlations between constructs in a ranked grid, Pearson's product-moment correlation coefficient "r" is appropriate; where "r" is calculated ignoring missing entries [LEACH81]. To some degree the initial ordering of elements and constructs may cause slight differences in the results using distance-based analysis. This is due to weighting that occurs.

Various distance-based analyses of grids have two common factors which can restrict their application in some contexts. First, the structure exhibited is limited in its semantics to a symmetric relation of "neighborness" between the items clustered. Secondly, the analysis produces results about distances, components, connections, geometrical relationships,

and so on which represent a different way of looking at the data. Distance-based grid analysis relies on all constructs having a rating for all of the elements. A possible alternative is to have a value for "true" for both poles and another for "false" for both poles in addition to the strictly true/false or rating scale. Principal component analysis is the most widely known distance-based analysis technique.

Principal component analysis, as a distance-based technique, is a process of iteration of projecting constructs onto different axes which account for different degrees of distance. Analysis done using the "principal components" technique requires no assumptions about the data being analyzed. It is essentially an analysis of the total variance of the data and can be done by row and column. Under the "principal components" approach, the intra-class correlation among elements can be easily computed as each construct is elicited and may be said to provide a measure of cognitive complexity at that stage of elicitation. This correlation can be used to determine when to quit eliciting constructs [BELL81]. Principal component analysis is related to factor analysis of semantic space used in the study of semantic differentials. In this case the entire set of vectors is analyzed to determine a set of axes such that the projection of each construct onto the first axis accounts for most of the distance between them, the projection on the second axis accounts for most of the remaining distance, and so on [SHAW80].

Principal component analysis asks the question: "How many independent dimensions are needed to describe all relationships within the current matrix?" Each dimension represents a variable. This approach makes it easy to perform a change in the displayed Grid. The great appeal to this method lies in its simple visual presentation of data. Not all of the variance is represented properly by the limiting

two axes of paper. The INGRID system, written by Pat Slater, Jane Chetwynd, and others is the best known implementation of this technique.

LOGICAL ANALYSIS

An alternative way of looking at grids considers an assignment of truth-values or logical predicates. Logical analysis views constructs as predicates applied to elements. Logical predicates are derived from implications or entailments found in the grid while forming a hierarchy from the listed constructs. In cases where rating grids are used, fuzzy predicates are used. Two constructs are equivalent in this light if for all elements, the left hand pole (LHP) rating for an element for one construct is the same as the LHP rating for that same element and a different construct. Because of the inversion relationship that exists for constructs, one construct pole is the logical negation of the other pole. Fuzzy logic has been applied to constructs as a way of representing their applicability to a particular element. The use of entailment for elements is a way of determining similarity or a preference order. This analysis technique is best represented by the ENTAIL program.

ENTAIL (Entailment Nets Through Analyzing Implicational Links) derives asymmetric implications between construct poles so that one can infer how a new element might be rated on one construct given how it it is rated on others. ENTAIL takes the expert's distinctions to be fuzzy predicates. The results of an ENTAIL analysis are essentially degrees of membership to equivalences and entailments or preferences. ENTAIL originally derived all entailments possible that were not disconfirmed by the grid data. Therefore, the significance of some entailments was higher than others. In many cases, users of the system were surprised at the relationships revealed.

Figures 3.7a-3.7c take a simple decision-action table approach to show how a repertory grid can be used to generate pseudo-rules.

DECISION-ACTION TABLE

KNOWLEDGE	CASES					
DECISION CRITERIA	1	2	3	4	5	6
INSIDE ACTIVITY	YES	YES	NO	NO	NO	NO
CONTACT ACTIVITY	NO	YES	YES	NO	NO	NO
TEAM ACTIVITY	NO	YES	YES	YES	NO	NO
SEASONAL ACTIVITY	NO	YES	YES	YES	YES	NO
ACTION						
CHOOSE **BIKING**	----	----	----	----	YES	----
CHOOSE **JOGGIING/RUNNING**	----	----	----	----	----	YES
CHOOSE **SWIMMING**	YES	----	----	----	----	----
CHOOSE **FOOTBALL**	----	----	YES	----	----	----
CHOOSE **BASEBALL**	----	----	----	YES	----	----
CHOOSE **BASKETBALL**	----	YES	----	----	----	----
EQUIPMENT NEEDED	----	YES	YES	YES	YES	----

This partial DECISION/ACTION table can be considered similiar to a repertory (rating) grid. The "action" is analagous to "elements" and the "decision criteria" is analagous to "constructs." Ratings in this case are binary truth-values. Elements or the resulting "action" can be the basis for another grid or decision/action table.

Figure 3.7a

REPERTORY GRID

CONSTRUCTS		ELEMENTS						CONSTRUCTS
EMERGENT POLE		1	2	3	4	5	6	IMPLICIT POLE
C1	INSIDE ACTIVITY	X	X					OUTSIDE ACTIVITY
C2	CONTACT ACTIVITY		X	X				NON-CONTACT ACTIVITY
C3	TEAM ACTIVITY		X	X	X			INDIVIDUAL ACTIVITY
C4	SEASONAL ACTIVITY		X	X	X	X		YEAR-ROUND ACTIVITY
		SWIMMING	BASKETBALL	FOOTBALL	BASEBALL	BIKING	JOG-RUN	

Figure 3.7b

RULE #1: IF inside activity
AND no contact activity
AND no team activity
AND no seasonal activity
THEN
activity is swimming

RULE #2: IF inside activity
AND contact activity
AND team activity
AND seasonal activity
THEN
activity is basketball
AND equipment is needed

RULE #3: IF no inside activity
AND contact activity
AND team activity
AND seasonal activity
THEN
activity is football
AND equipment is needed

RULE #4: IF no inside activity
AND no contact activity
AND team activity
AND seasonal activity
THEN
activity is baseball
AND equipment is needed

RULE #5: IF no inside activity
AND no contact activity
AND no team activity
AND seasonal activity
THEN
activity is biking
AND equipment is needed

RULE #6: IF no inside activity
AND no contact activity
AND no team activity
AND no seasonal activity
THEN
activity is jogging/running

Figure 3.7c

Thus, relations induced involve trade-offs between fuzzy truth values and the probability of their being correct. Despite its different approach, ENTAIL has many of the same features incorporated into INGRID, QARMS, and FOCUS [SHAW80].

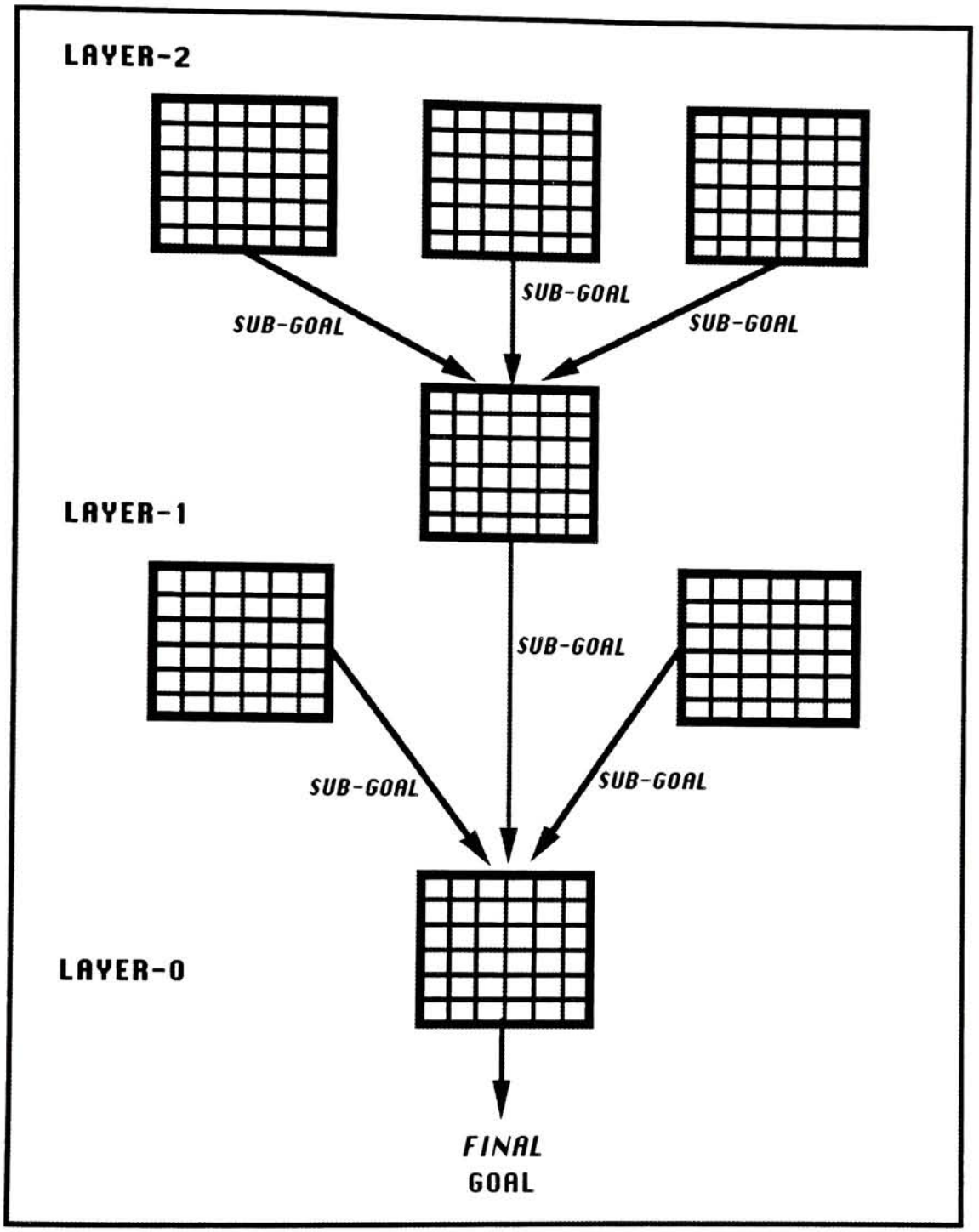
GRID SUMMARY

Several tips should be remembered when working with Grids. First, decide on a purpose. An evaluation of any kind has three components: a purpose, a judgement, and information on which the judgement is made. Before beginning an evaluation, one must know the kind of information needed and what factors will be used to evaluate this information and make a judgement. Other things to consider include: "Whose perspective to obtain?", "Who are the experts in the situation?", "Who will use the results of the Grid and how will they be used?". This amounts to having a plan on how the grid and its results will be handled. Avoid interpretation of constructs, they may not be true. Also, the conversation may be just as valuable as the grid, so stay tuned.

An evaluation of the technique can be done by applying the grid to oneself and then analyzing the results. In this way one can become familiar with the method of the grid and also evaluate the grid since it can be validated rather easily. As a simple exercise in the use of the "grid technique" (which also can be used to determine how much a person knows about a subject), do the following. Determine at least nine examples (elements) within the class of a topic (books, movies, etc.). For each unique set of three, list as many ways in which two are the same and a third is different. Repeat this process until no more differences can be identified. As a further action, determine if differentiations can be grouped (so that some areas predominate). Once these activities are done, the

basic grid has been defined. The next steps consider rating and analysis techniques. This exercise is often suggested to experts before they use any computer-based tool using grid methodology.

It should not take much thought to recognize the possibilities of using the "grid technique" as it applies to knowledge acquisition. Distinctions captured in grids can be converted to other representations such as production rules, fuzzy sets, networks, or frames. Large problems can be broken into pieces and represented in multiple grids and arranged in hierarchies and lattices. Hierarchies are organized around cases, knowledge sources (ie. experts), solutions, and traits. The elements of a grid represent final conclusions or consequents which may be used as input to other grids. In the case where information is in the form of multi-valued logic rather than truth-value logic, grid ratings can represent non-ordinal traits (eg. color). PCT and the "grid technique" thus appear to be an alternative to both knowledge elicitation and knowledge modelling.



A GRID, through its derived implications, eventually will reach a "**GOAL.**" A single Grid naturally has intermediate conclusions. However, using this technique, several layers can be used to reach ***SUB-GOALS***. The *SUB-GOALS* of one layer become the **ELEMENTS** of the next layer closer toward the final goal.

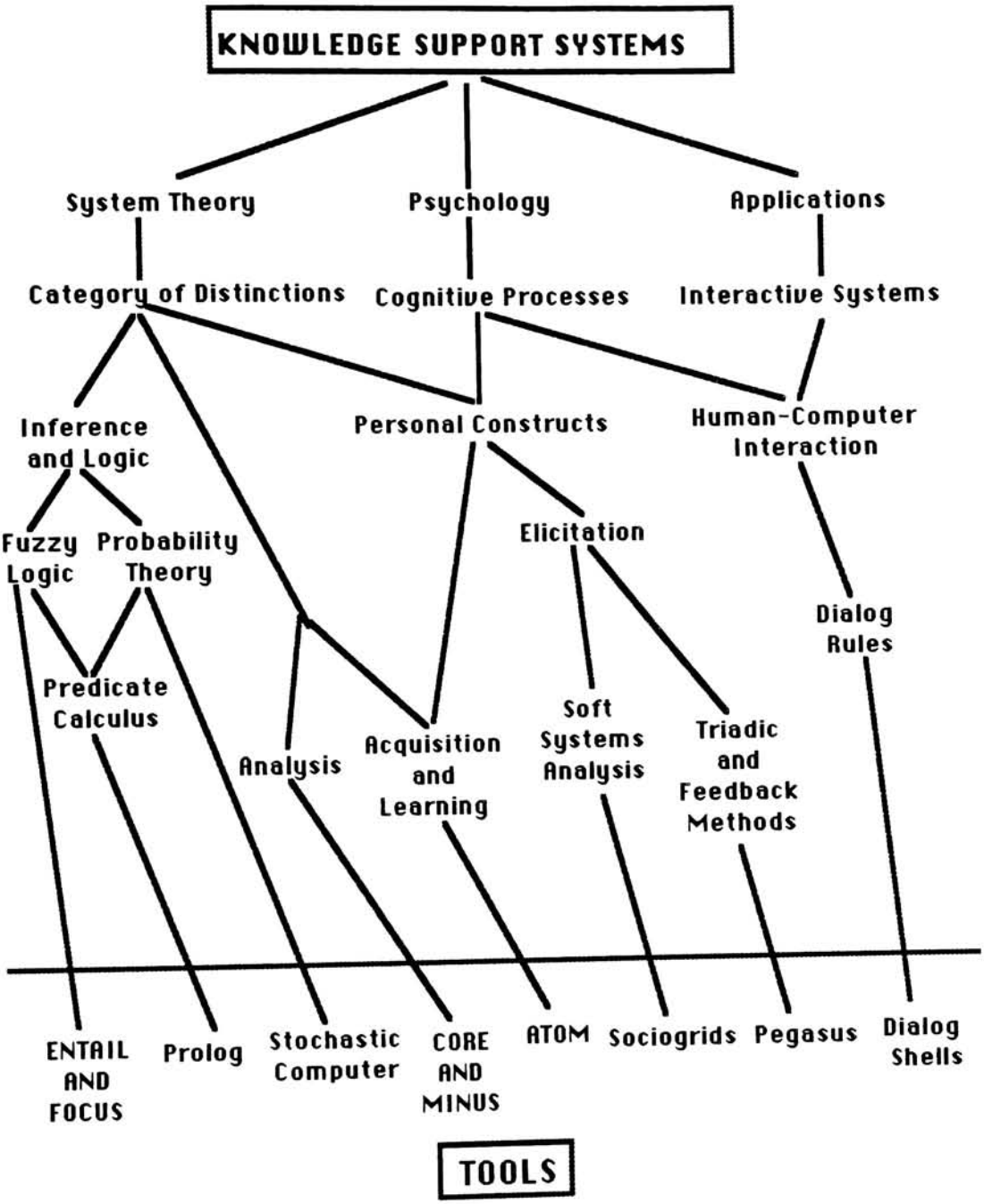
Figure 3.8

CHAPTER 4 SURVEY OF KNOWLEDGE ACQUISITION TOOLS

INTRODUCTION

Knowledge acquisition tools must be developed that are easy to learn and use. They should have characteristics which include use natural forms of expression for the application, lack machine-dependent details in the knowledge specification, are capable of integration with other languages, editors, and facilities that are used on the same system, and are easy to learn and use by both programmers, systems analysts, and domain users and experts [SOWA84]. A major factor is integration. Any tool developed should be able to integrate with other systems or tools with relative ease. Knowledge acquisition tools all have some form of dialog with the expert user, but the technique used to gather and store data varies greatly. They may use a form of strictly automated interview, a questionnaire, repertory grids (or rating grids), influence diagrams, or other derivations or combinations of these techniques. Some tools are more graphical than others and some are more complex and closely tied to knowledge-based system development than others, but they all are an attempt to solve the same problems. In addition, some tools may be more domain independent than others, but the basic theory used is the same.

Figure 4.1 tries to link many of the current methods with some of the research tools.



This diagram (modified from [BOOSE86]) shows some knowledge support systems and some tools which implement them. ENTAIL, FOCUS, CORE, MINUS, Sociogrids, and Pegasus are parts of the PLANET system (SHAW).

Figure 4.1

A growing number of tools are incorporating repertory grid methods from PCT (ETS/Aquinas, PLANET, KRITON, NEXTRA/KSS0, FMS Aid, and KITTEN). Grid-based tools help in rapid-prototyping and knowledge base analysis, refinement, testing, and delivery. These tools interview experts directly and help them refine, expand, analyze, and test their problem-solving knowledge. Grid-centered tools work best on analysis problems (those whose solutions can be comfortably enumerated such as classification and diagnosis). Advantages of using grid-based tools include: rapid prototyping possible, feasibility study facilitation, building a vocabulary becomes easier, solution and trait elicitation is enhanced, interactive testing and refinement is feasible, implication discovery, conflict point identification, generation of expert enthusiasm, ease of non-KE use, training of KBS concepts to users, data gathering, analytic capabilities can uncover inconsistencies, and maintenance can be controlled to maintain consistency [BOOSE88]. Knowledge acquisition tools using repertory grids incorporate techniques for eliciting the expert's vocabulary and also to structure some aspects of the domain knowledge. These tools, as well as others, assume that the user believes the task at hand can be solved using a form of classification (some feel that almost "any" problem can be modeled as a classification problem).

TOOLS RELATED TO PCT

ETS

The Expertise Transfer System (ETS) is a computer program that interviews experts and helps build Expert Systems. The interviewing methodology is borrowed from a branch of psychology called Personal Construct Theory. Psychologists use personal construct methods to help patients explore their view of the world; ETS uses these methods to help experts explore the way they solve problems. The goal of the system was to significantly shorten the knowledge acquisition process and also improve the quality of elicited problem-solving knowledge. ETS incorporates this methodology for knowledge elicitation, testing, combination, and expert system delivery. Benefits of ETS include: 1) its ability to shorten the KE process (by gathering traits, the domain vocabulary, and the conclusion set), 2) its prototyping features, 3) the improved quality of elicited information, and 4) the ability to gain early expert acceptance. The major limitation is that it is nearly impossible for the program itself to generate a robust model of the application domain (This situation is not unique to ETS.). An imperfect model, however, is better than none.

ETS is written in Interlisp-D and implemented on the Xerox 1100 family of Lisp machines. It is also implemented in Interlisp-10 on a DEC VAX and in Unix/C on an IBM PC. Other versions exist which target representations for KS300/EMYCIN, S.1, M.1, TI Personal Consultant, OPS5, MRS (a Stanford research tool), Prolog, KEE, and SUDV (a Boeing C-based tool). ETS is applicable to structured selection (analytic) types of expert systems, particularly classification problems. The system has been used to build consultation

systems that combine expertise from multiple experts. Often experts begin using ETS without a clear problem structure in mind. Since its implementation, several hundred prototype systems have been built.

Ideally, knowledge is elicited from information sources and placed into an information base. From there, it may be analyzed and organized into knowledge bases, which may be directly used with expert system building tools. These tools are initially used to test the knowledge and "test case histories" are built as the knowledge base is incrementally refined. Individual knowledge bases can then be combined into knowledge networks, where more testing occurs. Finally, delivery systems are tailored for efficiency and on-the-job case records are kept to improve the system [BOOSE86]. ETS makes a shallow path through this ideal framework. Experts are interviewed. Problem-solving knowledge is captured and stored in rating grids. Analytic methods are used to help the expert refine the problem-solving ability of the knowledge in the rating grid. The rating grid is then transformed into a set of production rules with strengths of belief to handle uncertainty situations. The expert builds cases by testing rules. Production rules internal to ETS or in a variety of KBS tools (EMYCIN, S.1,KEE) are used for test consultations.

ETS and the domain-expert user initially build a grid using a list of elements supplied by the expert. However, incremental interviewing can also be used to start the grid. In this case, three elements are gathered to start the process. Once several traits have been elicited, ratings are given and the process is re-iterated to build up the grid. The traits that the expert provides are dependent upon the order in which the system combines the elements into triads. The goal is to find a set of traits sufficient to solve the problem at hand, not to find them all. In other words, it is

desirable to find a minimal, complete set of traits that is also consistent in its reasoning. ETS elicits traits that may converge on such a set. If necessary, the grid is revised to eliminate overlapping, equivalent, or unimportant information. Furthermore, ETS can combine the rating grids of several experts into a single consultation system. The expert user is asked to rate each element against each pair of traits and thus transform the grid into a rating grid. A rating scale of 1-5 is used with additional values of "B"(represents both traits) and "N" (represents neither trait) also available.

Implications among traits are found based on information provided in the grid. Generalizations produced may or may not be correct. The process used to derive implications is a form of induction. Reciprocal relationships found are removed by combining traits. These implications are categorized as similar, entailment, or ambiguous. Equivalent traits may be combined at some later time, if the equivalence prevails. Implications may also define hierarchical relationships. Laddering will help define the hierarchy. Laddering is also used to uncover inconsistencies and to breakdown ambiguous relationships into their component parts. Implications found and presented should correspond with paths in the expert's construct hierarchy even though it may be surprising to him.

After constructing an implication tree and performing the implication analysis, rules are generated. These rules are either "conclusion rules" or "intermediate rules." A certainty factor is associated with each rule. Conclusion rules, named by the expert, are created from ratings in the grid. Rule names will be used as labels for variables when rules are generated. Intermediate rules are based on the relationships in the implication trees. Each implication is responsible for generating a rule. These rules contribute

pieces of evidence at a higher level than that of the conclusion. Two rules may be generated for each rating - a positive one and a negative one. Rules will not be created for ratings of '3' (the mid-point on the rating scale).

A refinement process begins once the initial set of rules are generated from the grid. Once the rules have been generated, the expert can use ETS's limited consultation facility or another KBS shell to test the knowledge. If the expert disagrees with particular rules, a rating examination and laddering process may be used to identify sources of potential conflict. Ratings may be changed, exception elements may be entered into the grid, or new constructs may be added. Refinement may also result from traits that are volunteered at any time or by editing traits, elements, concepts, and ratings. The implications are then re-generated and a new implication tree created. The knowledge refinement cycle continues until the expert feels that a satisfactory level of expertise is reached or the knowledge base is sent on to another KBS building tool.

Showing the user multiple forms of knowledge is an important aspect of ETS. Each representation shift helps the expert think about the problem in a new way and tends to eliminate inconsistencies and conflicts over time. In addition to implication trees and rules, ETS generates reports, listings, journal transcripts, and screen snapshots. When reviewing these outputs, the expert can think of new traits and elements and may often think of new ways to structure the problem. For example, the problem may be divided into subproblems and thus over several rating grids. These reports are also helpful in later knowledge engineering activities. This information is especially valuable in initiating discussions at some later time. The basic vocabulary, important problem traits, an implication hierarchy, and conflict areas are available. This is

important since manual techniques for refinement are still required.

AQUINAS

AQUINAS is an expanded version of an the tool ETS (Expertise Transfer System). The AQUINAS system is an integrated toolset or workbench which interviews experts and helps them analyze, test, and refine knowledge. The components consist of repertory grid tools, hierarchical structure tools, uncertainty tools, an internal reasoning engine, induction tools, and multiple expert tools. The goal of AQUINAS is to solve enough cases so that the knowledge gathered is "sufficient" to solve new cases. The analysis tools and elicitation methods used help the expert think about the current problem in new ways and may also point to conflicts and inconsistencies. AQUINAS is good for analysis problems whose solutions may be enumerated (ie. classification, interpretation, diagnosis). It can be a stand alone personal decision aid, a teaching aid, a group data gathering and negotiation tool, a feasibility study tool, or some other related tool.

Results from consultations are derived from information propagated through hierarchies of repertory grids built through automated interviews. During the initial interview with AQUINAS, the expert is asked to specify a case or set of cases defining a problem where the knowledge is useful. Next the expert is asked to supply a set of possible solutions. Traits are then elicited to distinguish the solutions. Specific traits or solutions may be generalized or broken down into more specific items with the addition of nested grids. Ratings used in building the grids may be determined

directly from the expert or they may be derived through a propagation scheme. The strategies used to build and refine hierarchies include laddering, cluster analysis, and trait-value examination. AQUINAS delivers knowledge by creating a knowledge-based system in one of several different KBS shells. A variety of uncertainty handling techniques are used. These include Mycin-like certainty factors, fuzzy logic, Bayesian probabilities, as well as other combinations.

A Dialog Manager is used as a common interface to the collection of integrated tools. This interface also provides a help facility and may suggest basic expansions or refinements. Consultation review may result in several actions including changing ratings in a grid, changing a trait's weight, adding or deleting a trait, adding or deleting a solution, changing the expectations of outcomes, or re-organizing existing knowledge. Basic learning is accomplished through examples, analogy, observation, and access to session histories. Multiple knowledge sources may also be used, either separately or combined. In the end, results are displayed in a ranked order listing.

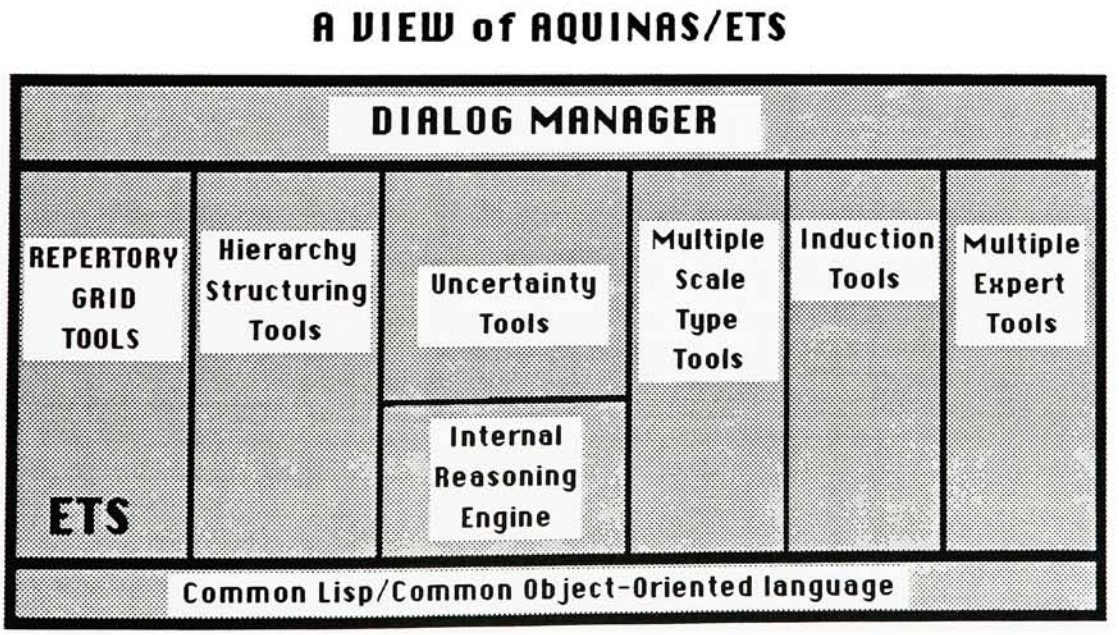


Figure 4.2

PLANET

PLANET (Personal Learning Analysis Negotiation and Elicitation Techniques), another interviewing program based on methods from PCT, combines ideas from systems theory, psychology, and application methodologies. The PLANET system consists of a set of programs for the elicitation and analysis of repertory grid data from a variety of perspectives. Grid analysis techniques used address single grids, pairs of grids, and groups of grids. Some techniques compare and contrast different grids. The set of programs within PLANET can be divided naturally into four groups: construct elicitation, single construct system analysis, multiple construct system analysis, and data administration (DATA, INPUT, OUTPUT, and other programs which provide facilities to manage databases of construct data). The original intended use for PLANET involved the education process. Originally developed in BASIC on a PDP10, it has since been ported to the Apple II.

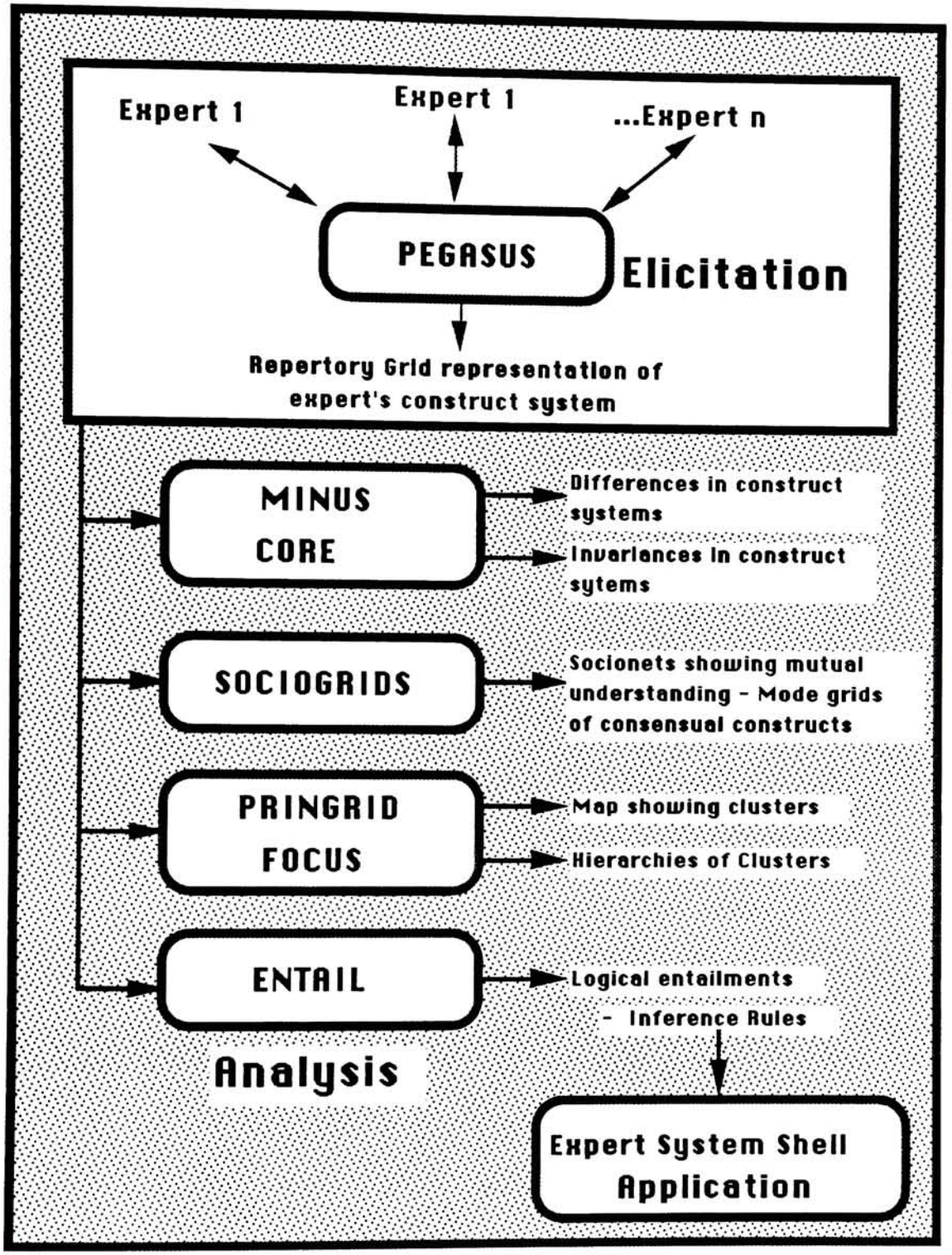
The construct elicitation programs consist of PEGASUS and ARGUS. Pegasus is a suite of tools which elicit constructs through a conversational dialog. MIN-PEGASUS is the version which is closest to the paper-and-pencil technique (no on-going feedback is given, but opportunities to review and revise the content are given). PEGASUS can be used to explore the relationship of an individual with another individual(or group). This program performs on-going analysis of the links between elements and constructs. The most commonly used version incorporates continual commentary on patterns in the responses. PEGASUS-BANK uses an "expert" grid to compare against the elicited constructs of the user. This tool thus gives the capability to compare the current user's constructs with a stored set of expert-derived constructs. PRE-PEGASUS allows the user to continue an

elicitation started at some earlier time. The ARGUS program is a variant of PEGASUS which elicits a set of grids simultaneously from one person holding several roles or points of view.

Single construct system analysis is handled through the FOCUS, PRINGRID, and ENTAIL set of programs. FOCUS provides hierarchical clustering of an expert's construct system. The FOCUS program uses a two-way cluster analytic technique to reorder systematically the rows of constructs and columns of elements to produce a focused grid showing the least variation between adjacent constructs and elements. The focused grid produces results in a form which allows the person to reflect on his patterns of meaning by retaining the original responses, grouped using cluster analytic techniques. FOCI is a version of the FOCUS program which displays an interpretation of the results. SPACED displays another variation of the final printout which "blocks" the focused grid in order to indicate those elements and constructs according to the degree of likeness between adjacent lines. The "focus" algorithm is used upon elicitation of new elements or constructs; not just at the completion of a grid. PRINGRID performs non-hierarchical cluster analysis based on "principal components." Information from this analysis can be used to gauge the major dimensions along which distinctions are being made. ENTAIL conducts a logical analysis of the construct system. The expert's distinctions are taken to be fuzzy predicates. The structure generated can be used as a decision tree expressing the relationship between the expert's data and his conclusions. It can also provide rules for input to a KBS shell.

The multiple construct system analysis programs include MINUS, CORE, and SOCIOGRIDS. MINUS compares two grids based on the same elements and constructs in order to highlight the

similarities and differences. The similarities and differences between two grids are determined by superimposing one on the other. The resulting matrix is then "focused" to identify those constructs and elements being used the same way. CORE provides an interactive comparison of grids to determine where there are differences and when deemed appropriate, extract the invariances. The CORE program can be used to chart change in a person over time and to find a level of understanding and agreement between two people. CORE is also used to determine "core" constructs. SOCIOGRIDS analyzes the entire construct system generated by a number of people construing the same elements from their individual viewpoints. Socionets are produced showing the relationships between individual construct systems within the group. A "mode" grid is also produced reflecting the constructs shared across the group. Shared understanding within small groups can be investigated using the SOCIOGRIDS program.



modified from [SHAW80]

Figure 4.3

KRITON

KRITON makes a basic assumption that no single knowledge acquisition method will be powerful enough to overcome the knowledge bottleneck. Therefore, KRITON has become a hybrid tool employing several knowledge acquisition methods. The system makes use of automated and semi-automated techniques to capture human expertise of static knowledge. Knowledge elicitation may take the form of forward scenario simulation, laddering, goal decomposition, procedural simulation, or pure re-classification and may also involve protocol analysis or text analysis. As originally implemented, KRITON used a Xerox-1108 machine with Interlisp-D using object-oriented features from LOOPS.

The techniques used include automated interviews, text analysis, and protocol analysis. Interview techniques used to explore the domain completely automate a combination of the repertory grid technique, forward scenario simulation, and laddering. The interview conducted produces structured objects at the intermediate knowledge representation level. This representation allows for easy integration of different knowledge sources. Frames, rules, and constraint generators which operate on the intermediate representation are used to build the final knowledge base. Protocol analysis and text analysis are used to add information to the knowledge base as appropriate. A component called "Watcher" is defined as a demon which looks for missing, incomplete knowledge.

NEXTRA

Neuron Data has recently (1990) released NEXTRA for commercial use. NEXTRA is a commercialized version of KSS0. Currently it is a Macintosh-based tool using the grid methodology of Personal Construct theory that interfaces to the Nexpert Object KBS shell. The need for a knowledge engineer still remains though much of the up-front work can be done with the tool rather than manually. The system initially prompts for domain specific information to gain a starting point and also to help develop a vocabulary. It then leads the expert through the process of building a grid. A user of the system does not provide any type of weighting factors. However, approximations are made by the system. Information gathered using the grid technique is shown through various graphical means.

NEXTRA uses several analysis techniques for processing the grid built. Among them are clustering. Clustering is done to determine the degree of similarity between elements and/or constructs. Principal component analysis is also done. Information determined from the principal component analysis is used in an induction-like algorithm to generate relations. In order to keep it simple, no consistency checking or other validation type activity is performed on the results. At any time, however, if the user does not agree with the analysis data, the grid can be changed appropriately. When the user is satisfied, the relations generated are transformed into either frames or rules and possibly objects.

NON-PCT TOOLS

Non-PCT based tools vary tremendously in the approaches taken. Almost every aspect of the knowledge acquisition and elicitation process has been looked at in some way in developing tools. Users of these tools vary from AI programmers to traditional programmers to domain experts. As such, tools may present the structure of the knowledge representation, the problem-solving strategy, or the domain model in various ways. In addition to being geared toward different types of users, tools are also geared toward different types of problems. Even with these many existing tools, nothing comes close to an ideal tool. The common approach is to generate a tool to select a problem-solving strategy and then select the most appropriate technique. The following sections describe, very briefly, a few of these tools.

TEIRESIAS

TEIRESIAS provides a number of tools which are designed to aid in the construction, maintenance, and use of a knowledge base system. The knowledge acquisition system was written in Interlisp to work with MYCIN, one of the first Knowledge-Based Systems to gain wide recognition. The system acquires new rules about the problem domain through an interaction that allows users to state rules in a restricted subset of English. Although the user has the final say, the system may make suggestions regarding completeness and consistency. If necessary, the system will help the user debug "problem information." TEIRESIAS originally was developed to run on a DEC PDP-10.

The goals of TEIRESIAS were two-fold. First, it should be possible for an expert in the domain of an application to "educate" the production system interactively, commenting on and correcting its behavior. Secondly, it should be possible for the expert to assemble and maintain a large body of knowledge. The central theme used to meet these goals was the exploration and use of meta-level knowledge. An attempt was made to make possible a system with both the capacity to use its knowledge directly and the ability to examine it, abstract it, and direct its application. Thus the program contains both object-level representations describing the external world and meta-level representations describing the internal world of representations [DAVIS82]. Other themes found in the approach included: 1) task-specific high-level languages make code easier to read, 2) knowledge in programs should be explicit and accessible, 3) programs can be given access to control and use an understanding of their own representations, 4) programs can be self understanding (representations, data structures, behavior), 5) a representation can usually be more than a densely encoded string of bits, 6) A program can have some grasp on its own complexity, and 7) programs can be self-adjusting.

TEIRESIAS can be viewed as a teacher who continually challenges a student with new problems to solve and carefully observes the student's performance. Deduction is used to keep questions at a minimum. The knowledge acquisition process is viewed as information transfer from an expert to a program (described as "interactive transfer of expertise"). The system does not attempt to derive new knowledge; it "listens" and comments appropriately to help the expert augment the knowledge base, thus requiring strong cooperation from the expert. Some of this is handled through a small amount of natural language handling. The techniques used make it possible for an expert to teach the system new rules which are expressed in terms of known concepts. Other

methods are used to teach the system new conceptual primitives and new types of concepts. A purely statistical approach is used for concept formation. To some degree, a model-based approach is also used. A debugging process exists to help clarify and validate knowledge. This debugging process is part of the consistency checking activities. The focus is on adding knowledge to a knowledge base.

Knowledge, stored in the form of rules, is basically of two types. Some knowledge is used to describe the system at different levels of detail while other knowledge is used to classify knowledge according to its level of generality. This classification knowledge, known as meta-knowledge, can exist as rule models, data structure schemata, or meta-rules. Rule models are the content of inference rules used for acquisition of new rules. Data structure schemata is the structure of conceptual primitives used for acquisition of new conceptual primitives. Meta-rules describe the use of object-level rules which guide the use of object-level knowledge. These rules are handled in a way such that they can accommodate inexact knowledge.

KREME

KREME (Knowledge Representation Editing & Modeling Environment) is and experimental environment (using frame representations and graphical displays) for developing and editing large knowledge-base systems in a variety of representation styles. The goal in building this system tool was to build an environment in which existing knowledge representation languages could be integrated and organized as components of a coherent global representation system.

Validation and consistency checking is a vital part of the architecture and is implemented through a strong notion of meta-level knowledge. The tool also has Mycin-like certainty factors for handling uncertainty and contains a mechanism for combining information from multiple knowledge sources into a single unified whole. Some of the other features include the existence of an explanation facility, and a history and tracing mechanism. The capability also exists to build meaningful generalizations based on the available base of knowledge.

Special purpose editing facilities have been included in KREME which permit knowledge to be viewed and represented in several formalisms. Editors exist within KREME for four distinct representation methods. These methods include frames, rules, procedures, and attached behaviors or methods defined as functions. For each of these, one or more editor views may exist. All of the definitions manipulated by any editor are read and stored in lisp-readable text files of "defining forms." These forms allow editing to occur outside of the KREME environment. The KREME tool is built in such a way that other knowledge representation schemes can be added with little difficulty. All of the knowledge representation techniques are implemented as combinable objects using FLAVORS (a version of Lisp).

SALT

SALT is a model-based system that interviews experts in the configuration domain. It was first used to configure elevators. SALT is intended for use by domain experts to create and maintain systems that perform constraint-satisfaction tasks. The approach used by SALT is successful

in that the expert can define the search tree used in problem-solving. SALT uses a "propose and revise" method for constructing systems. The process is an iteration of defining a plan, identifying constraints on that plan, and then proposing a new plan which identifies fixes for the constraint violations. Among other features, an explanation facility exists.

Much of the power of SALT comes from its ability to make strong assumptions about the problem-solving strategy used by the expert systems it creates. SALT acquires knowledge from an expert and generates a domain-specific knowledge base compiled into rules. These rules are then used by a system shell to create an expert or knowledge-based system. Knowledge is represented according to the role it will play in the problem-solving strategy. These roles include: 1) propose a design extension, 2) identify a constraint, and 3) propose a fix. SALT also does consistency and completeness checking to assure convergence toward a solution. However, SALT is weak in its ability to elicit knowledge to handle tradeoffs among alternatives.

INFORM

INFORM (INfluence diagram FORMer) is a domain-independent expert directed knowledge acquisition aid for KBS development. It is a system intended to replace, at least in part, the expertise of a knowledge engineer. This tool applies decision-analysis knowledge engineering in the form of influence diagrams as a knowledge structure and computational representation. The processing uses Bayesian probability for uncertainty, influence diagrams combined with software engineering tools, and performance refinement

techniques. Three performance goals indicate what INFORM is intended to achieve as a knowledge acquisition tool. These goals address sufficiency, correctness, and the ability to provide insight into the application domain. As with many other knowledge acquisition tools, hints and suggestions are made for refinements. INFORM is seen as best fitting applications which use heuristic classification problem-solving.

The INFORM tool is implemented in the C language and automatically generates complex influence diagrams based upon input received. Influence diagrams graphically represent the structure of the decision problem but maintain the computational utility of a decision tree. Using this scheme, there are three layers of knowledge representation - relational, functional, and numerical. This hierarchy tries to capture the way people tend to model knowledge. This tendency is thought to be simple to complex and from conceptual to numeric. The decision-analysis cycle is used to capture knowledge. This cycle is an iterative and interactive technique for assuring that essential steps in the decision process are captured.

TOOLS SUMMARY

The tools presented here are only a representative few that exist in various research or commercialized states. Many of the tools discussed and others have acted as rapid prototyping tools for generating knowledge-based systems. They all basically create a meaningful "shell" of a system that exemplifies what is required. Without such tools the initial "priming" of a knowledge-based system shell is a difficult task. In the absence of automated tools, the knowledge engineer has a major role to play as an

intermediary between the expert and the system. With such tools, the expert becomes the dominant player in the knowledge acquisition process. The knowledge engineer can thus concentrate on more technical aspects of development and support the expert where needed. This shift in roles goes a long way toward solving communication related issues and reduces the time required to build production systems.

CHAPTER 5 KE-KIT IMPLEMENTATION

IDEAL KNOWLEDGE ACQUISITION TOOL

An ideal architecture of a knowledge acquisition system would have a natural language interface (possibly combined with graphics) that interacts with a dialogue or knowledge elicitation manager. This portion of the system would be responsible for gathering knowledge. The next level would include a "knowledge translation" manager consisting of several functions. These functions should include activities for consistency checking, induction, internal reasoning, uncertainty handling, and knowledge base generation (the AI version of code generators). In theory, the knowledge base generation aspect of the system could create ANY knowledge representation scheme that was considered appropriate for the domain or target application environment. This really means that the system would automatically create rules, frames, or other objects and make them available for use by a knowledge-based system. Taking this a degree further would mean that rules, frames, or objects for several different underlying implementations could be created. For example, once it was determined that rules would be generated, rules could be created for PROLOG, VP-EXPERT, PC-EASY, LEVEL 5, or any other KBS tool available for application development. An example ideal tool architecture is shown in figure 5.1.

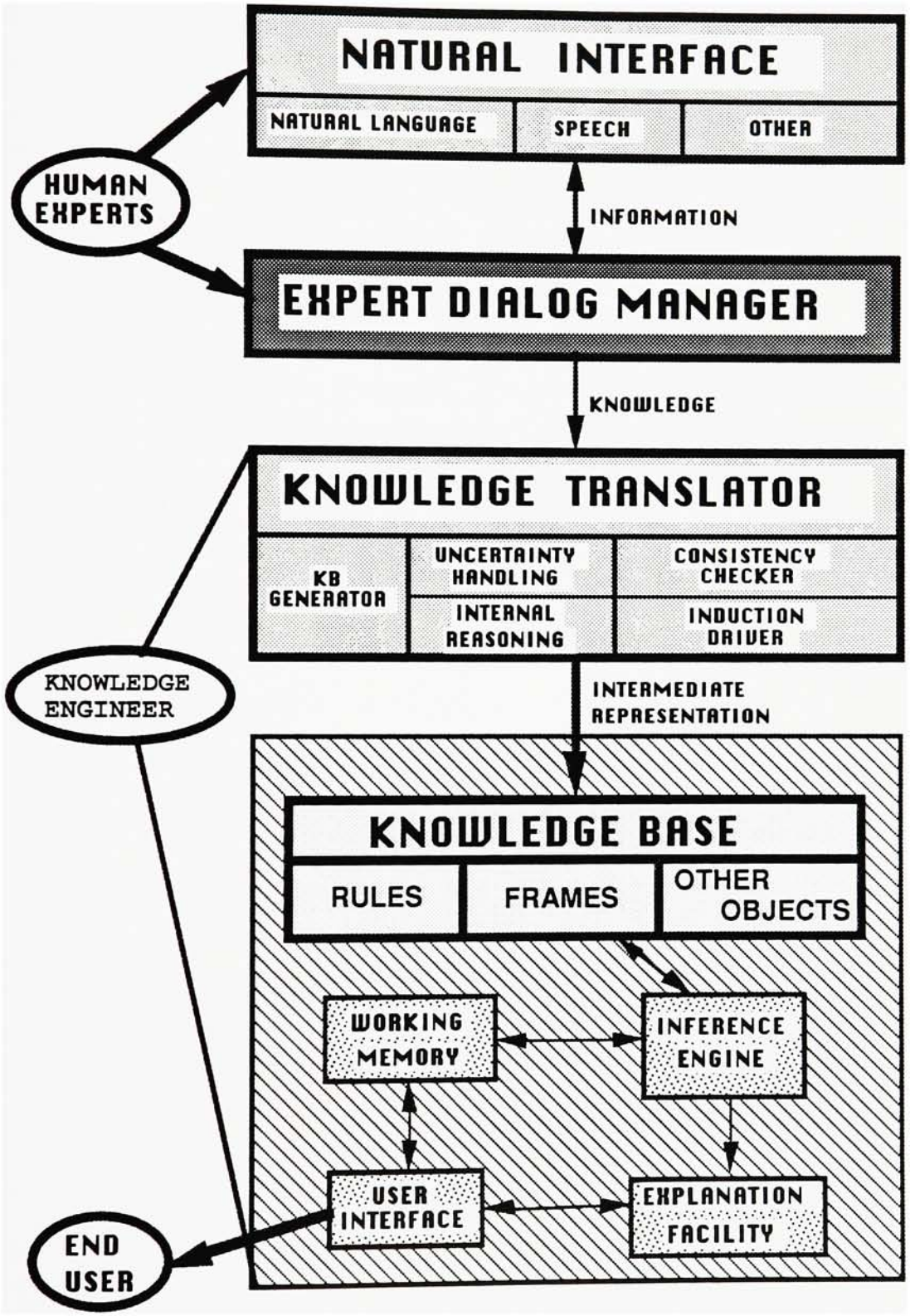

NATURAL INTERFACE
NATURAL LANGUAGE
SPEECH
OTHER
HUMAN EXPERTS
INFORMATION
EXPERT DIALOG MANAGER
KNOWLEDGE
KNOWLEDGE TRANSLATOR
KB GENERATOR
UNCERTAINTY HANDLING
CONSISTENCY CHECKER
INTERNAL REASONING
INDUCTION DRIVER
KNOWLEDGE ENGINEER
INTERMEDIATE REPRESENTATION
KNOWLEDGE BASE
RULES
FRAMES
OTHER OBJECTS
WORKING MEMORY
INFERENCE ENGINE
USER INTERFACE
EXPLANATION FACILITY
END USER

Figure 5.1

Whatever the architecture, the system created should be flexible, easy to maintain, easy to use, and contain a high degree of validity. With this in mind, it makes sense to start simple. A goal should be to acquire knowledge and expertise while storing it in a simple knowledge representation. In this way advanced features can be added later with relative ease. Thus, a good basis for an automated knowledge acquisition tool would be a framework that could grow. This basic framework should consist of a "Dialog Manager" performing knowledge acquisition, a "Knowledge Translator" (performing some degree of induction, reasoning, and consistency checking), and a "Knowledge-Base Generator" that will generate rules usable by a Knowledge-Base system shell to test and execute the knowledge gathered. Of course, other utility and user-interface functions are also required.

KE-KIT

INTRODUCTION

KE-KIT (Knowledge Engineering - tool KIT), is a first step toward an "ideal" tool. The tool itself is a "proof-of-concept" or prototype system. Its main objective is to implement the repertory grid (based on personal construct theory) approach to knowledge acquisition. The second main objective is to generate a knowledge representation usable by an existing knowledge-base system tool. This representation should be flexible so that various other target formats can be created. Other objectives considered when designing and implementing KE-KIT include the ease of expansion, the ease of use, and the inclusion of associated knowledge engineering

functions that might be needed. The ultimate goal is for KE-KIT to be usable by a domain expert with little or no help from a trained knowledge engineer.

Many aspects of the basic design are similar to the ETS tool. The main driver behind the Dialog Manger of KE-KIT is the repertory grid approach to knowledge acquisition. In its current state a one-grid-per-problem approach is a used while allowing for an extension to a hierarchy of grids. Most of the interviewing process is initiated by the system user and the use of menu options. One of these options gathers limited background material on the application being addressed. The option responsible for the "core" knowledge elicitation is almost entirely dependant upon the user. The only exception is the method and presentation used. The Knowledge Translator uses an implication analysis function based on a decision-action table approach to transform the elicited knowledge into a usable form. The Knowledge-Base Generator implemented presents a menu of target KBS shells/tools, however only one (VP-EXPERT 2.0 using rules) is functional at this time. See figure 5.2 for a high-level view of KE-KIT.

Some additional functions have been implemented to support the basic knowledge acquisition and knowledge-base generation processes. An application evaluation function is included to help the user decide if the proposed application and the domain are suitable. Also, several facilities exist for reviewing information either in a printed form or by an on-line display. Other support functions include a system overview, a help listing, and a defined list of terms. These three functions can be used in any combination to help an inexperienced user through the use of KE-KIT.

The "class" of problems to be considered by KE-KIT all relate to analysis in some way (see figure 1.1). Problem areas

including classification, diagnosis, and selection, are all candidate applications for KE-KIT. In each of these problem areas the conclusions or recommendations can be enumerated prior to problem analysis. These problem areas, since they are considered easier to implement, allow greater emphasis to be placed upon the KA process rather than on complex problem solutions. It is believed that the diagnosis and selection type problems can be handled through the rather simple classification approach. These "simple" problems also make the knowledge representation approach and the knowledge-base generation function easier to implement. This is true since most such problems can be addressed using basic rules rather than more complex representations.

KE-KIT is implemented on a PS/2 model 55sx (a 80386 processor) running DOS 4.0 using Turbo-C (version 2.0 using the Large memory model) by Borland and a set of C library routines called "C Utility Library" (version 5.0) from South Mountain Software. Other hardware used include a color VGA (vector graphics array) monitor and an Epson (model LQ-510) printer. The additional library of routines is used to implement various menus, windows, and other functions not readily available via the C compiler. This tool set was used because of its accessibility, convenience, and the perceived future of system processing and development environments. Several other utility libraries were evaluated and were used at various phases during implementation. The "C Utility Library" was chosen because of its ease of use, its low learning curve, its ease of debugging, and its relatively well written documentation. It is hoped that using this implementation environment will allow much flexibility for further enhancements.

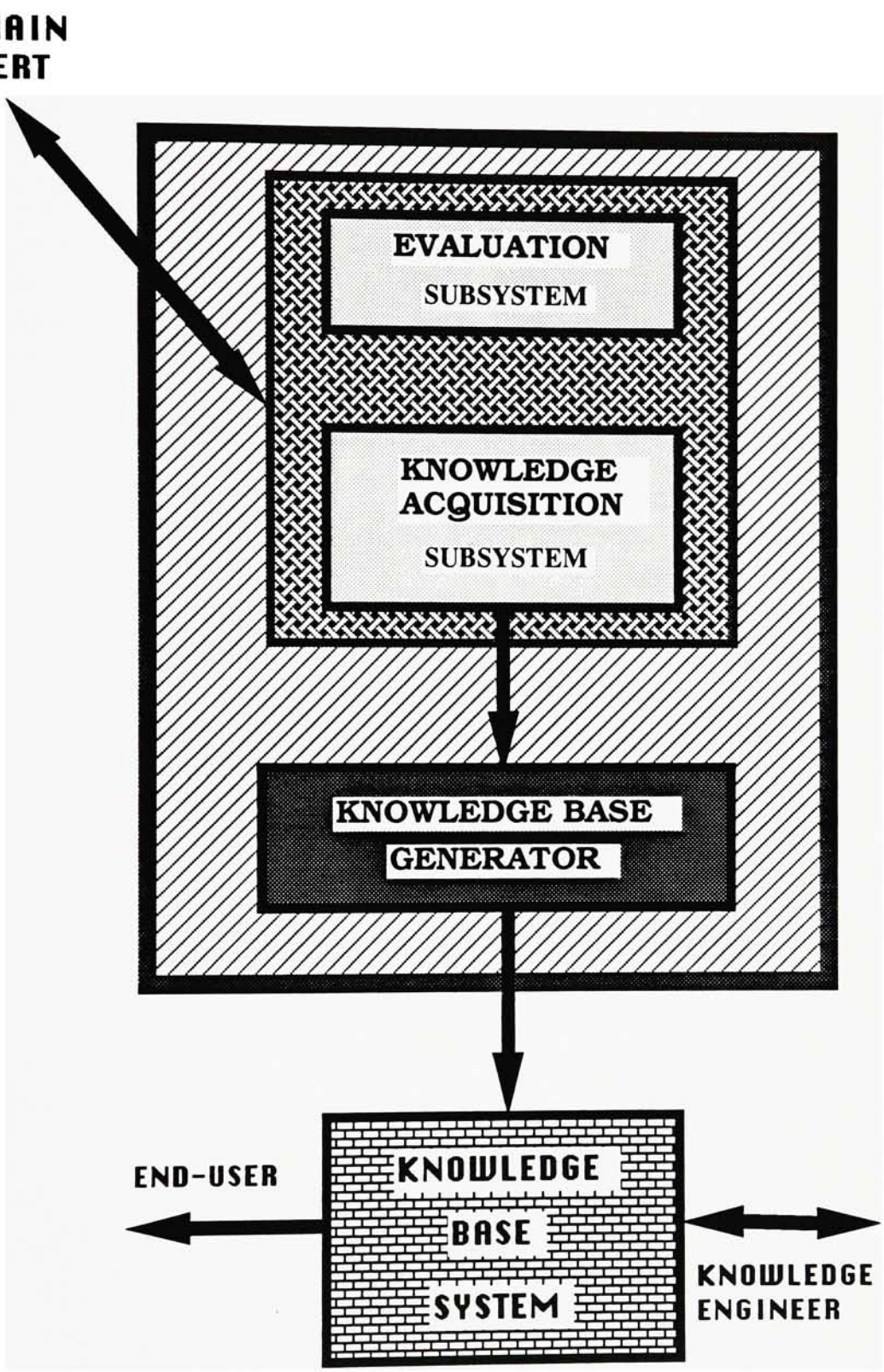

DOMAIN EXPERT
EVALUATION SUBSYSTEM
KNOWLEDGE ACQUISITION SUBSYSTEM
KNOWLEDGE BASE GENERATOR
END-USER
KNOWLEDGE BASE SYSTEM
KNOWLEDGE ENGINEER

Figure 5.2

DEVELOPMENT APPROACH

Several assumptions and limitations had to be defined about the implementation of KE-KIT in order to limit the scope yet still make the program worthy. The main objectives of demonstrating the knowledge acquisition technique and subsequently generating a knowledge-base tool format was an overriding factor in many decisions (not only in the design, but also as implementation progressed). Before implementation of KE-KIT began, a high-level design was created. Figures 5.2 and 5.3 represent this high-level design. The detail design mainly consisted of a set of data flow diagrams (see the appendix B). These tools were enough to define an "initial" program. Even with the use of these traditional software building tools, some coding techniques and approaches did not become clear until after several were actually coded and tested. In many cases, a new approach was necessary before testing began.

Initially, a skeleton system was developed. As new routines were coded, they were tested as much as possible individually before integrating them into the overall program. In affect, a prototyping approach was used to develop the deliverable program. Many of the routines could still be tested individually at later stages if major changes or some other reason warranted isolated testing. The implementation process tried to follow the initial design as much as possible. However, not far into the process it became apparent that the scope had to be refined and tightened. Complexity and lack of experience were sufficient to change some initial plans. Some aspects of the original design would require much greater research, design, and implementation time (eg. multiple-grid approach, implication analysis, etc.).

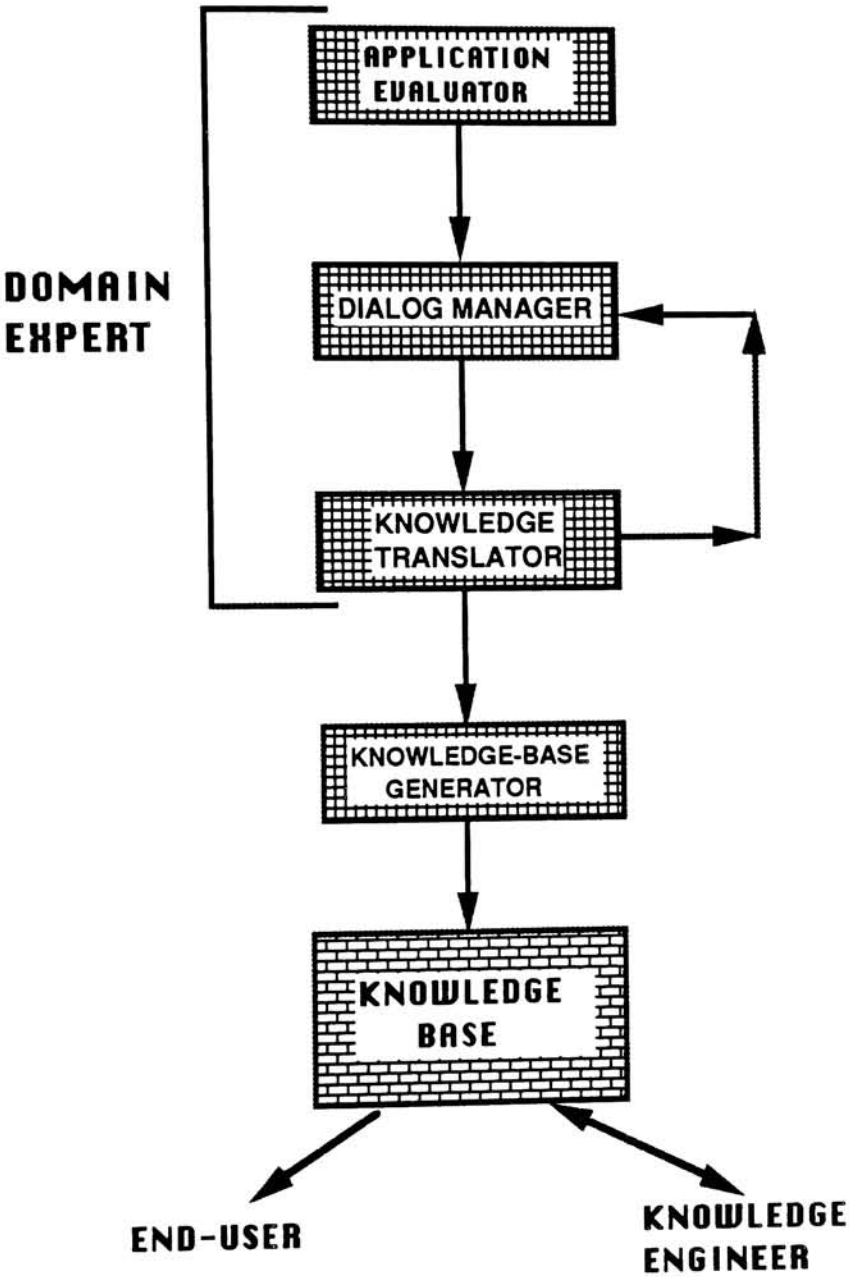

APPLICATION EVALUATOR
DOMAIN EXPERT
DIALOG MANAGER
KNOWLEDGE TRANSLATOR
KNOWLEDGE-BASE GENERATOR
KNOWLEDGE BASE
END-USER
KNOWLEDGE ENGINEER

Figure 5.3

ASSUMPTIONS

Several assumptions about the final "prototype" had to be made in order to make it manageable, useful, and still beneficial. The system implemented must prove itself as a worthy attempt toward the objective of automated knowledge acquisition and should also allow enhancements with relative ease. Most of the initial assumptions should be altered and the restrictions lifted in order to to implement enhancements at some later time. However, they are beyond the scope of this "prototype." The details of KE-KIT and how these assumptions are addressed are described in another section.

One expert or knowledge source is assumed for each user-defined application. The acquired knowledge is taken with complete certainty. Uncertainty of any kind is not handled via the knowledge or the rules that may be generated. The control strategy assumed for all applications is backward chaining. This is results from the type of problems to be addressed by KE-KIT. As previously mentioned, the conclusions or recommendations can be enumerated prior to any problem analysis. In this way, the application has a clear scope and definition. Any information not explicitly given is assumed not to exist and does not apply to any conclusion or recommendation.

A repertory grid technique is used as the basic technique for knowledge acquisition. The GRID technique uses a rating scale of 1/0 which is converted to YES/NO for system use. Only one grid is used for any developed application. A hierarchy of grids is not implemented due to the added complexity involved. Rudimentary logic-based analysis is used to generate implications and subsequent rules. A decision-action table approach is used. See figures 3.7a-3.7c for an example of this process.

The target knowledge representation is a rule structure. Thus the knowledge acquired must be convertible to a rule format. "Simple" compound rules are used (ie. no "or" conditions are used, but "and" conditions are commonly used). Any other rule format would be considered an optimization and would require "knowledge analysis." Other knowledge representations are not considered. However, as much as possible, the program is coded in such a way that it can be expanded to included multiple knowledge representations and multiple KBS shells or tools.

The capability to review given information exists. Acquired knowledge may be viewed "on-line" and/or in a printed form. This allows for limited validation and consistency checking. However, information can not be changed - only added. No external system interfaces (ie. databases, spreadsheets, etc.) are supported. The "deliverable system" is basically a knowledge acquisition tool with the added features of an application evaluation subsystem and a knowledge-base generator. Additional functions must be handled outside of KE-KIT. All final knowledge-base system activities (ie. enhancements, interfaces, testing, user interface creation, etc...) is the responsibility of a knowledge engineer. KE-KIT also includes a few basic file utilities and three related help functions in order to make the tool usable and easy to learn.

DETAIL DESIGN

KE-KIT is menu driven and is divided into three main functions. These include "File" utilities, "Knowledge Engineering" functions, and "Miscellaneous" support functions. Most of these major functions have underlying

routines. Since KE-KIT is menu driven, very little of the functions guide the user through the process. As a result, a typical user should consult the help facilities and/or have some experience using the system before embarking on a "real" application. The menu structure as implemented lends itself very well to modifications of most any kind. Thus, at some future time, new options can be added or options can be re-located or deleted.

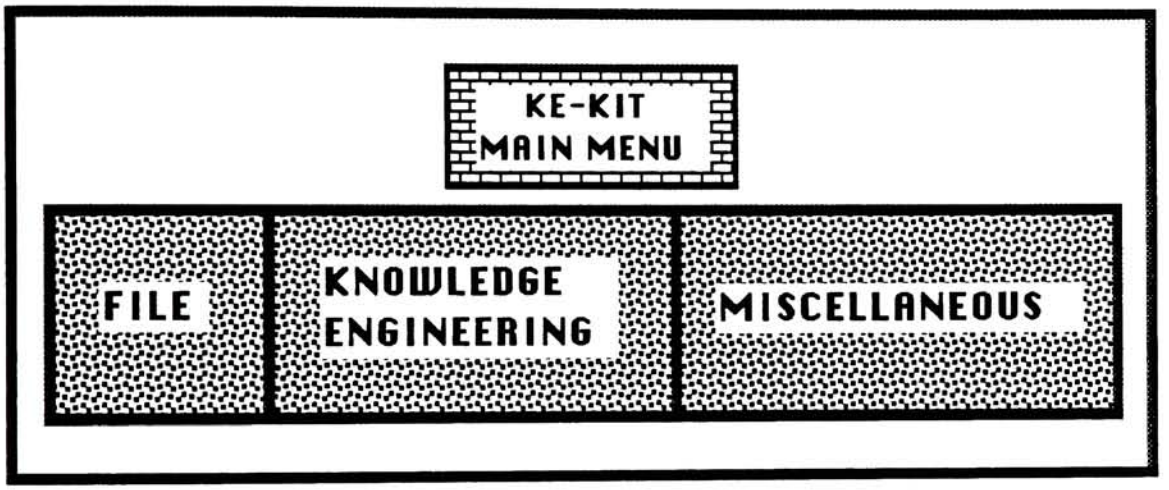


Figure 5.4

Each menu option, at all levels, has an associated description text string displayed as the option is highlighted. This text string is a minimal form of functional help for the option. Several options display appropriate message screens after execution in order to relay execution information to the user. For example, the few options that are available on the menu structure and not implemented display a message stating as such. Other functions, which execute entirely in the background (ie. implication derivation), display successful termination messages or error messages as appropriate.

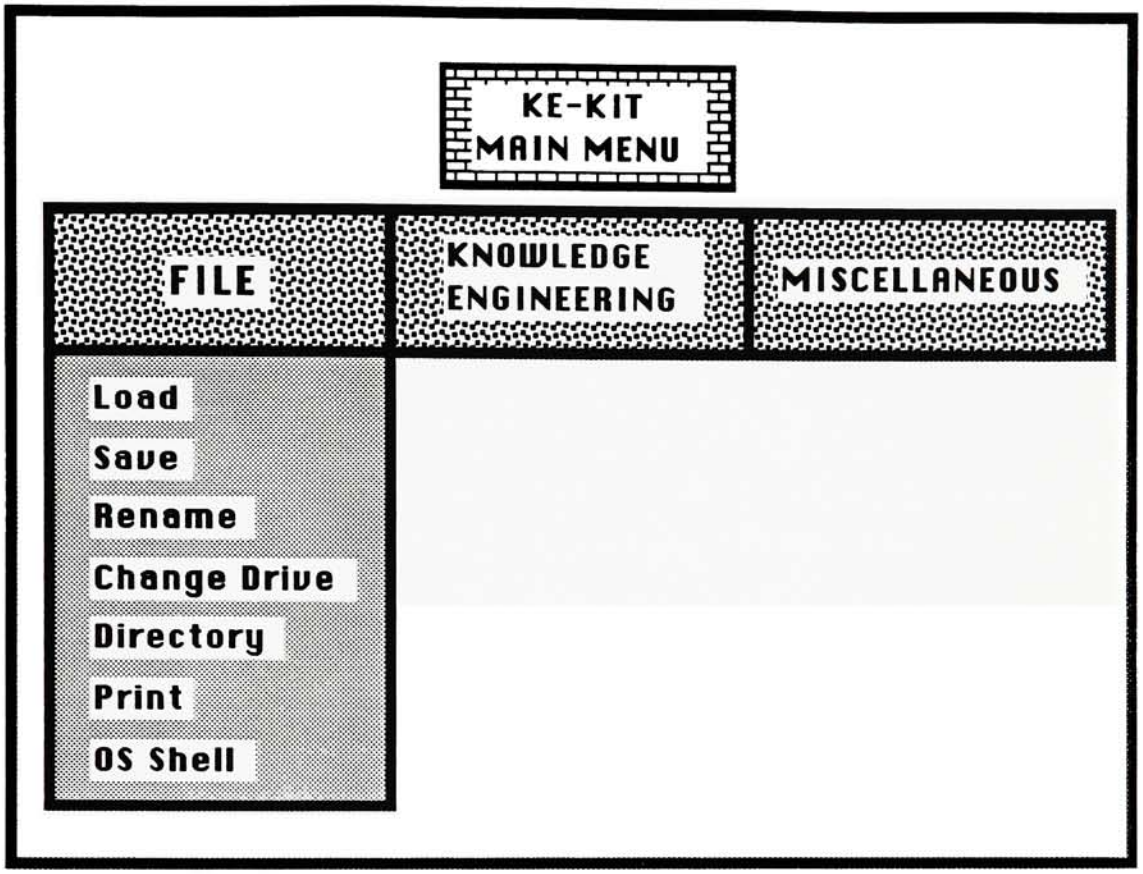


Figure 5.5

The file options given are basic in nature. However, not all of them are implemented. The implemented functions include "LOAD," "SAVE," "DIRECTORY," "PRINT," and "OS SHELL." The load function will load into working memory a previously saved application. Anything currently in working memory is lost. When the save option is selected, the information currently available (application background, elements, constructs, the rating grid) and the unused set of triads is saved to a given file name. The file name used is given an extension of "KIT." An example of saved knowledge can be found in figure 5.22 and Appendix F. The key to the format is listed in figure 5.23. Using this format, it is possible to change or add information by updating the saved file appropriately. If the current information is not saved and the program is halted for any reason, the information will be

lost. A directory listing of all available KE-KIT applications (those with file extensions of "KIT") in the current directory are made available with the directory function. All available knowledge known to KE-KIT for the current application is made available for printing using the print function. The user is given the option to choose various subsets of the knowledge or to print all available information. As implemented, the print utilities expect a printer to be available and ready to print. The printer is also assumed to be an Epson printer. Any of the non-implemented functions or other DOS based functions can be executed using the "OS SHELL" option. This option escapes out of KE-KIT temporarily and allows the user to execute other programs or utilities (ie. rename a file, delete a file, check space, etc.) from the operating system. Thus, file utilities not implemented within KE-KIT can be executed. Typing "EXIT" at the DOS prompt returns execution to KE-KIT.

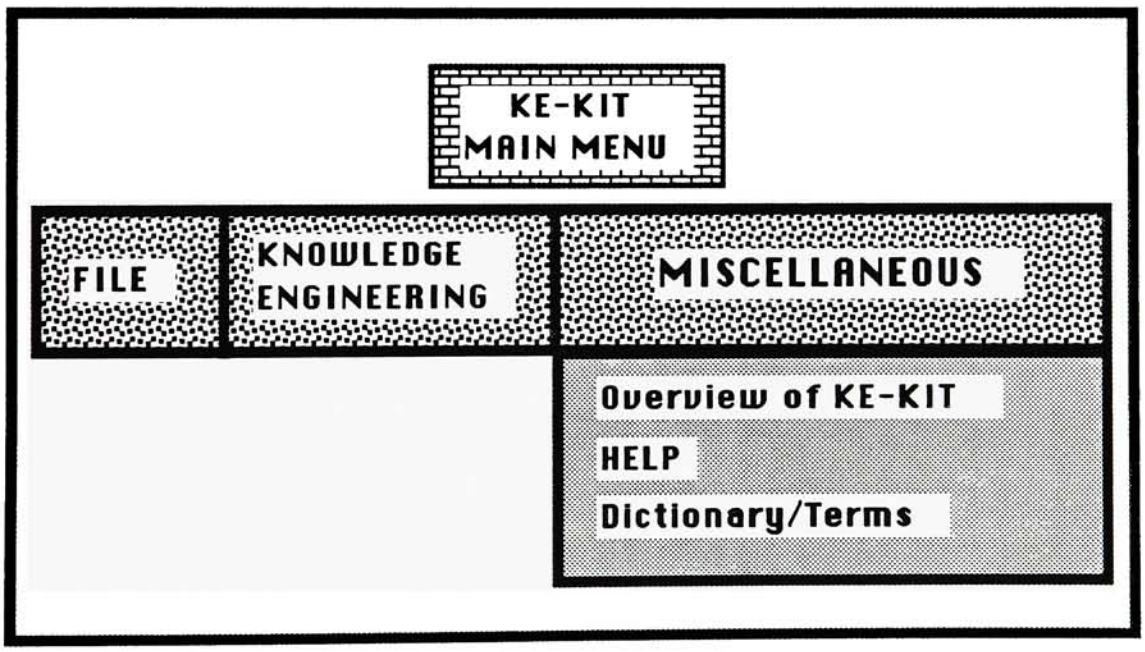


Figure 5.6

The current set of miscellaneous support functions are all related in some way to a help facility. Three options are

available. These include a system overview (OVERVIEW of KE-KIT), a system help facility (HELP), and a listing of terminology (DICTIONARY/TERMS). The dictionary function provides an in-depth description and definition of terms used within the system or related to knowledge engineering that a typical domain expert may not be familiar with. This is especially true of some of the terminology used with the knowledge acquisition process. All of these functions are implemented in such a way that the contents can easily be changed at any time. Each option has an associated "external" file that is displayed whenever selected. Once displayed, the user can scroll through the file either via a line at a time or a page display at a time. Also, several other browse commands are available (see the HELP listing in appendix D).

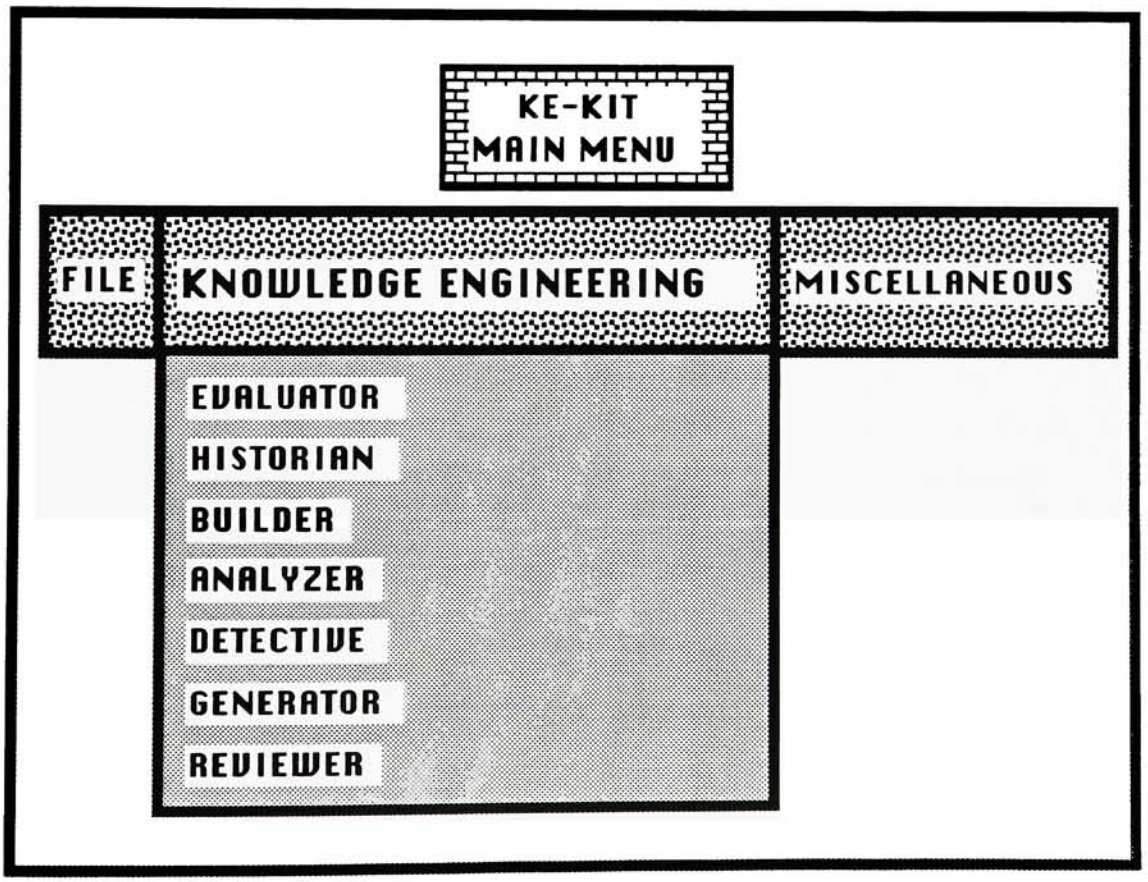


Figure 5.7

The crux of KE-KIT is the suite of knowledge engineering functions available. These include "EVALUATOR," "HISTORIAN," BUILDER," "ANALYZER," "DETECTIVE," "GENERATOR," and "REVIEWER." All of these are implemented except for ANALYZER. With the exception of EVALUATOR and HISTORIAN, all of these functions are tied together and collectively comprise the knowledge acquisition process. DETECTIVE and GENERATOR are used to determine knowledge implications and subsequently transform this information into a KBS shell usable format. See figure 5.8 for a view of the knowledge engineering functions.

EVALUATOR is the function within KE-KIT which allows a user to assess a proposed application for its feasibility. Four main areas are tested. These areas include system justification, domain expertise, problem characteristics, and group/political characteristics. Each area poses several questions to the user (see the appendix G for specifics). The user can answer each question by using a scale. Definitely is represented by "6," probably is represented by "4," maybe is represented by "2," and no or not applicable is represented by "0." Within each area, a weighting factor is used to place importance on various questions. Once all four areas have been addressed, a recommendation screen is displayed. A statement concerning each area is presented along with an overall recommendation. Potential problem areas may be identified and presented. Over time, this feature may teach users enough so that they can identify other KBS applications with high potential for success. The recommendation is only a high-level statement of the application feasibility and likelyhood for success. No information from this function is used in any further processing by KE-KIT. The user must make up his or her own mind about continuing.

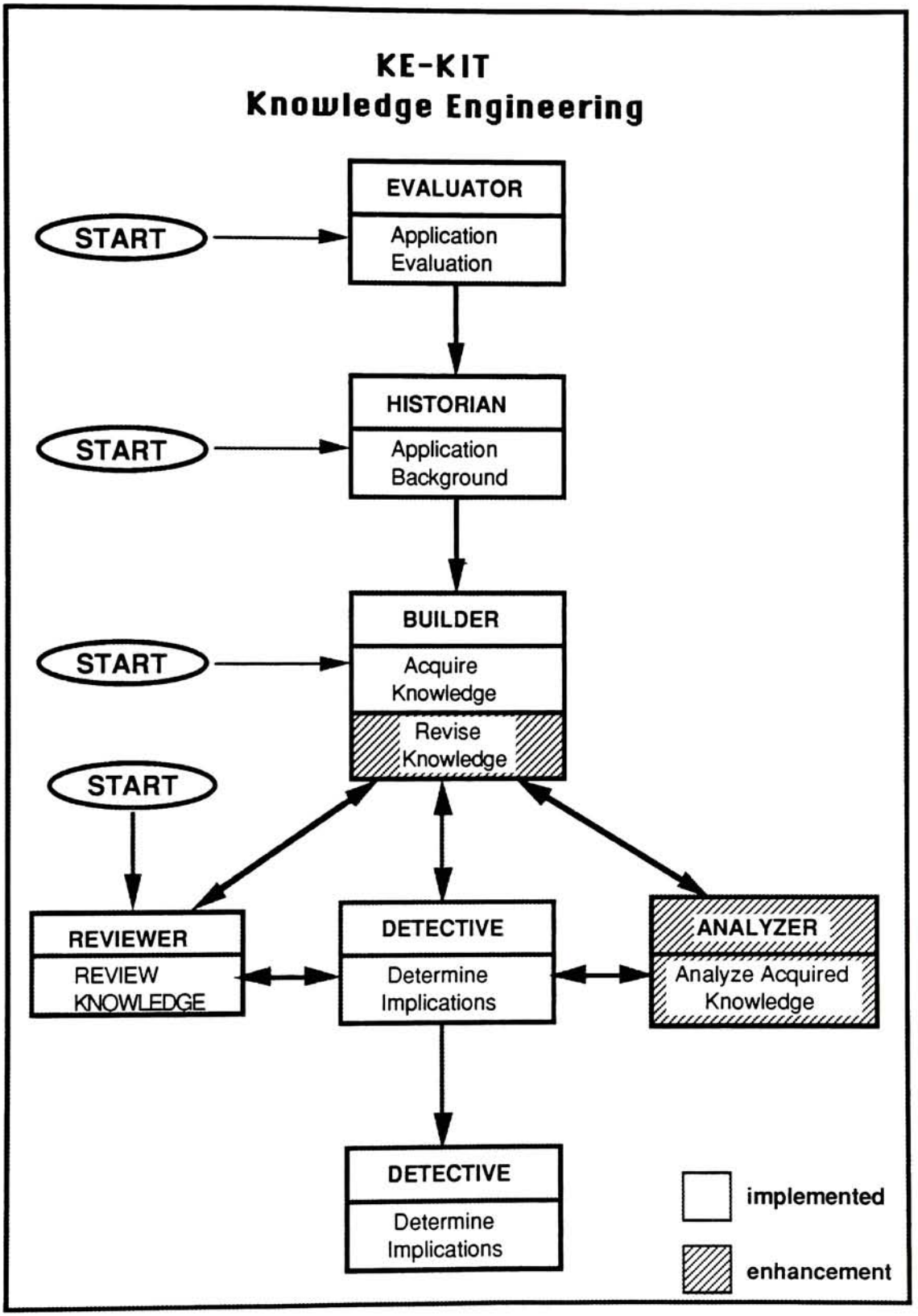

KE-KIT
Knowledge Engineering
EVALUATOR
Application Evaluation
START
HISTORIAN
Application Background
START
BUILDER
Acquire Knowledge
Revise Knowledge
START
START
REVIEWER
REVIEW KNOWLEDGE
DETECTIVE
Determine Implications
ANALYZER
Analyze Acquired Knowledge
DETECTIVE
Determine Implications
implemented
enhancement

Figure 5.8

The HISTORIAN function collects basic background information concerning the application being addressed. This is an optional function. However, the information is made available for later viewing via REVIEWER or the print facility. As a result, all requested information is initially assigned default values indicated as such. There are several pieces of requested information presented in a "form" format. The first, is intended to be the system user - the person making available their expertise. The user is then asked for an application name or title. Next the expert or experts are requested. In most cases, this should include the KE-KIT user. If multiple experts are involved, all should be included. Following this information, the intended users are requested. This should include anyone in the target audience of the application under development. The last two bits of information are directly related to the problem being addressed. First, the problem statement is requested. As much as possible about the problem being addressed should be entered. Lastly, the application purpose is requested. It is not likely that the application under development will address the entire problem stated or in existence. Therefore, this statement defines the scope of the application. Once given, none of the background information is changeable directly within KE-KIT. However, a saved application file can be altered to change the contents of the background data. The specifics of this is given at the end of this chapter.

BUILDER is the function within KE-KIT which performs the major knowledge engineering function of knowledge acquisition. Specifically, BUILDER implements a repertory grid method of knowledge acquisition. This approach has its beginnings as part of Personal Construct Theory. Knowledge acquisition is performed with the help of several routines. These routines elicit elements which represent various conclusions or recommendations, constructs which represent

traits or characteristics which distinguish the elements, and ratings that apply each element to the constructs.

The initial knowledge acquisition process begins by asking a user for a set of six elements. In addition to the elements, an optional description or further text of any kind can be given. However, this information is only used as a form of documentation at this time. It is assumed that six different conclusions can be reached in solving the "problem" at hand. This number of required elements was chosen to allow sufficient processing in other aspects of the acquisition process and also to create a minimal solution set.

Immediately after collecting the initial set of elements, the system automatically creates a set of triads. The method used to generate this list is based on a random number generator. Given this set of elements, the number of combinations possible for the set is calculated (in this case six elements produce twenty different combinations). Subsequently. each combination of elements is generated and stored as part of a linked list (see figure 5.11). It is hoped that these combinations are sufficiently random so that the triads presented can adequately help the user define constructs.

Construct elicitation begins with the presentation of the first set of three elements. Once this triad is used, it is removed from the triad list and is no longer available. For the presented elements, the user is expected to identify which two are similar in some way while being different from the third. Once this is done, labels or definitions are given. The left pole of the construct is given a label for the similarity while the right pole is given a label for the difference. In addition, the user can optionally provide a construct name and a description. However, this information

is not used other than as form of documentation in the current state of KE-KIT.

Next the process begins building a rating grid. The current defined construct and the current triad are presented to the user and is asked to rate the elements. A rating of "1" is given to an element if the left pole of the construct applies and a rating of "0" is given if the right pole applies. Elements which do not apply to either pole of a construct are given a rating of "N" and a rating of "B" is used if both poles apply. The emphasis is placed on the left or emergent pole and thus the rating of "1" can essentially be taken to mean "yes." A rating of "0" then means that the element does not apply to the emergent pole (it applies to the implicit pole). This emphasis becomes important when laddering is implemented (a future enhancement). Once the current triad of elements is given ratings, all remaining elements are put through the same rating process. This is known as the "across method" of assigning element-to-construct ratings.

After rating each element on the current construct, the knowledge acquisition process continues. The user is asked if more information is available. If this question is answered in the affirmative, the user is then asked if the current triad is to be used again or if another one should be used. This gives the user a chance to define several constructs for the same set of three elements. This process continues until either the user has no further information to provide or the system defined limit of fifteen constructs is reached. This limit is arbitrary. However, a multiple grid implementation would allow fifteen constructs per grid.

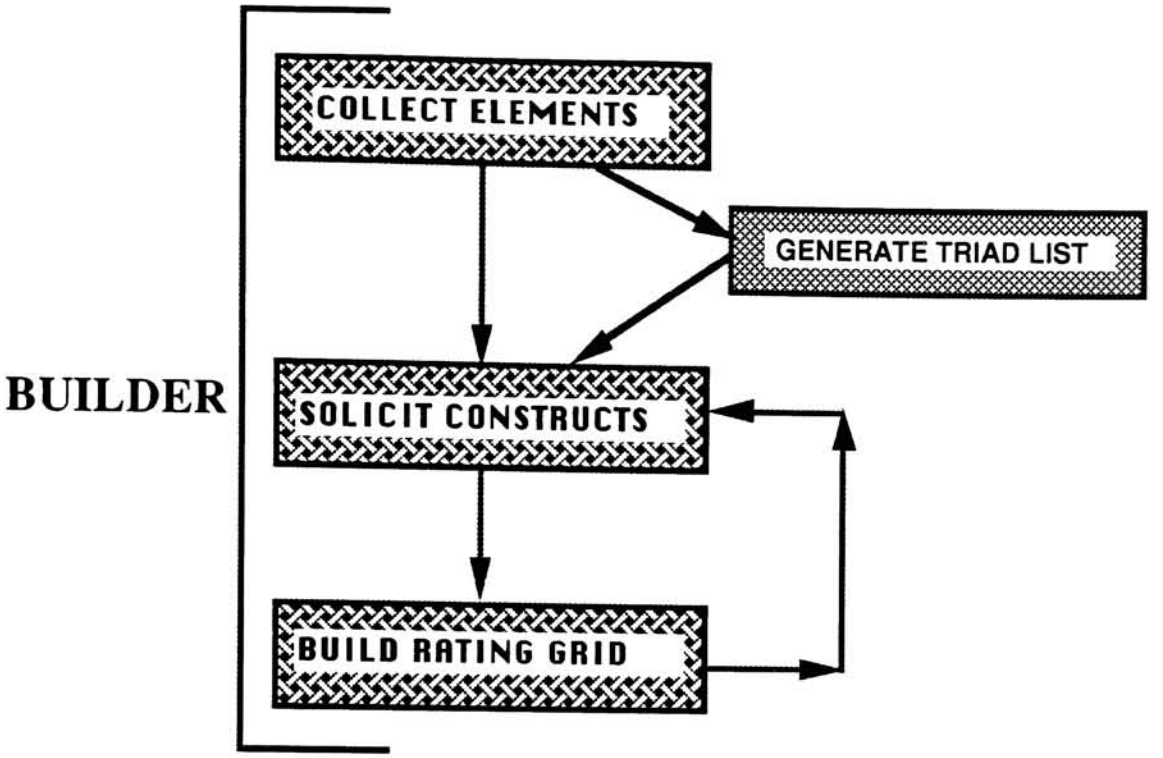


Figure 5.9

Any and all implications available are detected using DETECTIVE. This function can be executed at any time. However, unless some information is first processed via BUILDER, nothing will result. At a bare minimum, one construct must exist with ratings for each solicited element (ie. a minimal rating grid is available). A decision-action table approach is used to generate implications (see examples in Appendix F). In this way, elements become conclusions and construct pole labels become conditions. Using this method, one compound implication will result for each element.

The construct pole label used for generating implications depends upon the rating given. A rating of 1 indicates that the left or emergent pole label will be used, while a rating of 0 indicates the implicit or right pole is used. Using this method avoids the need for negative logic. The absence of any explicit condition (ie. the lack of a construct or a

representative pole) means that it does exist or apply in the problem space. All other ratings (indicating both or neither construct pole applies) are ignored. Each implication is stored as a compound linked list. This is known as the "implication list." A list node exists for each implication conclusion or element. In addition, each compound condition is comprised of a linked list of traits or construct labels. This list is known as the "if list." See figures 5.13 and 5.14 for details of the list structures.

GENERATOR is the function which takes the available implications and transforms them into a usable format for a chosen KBS shell or tool. The function will execute if no implications are currently available. However, in this case, only non-rule requirements are met. A choice is given for the target format. Several commercially available tools are presented in an alphabetical list, but only one choice -- VP-EXPERT -- is implemented. This function is implemented in such a way that any target format can be added with relative ease. In most cases, a current menu option would have to be turned on. In other cases, a new menu option would have to be added. In either case the routine required for the transformation would need to be made available.

VP-EXPERT is a rule-based tool. Therefore, transformation from the implications found in DETECTIVE to the rule format required by VP-EXPERT becomes relatively easy. In addition to rules, VP-EXPERT requires other system preparation to make the information somewhat usable. A basic "ACTIONS" block is generated to identify the system and start execution. In addition, a set of default questions and valid answers are generated using the list of constructs. The information needed to generate these "ask" and "choice" statements are found in the list of constructs. A valid choice for each question is either a YES or NO. The default questions basically ask which construct pole is applicable. See

appendix F for sample systems. Due to the current VP-EXPERT constraints, the "variable" names created by KE-KIT do not follow expected syntax. Doing so would require an added level of complexity not desired at this time.

The final Knowledge Engineering support function is REVIEWER. This function displays available information for on-line viewing. All available knowledge known to KE-KIT for the current application is made available. The user is given the option to choose various subsets of the knowledge or to display all available information. Everything is presented in a browse mode only - nothing can be altered. This function gives the user a chance to review information before deciding follow up steps. These steps may include giving more constructs, determining implications, or generating an initial knowledge-base format among other possibilities.

IMPLEMENTATION ISSUES

As previously stated, KE-KIT has been designed and implemented with expansion and enhancements in mind. Most of the information that is acquired from the user is stored in a linked list format. Every type of knowledge used in some way has its own structure (see figures 5.10 - 5.15). Each of the important list structures necessary to process a grid is defined to be a doubly linked list. However, in the current state of KE-KIT only the "forward" link is used for processing. In most cases the linked lists are processed in a "last in first out" approach. The important bit of information implemented with expansion in mind is the level number. Information from various levels can be (or it is believed) added to the same list as appropriate and

processing can take place as designed as long as the level numbers are used correctly.

A grid usually thought of as a two-dimensional array or a matrix is not implemented as such. Because of the information stored in the grid structure, a grid node is defined for each element (see figure 5.15). Within the grid node, an array of constructs and their relating ratings for that element is kept. The first time the element is rated on any construct, a grid node is created. All future ratings associated with that element cause an update to the already created node to occur. Note also that since the grid node keeps track of the level for which it applies, grids from other levels can be added to the same linked list.

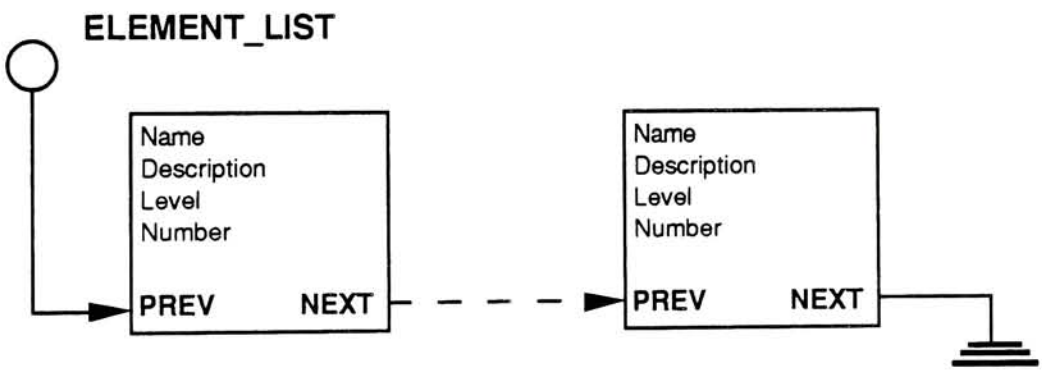


Figure 5.10

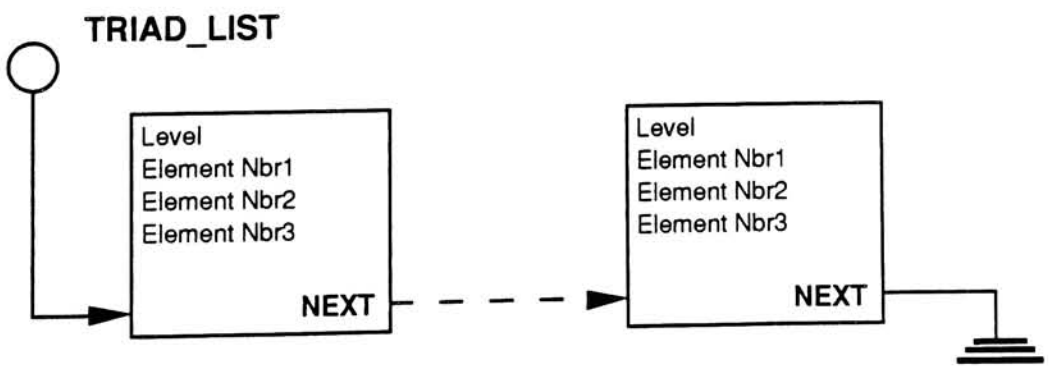


Figure 5.11

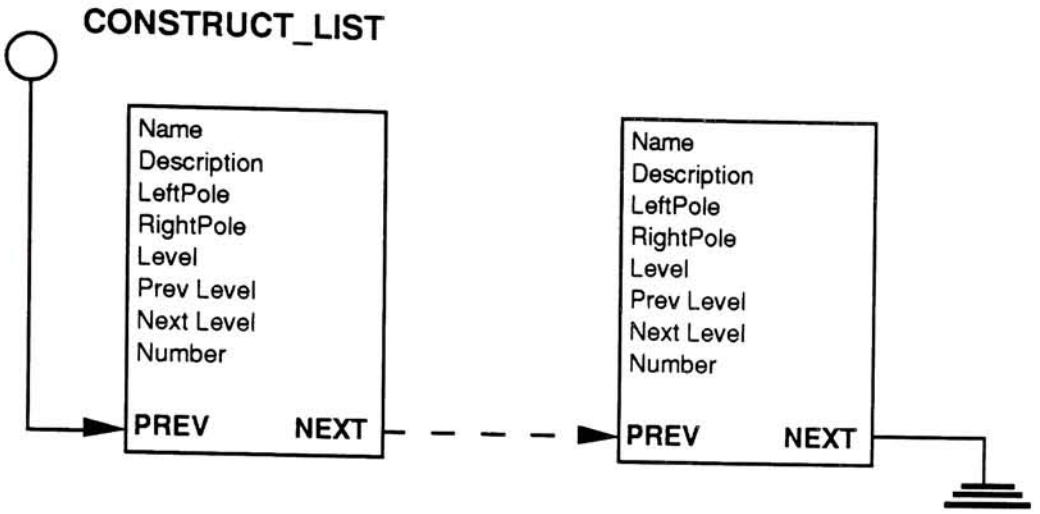


Figure 5.12

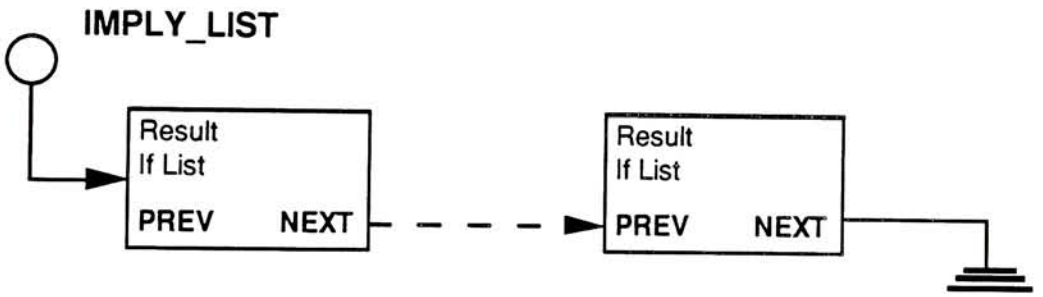


Figure 5.13

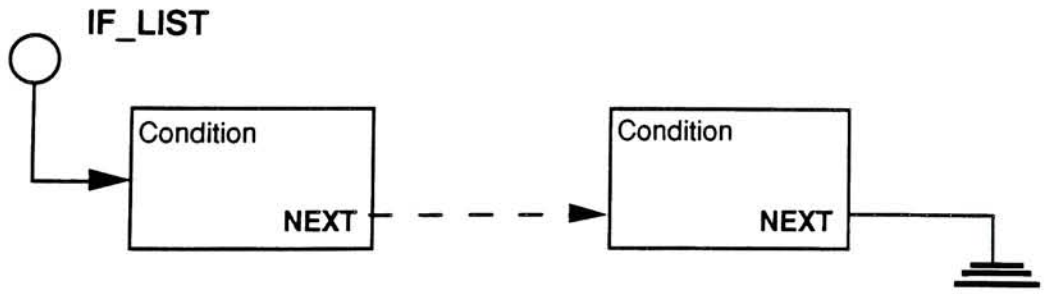


Figure 5.14

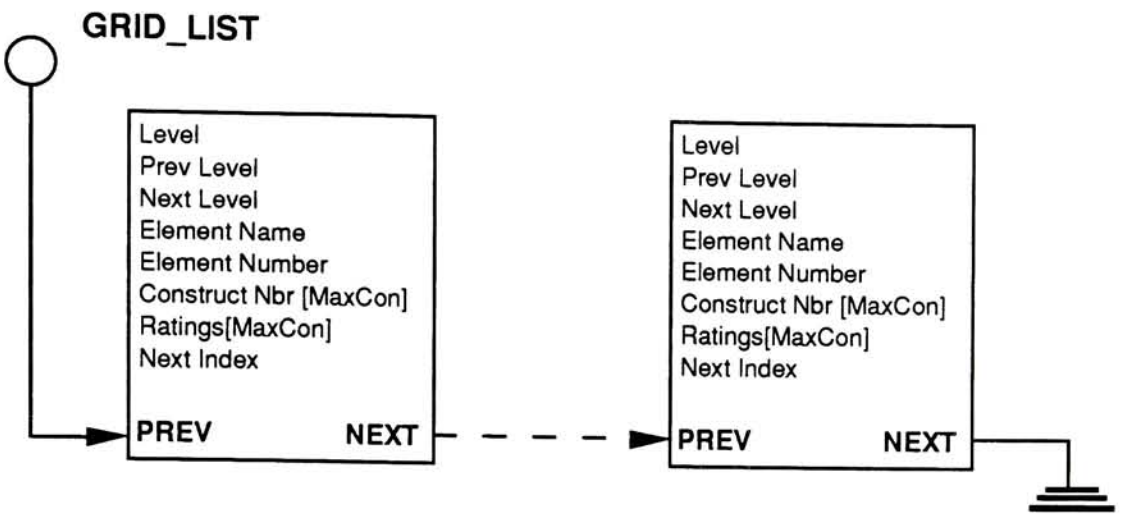


Figure 5.15

The linked list utilities needed for the current state of KE-KIT are very limited. Obviously routines are needed to create nodes and add them to the appropriate list. All nodes are added to the beginning of its respective list. Routines also have been implemented to locate particular nodes in the list. All searches initiated are based upon either a construct number or an element number. For elements and constructs, these searches are for informational retrieval only. However, if a grid node is found based on a given element number, it is usually followed by a subsequent call to a routine that will update the node. The only specific delete routine required is for triad nodes. As triads are used, they are discarded from the list. All other information is either deleted upon termination of the program or via a routine which deletes the entire list for each information source just prior to loading in a new application. Other utility routines would be required if revisions to acquired information were allowed.

Since a hierarchy of grids should be implemented to handle complex problems, most of the information is stored with an associated level number. This number is intended to represent the level to which the information applies.

However, since KE-KIT only handles single grid problems, the level number for all instances is set to one by default. Using this implementation approach would lead to inter-connected grids and related information distinguished only by the level of applicability. The implementation then becomes more complex in dealing with the grid data. The list structure would become a compound link. Not only would "grid nodes" be connected in a linear fashion, they would also be linked in a hierarchical manner at possibly every node (as a result of laddering).

In order to be able to save an application in progress and subsequently be able to load that information back into KE-KIT, a scheme had to be devised that would allow for variable formatting. The one chosen relies on header information for each file record written from or read into KE-KIT. Each record created has a one character type code followed by a colon. For example, background information has as a header "B: "; elements use a header of "E: "; constructs use "C: "; grids use "G: "; and triads use "T: ". With the exception of background information, one list node of any type is written per line to a file created when the "save" function is initiated. The background information is divided over three separate lines for ease of field identification. For all record types, each value (with the exception of the header and the first value) is separated by a delimiter. The '*' character was chosen as a delimiter since its use is unlikely and it would also allow for easy reading of the file created. This delimiter character can easily be changed during enhancements if necessary. Using this scheme, it becomes a matter a simple formatting for output and parsing using the formatting rules for input.

USE of KE-KIT

The flow of control while using KE-KIT can take many paths. A large part depends upon the experience of the user and what state the current application is in. Since the system is menu driven, much of the execution is done in a loop fashion. A couple of possible execution paths are described below in figures 1.16 and 5.17. Keep in mind that these just represent two scenarios that may be common. Others are possible.

A user of the system has several options once reaching the main KE-KIT menu. The first time user may want to learn a little about KE-KIT by viewing the overview, help, and dictionary facilities. This will give the user a quick overview or the system and possibly an idea on how to proceed. Next the user should evaluate the proposed application with EVALUATOR. This is an optional step and may only be used by those unfamiliar with knowledge based systems and thus not sure of the applicability or if enough justification exists. However, the use of this function is highly recommended. After the problem domain evaluation, the user may decide to terminate the program or continue based upon the recommendation presented.

If the user decides to continue, the next step should be to provide some basic background data about the application to be developed. In order to do this, the HISTORIAN function will be executed. The following steps should start the knowledge acquisition process. Using BUILDER, the user will give information about problem solving in an iterative fashion. During this process a rating grid is built. The elements given will represent the conclusions or recommendations determined as a result of problem solving.

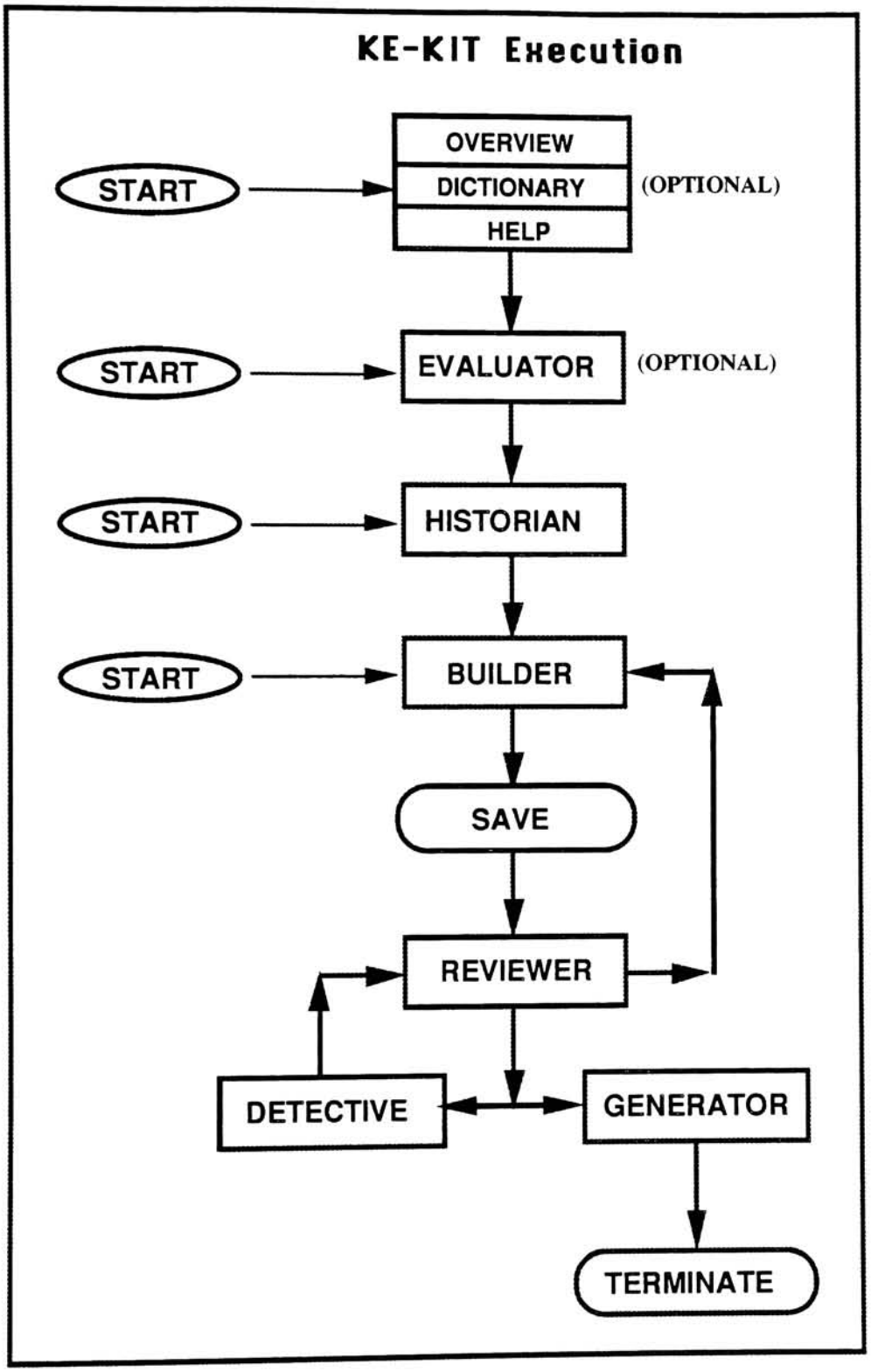

KE-KIT Execution
START
OVERVIEW
DICTIONARY
HELP
(OPTIONAL)
START
EVALUATOR
(OPTIONAL)
START
HISTORIAN
START
BUILDER
SAVE
REVIEWER
DETECTIVE
GENERATOR
TERMINATE

Figure 5.16

On the other hand, the constructs given will represent problem traits, characteristics, or attributes used to distinguish one element from another. At some point the user or the system (system limits have been reached) will determine that enough information has been given. At this point the knowledge acquisition process has concluded as far as KE-KIT is concerned.

The user now has a choice of what to do next. The recommended action would be to save the domain knowledge just given. This is done with the SAVE file utility. Next the DETECTIVE function can be invoked to determine the implications that exist based on the current set of information. Once this function terminates, the user should review the knowledge and other derived information using the REVIEWER routine. If the system limitation of constructs have not been reached and the information dows not appear to be complete, the user will have an opportunity to add more. When satisfied, the user can then initiate the GENERATOR routine and transform the current knowledge set into a VP-EXPERT format.

For systems in progress (as opposed to first time use), the first action will be to determine the appropriate file name and load the information using the appropriate file utility. All available information will be loaded into memory using its defined structure. The user should then start by reviewing the information available using REVIEWER. The next step would depend upon the current state of the application. Additional information can be added, background information could be given, implications could be built, or the target knowledge base format could be created.

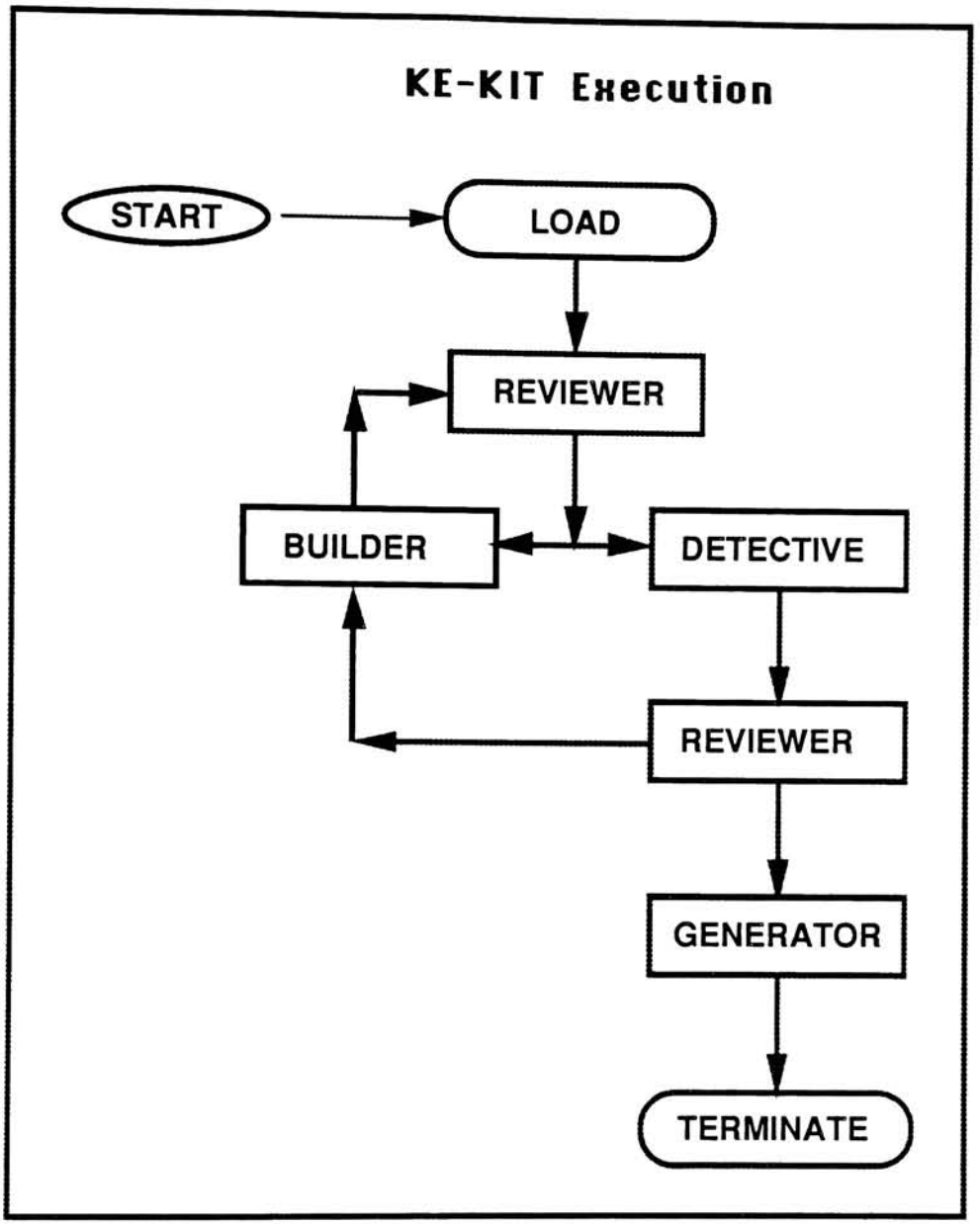


Figure 5.17

TESTING

During the development of KE-KIT, one main testing approach was used for the individual functions. Many of the

individual routines were developed and tested separately until a point of satisfaction was reached. At a point where several related functions (eg. BUILDER) were deemed to be completed, a sort of integration testing was performed. This development and testing phase continued in parallel with the initial development of the menu system. Many functions linked to the menu system executed a NULL routine until the function itself was sufficiently developed and tested. Once linking a routine to the menu, it still could be separated to perform individual testing as needed. With the menu and needed routines in place, the entire system could be put to the test.

Most of the testing of KE-KIT involved the knowledge acquisition routines. During the major development, little emphasis was placed upon the actual data used. The major concern was that the process work correctly with the data used. As many situations that could be thought of, were tested to various degrees. Testing KE-KIT with a real application in mind proved to be very beneficial in putting the "grid approach" and the implementation to the test. Several applications were tried. In addition, a couple of people not familiar with the system and the methodology used were asked to try the system.

Several applications were tested and attempted. Some of these were developed with off-the-cuff data; others were developed using examples from various knowledge-base system tool manuals and other such documentation. These applications included various sport classifications, animal classification, an investment advisor, and a trip advisor (see Appendix F for results). Even though the target user for KE-KIT is a domain expert, no testing took place which involved such a person. All of the application testing was done using simple classification style "problems." The main objective of testing was to demonstrate the use of the

technique and not necessarily to research the user perspective in detail. Due to many of the current limitations within KE-KIT, it was felt that using a domain expert to help with testing would not be very valuable.

One of the sport classification applications, based on an example taken from chapter 3 (see figures 3.7-3.9), is given later in this section. Figures 5.18 and 5.19 illustrate the repertory grid produced from collecting the initial six elements and four constructs. Once these were collected, a request was made to detect and generate a list of implications. These are listed using the format of REVIEWER in figure 5.20. It should not be hard to see that there is missing, inconsistent, and ambiguous knowledge within these implications. Following the implications is a sample VP-EXPERT program (figure 5.21) created using GENERATOR. Finally, after all of this is the saved knowledge (figure 5.22) that resulted after saving the application for future use. Notice the line headers to signify the type of knowledge. See figure 5.23 for the key to the saved knowledge format. Also, note that the elements and constructs are listed in reverse order (6 to 1, rather than 1 to 6). This results from the way the linked list structure is processed. This does not cause a problem with processing the knowledge since the correct relationships are always stored in the grid records.

REPERTORY GRID

CONSTRUCTS		ELEMENTS						CONSTRUCTS
EMERGENT POLE		1	2	3	4	5	6	IMPLICIT POLE
C1	YEAR-ROUND ACTIVITY	1	0	0	0	0	1	SEASONAL ACTIVITY
C2	INDIVIDUAL ACTIVITY	1	0	0	0	1	1	TEAM ACTIVITY
C3	NON-CONTACT ACTIVITY	1	N	0	1	1	1	CONTACT ACTIVITY
C4	OUTSIDE ACTIVITY	B	0	1	1	1	1	INSIDE ACTIVITY
		SWIMMING	BASKETBALL	FOOTBALL	BASEBALL	BIKING	JOGGING	

Figure 5.18

WORKING RATING GRID

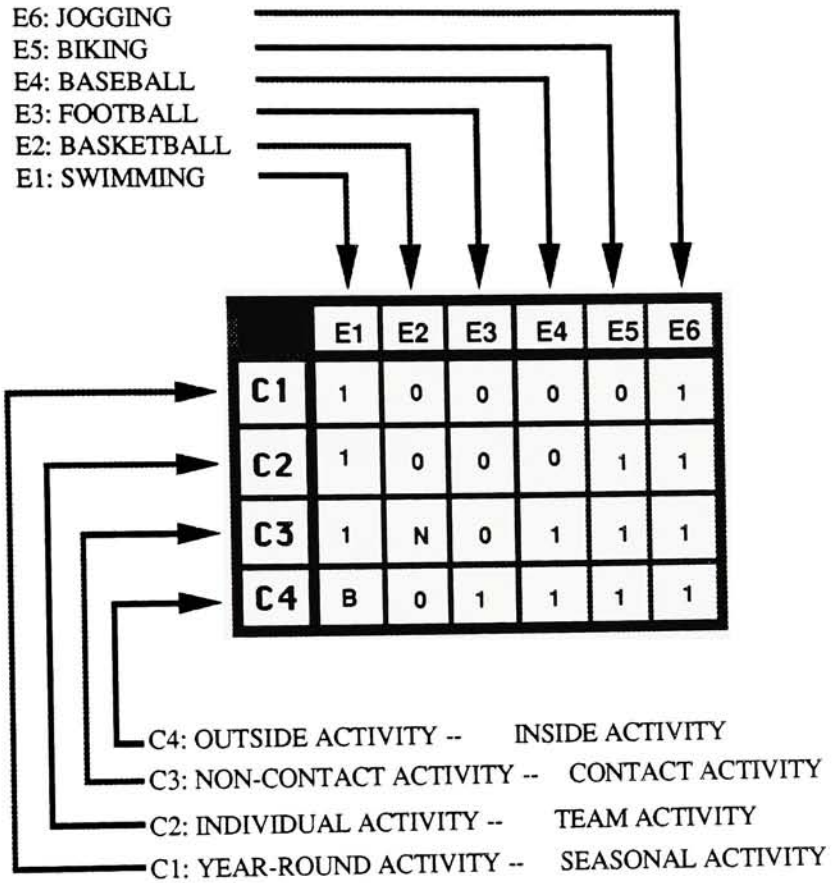


Figure 5.19

SPORT ACTIVITY APPLICATION IMPLICATIONS

Application Title: activity

IMPLICATIONS DERIVED

>> IF non-contact activity
AND individual activity
AND year-round activity
THEN swimming

```
>>  IF     outside activity
       AND non-contact activity
       AND individual activity
       AND seasonal activity
    THEN  biking

>>  IF     outside activity
       AND non-contact activity
       AND individual activity
       AND year-round activity
    THEN  jogging

>>  IF     outside activity
       AND non-contact activity
       AND team activity
       AND seasonal activity
    THEN  baseball

>>  IF     outside activity
       AND contact activity
       AND team activity
       AND seasonal activity
    THEN  football

>>  IF     inside activity
       AND team activity
       AND seasonal activity
    THEN  basketball
```

Figure 5.20

SPORT ACTIVITY APPLICATION
KE-KIT CREATED VP-EXPERT FILE

```
ACTIONS

    DISPLAY "Welcome to the  activity system.
        Press any key to begin. ~"

FIND ANSWER
END;
```

AUTOQUERY

```
ASK outside activity: "outside activity - Does this apply to the problem?"
ASK inside activity: "inside activity - Does this apply to the problem?"
ASK non-contact activity: "non-contact activity - Does this apply to the problem?"
ASK contact activity: "contact activity - Does this apply to the problem?"
ASK individual activity: "individual activity - Does this apply to the problem?"
ASK team activity: "team activity - Does this apply to the problem?"
ASK year-round activity: "year-round activity - Does this apply to the problem?"
ASK seasonal activity: "seasonal activity - Does this apply to the problem?"

CHOICES outside activity,inside activity:YES, NO;
CHOICES non-contact activity,contact activity:YES, NO;
CHOICES individual activity,team activity:YES, NO;
CHOICES year-round activity,seasonal activity:YES, NO;

RULE 1
IF      non-contact activity = YES
   AND   individual activity = YES
   AND   year-round activity = YES
THEN    ANSWER  = swimming;


RULE 2
IF      outside activity = YES
   AND   non-contact activity = YES
   AND   individual activity = YES
   AND   seasonal activity = YES
THEN    ANSWER  = biking;



RULE 3
IF      outside activity = YES
   AND   non-contact activity = YES
   AND   individual activity = YES
   AND   year-round activity = YES
THEN    ANSWER  = jogging;
```

```
RULE 4
IF      outside activity = YES
   AND   non-contact activity = YES
   AND   team activity = YES
   AND   seasonal activity = YES
THEN    ANSWER = baseball;

RULE 5
IF      outside activity = YES
   AND   contact activity = YES
   AND   team activity = YES
   AND   seasonal activity = YES
THEN    ANSWER = football;

RULE 6
IF      inside activity = YES
   AND   team activity = YES
   AND   seasonal activity = YES
THEN    ANSWER = basketball;
```

Figure 5.21

SPORT ACTIVITY APPLICATION
KE-KIT SAVED KNOWLEDGE

B: activity*John S. Parsons*01/08/92*Sports-minded people*People not familiar with sport activities
B: Many people are not familiar with sport activities
B: This system tries to help categorize sport activities

E: 1*6*jogging*sport activity
E: 1*5*biking*sport activity
E: 1*4*baseball*sport activity
E: 1*3*football*sport activity
E: 1*2*basketball*sport activity
E: 1*1*swimming*sport activity

C: 1*0*2*4*place activity held*place that activity is normally held*outside activity*inside activity
C: 1*0*2*3*physical requirements*what type of physical contact is expected*non-contact activity*contact activity
C: 1*0*2*2*participation*what type of participation is usually required*individual activity*team activity
C: 1*0*2*1*time of year for sport*time of year that sport is normally held*year-round activity*seasonal activity

G: 1*0*2*basketball*2*1*0*2*0*3*n*4*0*
G: 1*0*2*football*3*1*0*2*0*3*0*4*1*
G: 1*0*2*baseball*4*1*0*2*0*3*1*4*1*
G: 1*0*2*jogging*6*1*1*2*1*3*1*4*1*
G: 1*0*2*biking*5*1*0*2*1*3*1*4*1*
G: 1*0*2*swimming*1*1*1*2*1*3*1*4*b*

T: 1*3*5*1
T: 1*3*6*2
T: 1*3*6*4
T: 1*3*1*4
T: 1*5*2*6
T: 1*6*2*4
T: 1*1*5*2
T: 1*5*3*2
T: 1*3*2*4
T: 1*3*6*5
T: 1*2*4*1
T: 1*3*6*1
T: 1*3*1*2
T: 1*4*1*5
T: 1*5*6*4
T: 1*4*6*1

Figure 5.22

KEY TO STORED DATA

*Note: Each data value is separated by a **

Background (B)

- Line 1:
  - Application Name
  - User
  - Date Created
  - Experts
  - Intended Users
- Line 2:
  - Problem Statement
- Line 3:
  - Application Purpose

Elements (E) -> one line per element

- Level Number
- Element Number
- Element Name
- Element Description

Constructs (C) -> one line per construct

- Level Number
- Previous Level
- Next Level
- Construct Number
- Construct Label/Name
- Construct Description
- Left Pole
- Right Pole

Grid Records (G) -> one line per; each for a single element (construct/rating pairs are repeated)

- Level Number
- Previous Level
- Next Level
- Element Name
- Element Number
- Construct Number
- Rating

Remaining Triads -> one line per triad

- Level Number
- Element Number 1
- Element Number 2
- Element Number 3

Figure 5.23

To better understand the process used, it would be beneficial for the reader to step through the given example here (and those in appendix F). One should start with the information in the "saved knowledge" file. A grid or matrix should be built using the grid records in concert with the elements and constructs. The key to the stored data is listed following the data. Additional information can be added using the list of unused triads. Once the grid is built, the implications can be developed using the decision-action table approach outlined in chapter three. The difference between the example outlined and the implementation is that KE-KIT uses both poles of the construct for possible conditions. Keep in mind that ratings of "1" indicate the left pole, ratings of "0" represent the right pole, and all others are ignored when building implications.

CHAPTER 6 KE-KIT REFLECTIONS

INTRODUCTION

As already mentioned, KE-KIT has been implemented as a prototype system. Several problems and benefits have been identified and proven through the implementation and testing process. Since KE-KIT is just a prototype, there is a lot of room for enhancements and improvements. The enhancements fall into several categories. They include user-interface issues, further development of the repertory grid approach to knowledge acquisition, and integration issues. KE-KIT is only the beginning of a set of tools that would at the very least assist a knowledge engineer and in an ideal situation transfer many responsibilities to a domain expert.

PROBLEMS (Drawbacks) and LIMITATIONS

KE-KIT is by no means a system without problems or limitations. Some of these are a result of the assumptions made and the defined scope. Many others are inherent within the knowledge acquisition process and are the basis for much research. These problems can be categorized as general, implementation, or methodology problems. With further work, some of these can be addressed. However, other problems may require extensive work and/or a new approach in order to resolve.

General system problems may exist with any automated knowledge acquisition tool developed. A tool developed for automated knowledge acquisition should be able to handle various levels

and structures of knowledge. This includes handling various types of problems and knowledge representations. The repertory grid technique implemented in KE-KIT, however, is geared toward a limited application range and knowledge structure. This, thus, limits the type of knowledge handled to "casual knowledge" or to the conceptual structure of the problem resolution. As a result, a robust model of the problem and its resolution is not created. Control knowledge of any kind is not acquired. Also, the application areas are limited to those areas where solutions or recommendations can be enumerated prior to investigation. These application areas include classification and diagnosis. In order to solve multiple application types, multiple knowledge acquisition techniques need to be integrated into a single system. It would also be beneficial if these techniques were compatible and complimentary so that the deficiencies of one can be compensated for by another approach. In addition, real world problems often have multiple experts or knowledge sources. An automated tool should consider these as well as many other issues in order to be widely accepted and used. KE-KIT only addresses a very small portion of the knowledge acquisition process.

The implementation limitations mostly arise due to the confined scope of the project. Issues surrounding the user-interface and the environment are the most obvious. Commands and function key processing should be added to facilitate smoother and quicker use. Implementing a means to escape from some functions and displays without fully executing them would be beneficial. Context sensitive help and improved audits would also help to make the system more user-friendly. Other factors relate to the hardware used. No provisions have been made to handle non-color displays, various printers, and so forth. The system was developed and tested in one specific environment and not made portable.

Other implementation limitations specifically relate to the scope of the knowledge acquisition method used. The current system allows and does not recognize duplicate or overlapping constructs (see Appendix F for examples). Knowledge acquisition tools for knowledge analysis, uncertainty handling, conflict resolution, and incomplete knowledge among others would be very beneficial to the final knowledge format created. These tools would also increase the applicability of the tool tremendously. In addition, a knowledge refinement process is needed that is able to perform full updates - add, change, or delete. This process may include a means for testing the knowledge. The repertory grid approach should also be expanded to include a hierarchy of related grids. This addition along with a laddering technique will allow much more, comprehensive knowledge to be acquired about any given problem area. These enhancements to the grid implementation will also lift the limitation on the number of elements and constructs that can be processed. As a result, larger and more complex problem areas can be addressed. Other enhancements should be considered along the lines of the "target" knowledge representations that can be processed. Currently only rule formats specifically for the VP-EXPERT tool are generated. Ideally various formats, including frames and objects, could be created and processed.

The limitation of fifteen constructs may not be much of a problem if a multiple grid structure is implemented. However, the requirement of the use of six elements can be a problem. In some situations, six may be too many. In others, six may be to little. In using triads, a minimum of three is necessary. Thus using the grid technique requires a minimum of three elements. Keeping this in mind, the user should be able to enter any number of elements so long as at least three are given. This seems to rule out the creation of a system which has less than three possible outcomes (ie. yes-no conclusions).

The most obvious problem relating to the method implemented relates to its use. Unless someone has some prior use or training on the approach, results may be unpredictable and not very beneficial to the problem. Care should be used in selecting the set of elements. It may be difficult so select a representative set. The ones used should sufficiently represent the problem area. The elements used and the order presented may affect what constructs are acquired. Triads chosen on a truly random basis bring out the greatest contrast. However, the elements in triads should be changed frequently to avoid the tendency to dwell on a point. Within KE-KIT, the user has complete control over when triads change. Constructs defined should be applicable and should not totally be subjective especially if they are not commonly agreeable. Such defined constructs may lead to inconsistent or improper relations which others feel are wrong or misleading. It is also difficult to determine if a sufficient set of constructs has been given. Even though the technique claims to handle it, it becomes difficult to handle non-verbal labels for constructs.

Within KE-KIT, problems arise when a triad is presented and no new or applicable information can be given. There is no bypass option. This may lead to duplicate, conflicting, or non-applicable information. The user should have a way to bypass a set of elements in this situation. An alternative would be to have a background function watch for and catch such situations and minimally eliminate the duplicate constructs. Working within KE-KIT this way, a user can become frustrated. In many instances, a user may not be presented with a set of elements that would allow certain information to be collected. This, then, leads to incomplete knowledge. In these cases a user needs the ability to offer knowledge without having to wait for an applicable triad to be presented. This then leads to knowledge acquisition outside

of the repertory grid method. However, the grid could still be used.

Given that constructs represent the conditions of implications and these conditions can be true or false, it is difficult to see how multiple choice conditions would be implemented. For example, a construct named "color" may need to represent a variety of answers. In systems where valid multiple choice questions are possible, using the grid approach may force an unfamiliar structure on the knowledge by forcing the use of boolean possibilities. It may be possible to implement constructs which represent multiple choice questions using an appropriate rating scale. However, this approach may be limited to the granularity of the defined rating scale. As a result, another entirely different acquisition approach may be needed to handle such situations.

Using various rating scales and grid analysis techniques, different results are possible. Some of these results may be more correct than others. The rating scale and "analysis" technique implemented within KE-KIT is very basic and probably would not stand up under more complex problems. In fact, the method which KE-KIT uses for assigning ratings should be changed to handle more complex problems. One problem becomes very obvious unless a user is careful. As previously stated, the left pole should represent the similarity end of the construct. However, it is very easy to loose sight of this during the thought process and assign the similarity label to the right pole instead (an example can be seen in Appendix F). Because of the way KE-KIT processes the rating grid data, this is not a problem. It would become a big problem if a laddering technique were implemented. Also, the correct usage would probably be more important if an uncertainty management system were installed.

BENEFITS

The benefits of KE-KIT can generally be grouped into two categories. The first relates to automated knowledge acquisition tools in general and the other relates to the methodology implemented. The degree of applicability of these varies greatly. This is due primarily to the scope enforced on the delivered system. With appropriate enhancements and expansion of KE-KIT these benefits can become even greater.

One of the big benefits possible is the time savings involved as compared with manual means. The required knowledge engineering process is reduced. This addresses the primary issue involved with the knowledge acquisition bottleneck. KE-KIT eliminates several transformation steps normally found in purely manual development. Such a tool as KE-KIT also plays a role in gaining early user and expert acceptance for the project and proposed application. This is crucial for those that are initially skeptical. The system can be used to teach people in the domain area concepts about the technology used as well as about the problem being addressed. Along the same lines, such a system has the benefit of presenting expertise in an intermediate form (implications) - one step between the expert and a knowledge base format. In KE-KIT this benefit can be seen by using the review and print functions.

The methodology used in KE-KIT offers several other benefits. The approach removes any interviewer bias and thus the acquired knowledge becomes more objective. However, the possibility still remains for the user to add subjectivity. The repertory grid approach introduces a structure that can lead to high productivity. In essence a form of structured interview is being used. This approach forces the user to define concepts and relations found in solving the problem at hand. It provides a good means for goal (solution) decomposition. The elicitation process also has the side-

affect of defining a domain vocabulary to some degree or another. This greatly helps the understanding of any system developed. The rating scale approach used lends itself to the implementation of various uncertainty handling techniques. If necessary, the scale can be redefined with relative ease. In general, the technique forces the user to think about the concepts and discriminate them possibly in ways that were not really thought of before. With all things considered, the quality of knowledge acquired should be improved over manual means. Also, it is relatively easy to look at a grid record at some later point and understand its original meeting. This is not the case with many other techniques used for knowledge acquisition.

If nothing else, construct elicitation may be used as a focus for discussion in a manual nature. The grid used may be looked on as a particular form of structured interview. Repertory grids are based on a technique which appears to be quite simple. They are also very easy to modify and adapt. Using the grid process has the ability to clarify data in an easy quantifiable form while capturing distinctions among closely related concepts. The grid is only a technique and is limited only by the user's imagination. It can be adapted to many forms for the desired needs at hand. The way that a grid records interview data is a great advantage. A Grid record is full of information. A line of thought can almost always be triggered by looking at the record. Also, several people can work on the same problem and all can understand the interview records without much explanation. Distinctions captured in grids can be converted to other representations such as production rules, fuzzy sets, networks, or frames. Large problems can be broken into pieces and represented in multiple grids and arranged in hierarchies. The hierarchies can be organized around cases, knowledge sources (ie. experts), solutions, and/or traits.

LESSONS LEARNED

Several things were learned by implementing KE-KIT. The biggest one was a reinforcement of prior knowledge. Automating the knowledge acquisition process is no easy task. Many issues are involved and must be considered before designing or implementing a system. In addition, the implementation itself is not trivial. Key issues involve the user interface and the intermediate knowledge representation used as well as the knowledge acquisition process itself. In order to fully implement a production or commercial quality system, one needs to understand many techniques, approaches, and disciplines. These include computer science, artificial intelligence, software engineering, statistics, and psychology among other things. In most cases this may mean having a team of people design and implement the system.

The use of "construct elicitation" and the use of the grid technique has some promise. It has already been accepted as a technique for knowledge acquisition. This can be seen by the number of research tools using the method and also by the influence on the few commercially available knowledge acquisition tools. There is no one correct format for grids nor only one way to use them. Compared to other methods, this is an added advantage. This flexibility can allow the user to perform various analysis and possibly gather different forms and detail of knowledge. The approach is easy to implement. In addition it appears to be relatively easy to implement various uncertainty handling techniques. For example, it is not hard to see that ratings can be used to incorporate fuzzy logic concepts. However, the basic grid can not be used alone. Other knowledge acquisition methods should be integrated. In addition, proper knowledge analysis techniques must be used in order to make the tool useful.

RESEARCH AREAS

There are several areas that can be researched related to KE-KIT and knowledge acquisition. These areas can be classified as those relating directly to enhancements for KE-KIT and those dealing with knowledge acquisition in general. Much activity is currently taking place around the world for many of these areas. This is true because of the increasing demand for production quality knowledge systems, the lack of qualified knowledge engineers, and the ever present knowledge bottleneck. Based on this statement alone it should be obvious that automated tools are needed that can adequately handle a wide range of problems.

Areas of research that involve KE-KIT are varied. The implementation of the repertory grid technique in its current state is very basic and simplistic. Work can be done to introduce laddering and thus create a hierarchy of grids. A set of functions is needed to analyze the knowledge and thus handle inconsistent, missing, ambiguous, or conflicting information. This analysis function is critical to the future applicability of such a tool. Once this is done, it should be rather easy to implement one or more uncertainty handling mechanisms. Uncertainty can be handled at two levels. It can be done at the construct level (much the same as certainty applied to rules) and at the rating level (certainty applied to data). Another set of functions can be implemented to address the validation and revision issues. This may also involve testing functions and thus require the implementation of one or more knowledge control strategies. Other areas that can be investigated include rule or implication optimization, the use of multiple knowledge sources, the addition of explanation features, and the maintenance of existing knowledge bases.

One important enhancement does not directly involve the knowledge acquisition process. Knowledge representation improvements can be very valuable. This would also include providing for various other formats. These may include other rule formats, frames, or objects to be used in many other knowledge-base system shells. With knowledge representation standardization work underway in the research community, it might prove valuable to enhance KE-KIT using the proposed standards. This will allow KE-KIT to interface with many system shells in the near future.

Other major work on KE-KIT may not involve artificial intelligence at all. The current user-interface could use some improvements. An ideal re-design may involve an object-oriented àpproach. This is just one idea that may involve other computer science disciplines in building and enhancing an artificial intelligence tool. Another may involve using hypermedia as a form of help or as an explanation facility that will work with the acquired knowledge.

Automated knowledge acquisition tools once sufficiently developed, should also work with other tools. Some of these other tools have been mentioned at various points throughout this document. Adding a speech front-end would ease the user-interface burden and also make the tool available to a wider audience. A natural language and/or text understanding function would be able to do much of the same. Both would cause a wider range of applications and knowledge to be accessible. Where text or other non-sequentially processing of information is involved, hypermedia may play an important role. Neural networks and other machine learning approaches are also worth looking at in regards to knowledge acquisition. Integration is also an important issue to pursue. Being able to integrate with tools for traditional software or with traditional applications is becoming more important every day.

By now, one should be able to think of several other areas of study that can relate to the knowledge acquisition process. In the last few paragraphs, several study areas and research tasks have been identified. Most of these can represent individual projects or in some cases they can be combined. These are by no means a complete list. It is just a sample to show that many activities are required and needed to build a well rounded tool for handling such a complex task as knowledge acquisition. Many people are spending researching and experimenting with all of these areas. However, there appears to be no revolutionary discovery at this point. This can be said because the knowledge acquisition bottleneck still exists. In fact, it probably will be here for some time.

REFLECTIONS

This project is thought to have been very valuable. If nothing else, it has given a good insight into what is required to implement a knowledge acquisition tool. However, it is felt that the exposure to various other knowledge acquisition methods (in the research process) is also highly valuable. A knowledge engineer in the workplace must be able to use a variety of techniques and approaches in order to successfully elicit domain knowledge from an expert. Knowledge of these techniques, however, is not enough. Experience with them is important in order to use them effectively. One must know when and where to use them. Also, there is no one right way to elicit knowledge. Various approaches may seem to be attacking a problem from different angles, but they may result in the same knowledge.

There is a definite need for automated knowledge acquisition tools. The lack of well trained knowledge engineers, the

increasing demand for knowledge systems, and the ever present knowledge bottleneck are enough justification. However, such tools will not solve all of the existing problems. The need for knowledge engineers will not be completely eliminated, but the roles will be greatly changed if automated tools become wide spread. Knowledge engineers will still be needed for more complicated activities. The possibility exists for a "knowledge system revolution." More power will be into the hands of those unskilled in artificial intelligence. Experts using these tools may be able to show improved efficiency along with time savings. A greater level of acceptance for intelligent systems should result and thus increase the demand even more. These tools can make development much easier and give the expert a sense of ownership and thus greater motivation to capture existing knowledge into a computer system for others to use.

In order for this "revolution" to occur, much work needs to be done in order to define and establish standard approaches and techniques to knowledge acquisition and knowledge representation. Activities must take place similar to those which occurred during the "structured revolution" that introduced and won acceptance for improvements within the traditional software community. Commonly accepted approaches techniques must be introduced or current ones must be revised. In addition, knowledge representation schemes must be standardized. Without these common practices and standards, it will become very difficult to address a wide range of problems effectively and also interact with various other tools. It seems to be common practice now for each tool to use a different representation scheme. Having multiple schemes available so that knowledge can be looked at with several view points for analysis is good in many respects. However, hundreds of different representations only confuse and complicate the situation. It should be noted some of this

work is already in progress (eg. IMKA - Initiative for Managing Knowledge Assets).

The use of intermediate representations help clarify knowledge and it often makes important things explicit. Proper representations can expose constraints, make the knowledge complete and concise, and also make it easier to handle knowledge data while suppressing rarely used detail. Diagrams used to represent knowledge make it easier for an expert to verify. Decision tables, decision trees, various grids, flowcharts, and various classification hierarchies may also be used as representation schemes. However, unless the representation scheme is easy to understand and use as well as expressive, it may not be very useful. This is especially true if someone unfamiliar with the scheme is using it.

Any useful tool developed for knowledge acquisition should have a mechanism for validation and verification of the knowledge. A user must be able to verify that the information entered is represented correctly and is valid. A clearly defined and understood scope is important. Other issues also need to be addressed along with this process. Such things as incomplete or inconsistent knowledge and uncertainty (see appendix H) must be identified and addressed in some way. Naturally, functions such as testing and knowledge revision should be included as well. A tool lacking these functions can not be expected to assume many responsibilities of a knowledge engineer. At best, such a tool will be an aid.

A key to any software tool is its ability to integrate with other software tools or applications. It is not out of the ordinary then that an automated knowledge acquisition tool be expected to work with other AI-based tools as well as traditional software tools and applications. Having such a capability adds to the power and benefits that can be realized. Development steps can be reduced and thus save

time, money, and possible implementation frustrations. For example, having a knowledge acquisition tool work with a knowledge-base tool can save the time of transforming the knowledge representation of one tool to the representation of the other tool. Possible inaccurate transformations can also be avoided. Other benefits can be realized by integrating with such tools as speech recognition systems, natural language processing tools, hypermedia tools, various database managers, and other traditional tools. Hypermedia has many capabilities that are beneficial. Besides the capability to provide improved help and explanation functions, it can also be used for knowledge acquisition. Hypermedia may also play an important role in representation of acquired knowledge.

For the most part, software engineering for knowledge-based systems development has developed outside the framework of most existing software engineering methodologies. The "artificial intelligence" field has been considered different enough as to warrant its own approaches. Though there are differences, many activities are nearly the same. Many past problems in software engineering are now being addressed in knowledge engineering. Activities occurring with CASE (Computer-Aided Software Engineering) tools are also being dealt separately within artificial intelligence. For example, the emergence of automated knowledge acquisition tools (which also code rules or other objects) parallel the emergence in the last few years of mainstream code generators and other CASE tools. Several other similarities can be found between the two development approaches. It would probably benefit both if the development and methodology paths converged and worked toward common goals. After all, most developed systems are composed of traditional systems with various smaller AI components.

The reason for the differences between AI and traditional approaches may have stemmed from the history of artificial

intelligence and its introduction to the market place. Intelligent systems have always been touted as completely different than traditional systems. To some degree, this is true. However, it may only be true in the way a final product is developed and delivered. Many of the development steps needed are very similar (ie. feasibility, requirements, initial design, implementation, testing, maintenance). As a result of the lack of sharing, some methods used within the AI field are considered rather immature compared to software engineering methods. AI methods can learn a lot from traditional systems techniques and methods. There needs to be more activities done so that these two areas work more closely together.

CONCLUSION

The work presented here has covered a lot of ground. A quick look at the knowledge engineering activities and the knowledge acquisition process identified some problem areas. This introduction also looked at some manual techniques for handling the process. Following this introduction, an in depth look at Personal Construct Theory and the associated repertory grid was made available. Several existing tools using this theory as well as other techniques were briefly looked at. All of this was background data for the design and implementation of the prototype KE-KIT. In addition, some areas of future work were briefly outlined. It is hoped that the information presented here will prove valuable to someone in there future studies.

APPENDIX A

ROLE CONSTRUCT REPERTORY TEST (REP TEST)

Original Role Construct Repertory Test (Rep Test) proposed by George Kelly [Kelly55].

See the following sample grid and overlay sheet

Instructions: 12-14-53

This test comprises three parts: (1) CONCEPTUAL GRID, (2) CONCEPTUAL GRID OVERLAY SHEET, and (3) this set of INSTRUCTIONS. The test is designed to help the examiner to understand you and some of the people who have played a part in your life.

1. Start with the OVERLAY SHEET. Beginning with your own name, write the first names of the persons described. Write their names in the blanks provided. If you cannot remember a person's name, write his last name or something about him which will clearly bring to your mind the person's identity. You may keep this OVERLAY SHEET. The examiner will be interested only in what you write on the GRID.

2. Next, lay the OVERLAY SHEET sidewise across the top of the GRID so that the numbered blanks correspond to the numbered columns in the GRID. Note that the letters "M" and "F" appear at the heads of columns 10 to 22 inclusive. If the person whose name appears at the top of column 10 is a man, encircle the "M"; if it is a woman, encircle the "F." Do the same in the remaining columns.

3. Now move the OVERLAY SHEET down on the GRID until it is just above the first row of squares. Note that the three squares at the extreme right have circles in them. This means that you are first to consider the three people whose names appear on your OVERLAY SHEET in the last three columns - columns 20, 21, and 22. Think about these three people. Are two of them alike in some important way that distinguishes them from the third person? Keep thinking about them until you remember the important way in which two of them are alike and which sets them off from the third person.

When you have decided which two it is, and the important way in which they are alike, put an "X" in the two circles corresponding to the two who are alike. Do not put any mark in the third circle. Now write in the blank under "Construct" the word or short phrase that tells how these two are alike. Next write in the blank under "Contrast" what you consider to be the opposite of this characteristic.

4. Now consider each of the other nineteen persons whose names appear at the heads of columns 1 to 19. In addition to the persons whom you have marked with an "X," which ones also have this important characteristic? Put a check mark - not an "X" - under the name of each other person who has this important characteristic.

5. Now slide the OVERLAY SHEET down to the second row. Think about persons number 17, 18, and 19 - the three who have circles under their names. In what important way are two of these distinguished from the third? Put "X's" in the circles to show which two are alike. Write the "Construct" and the "Contrast" in the blanks at the right just as you did before. Then consider the other sixteen persons. Check the ones who also have the characteristic you have noted.

6. Complete the test in the way you have done the first two rows. Write your name and the date on the TEST SHEET and give it to the examiner. You may keep or destroy the other two sheets.

REPERTORY GRID

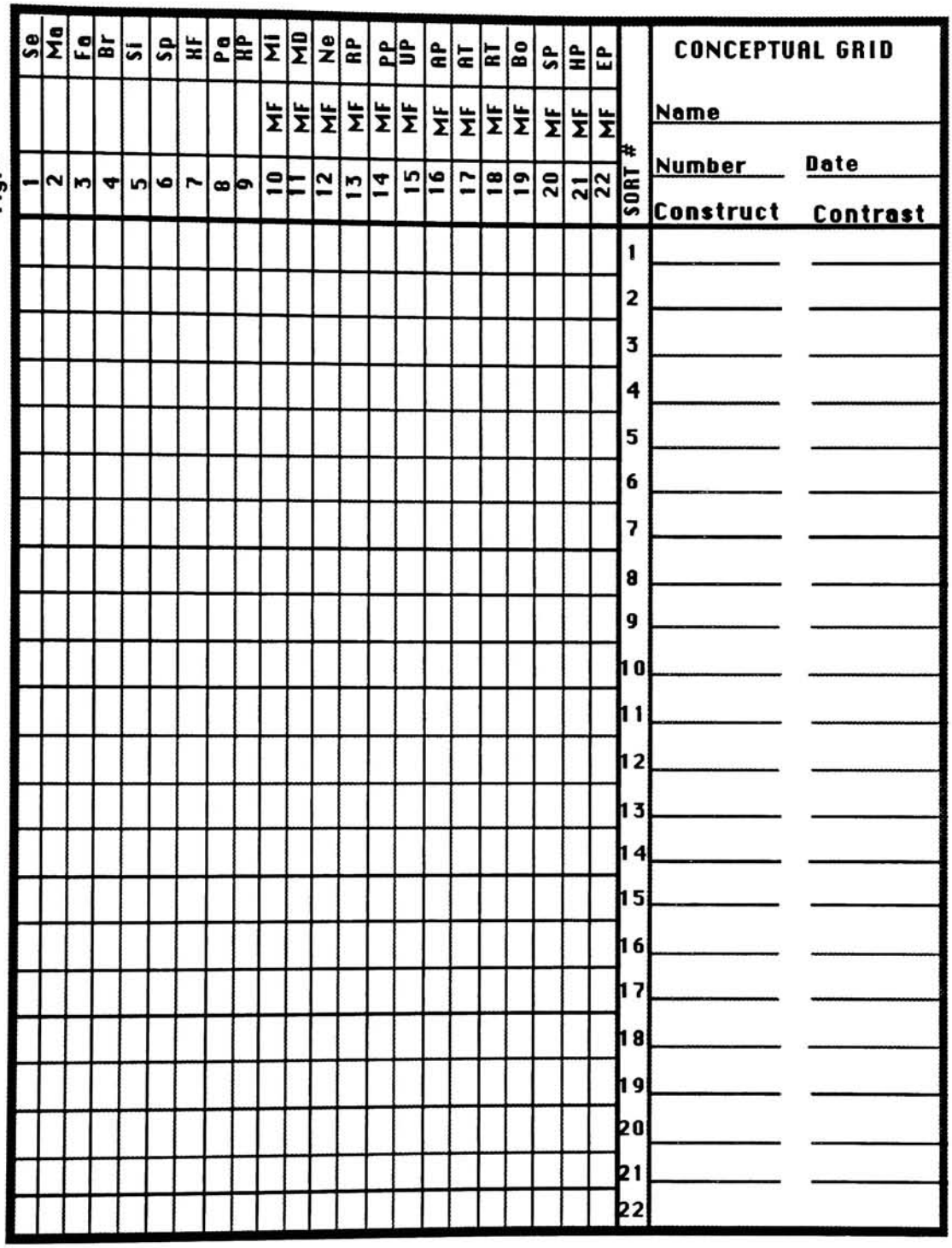


Figure A.1

CONCEPTUAL GRID and OVERLAY SHEET

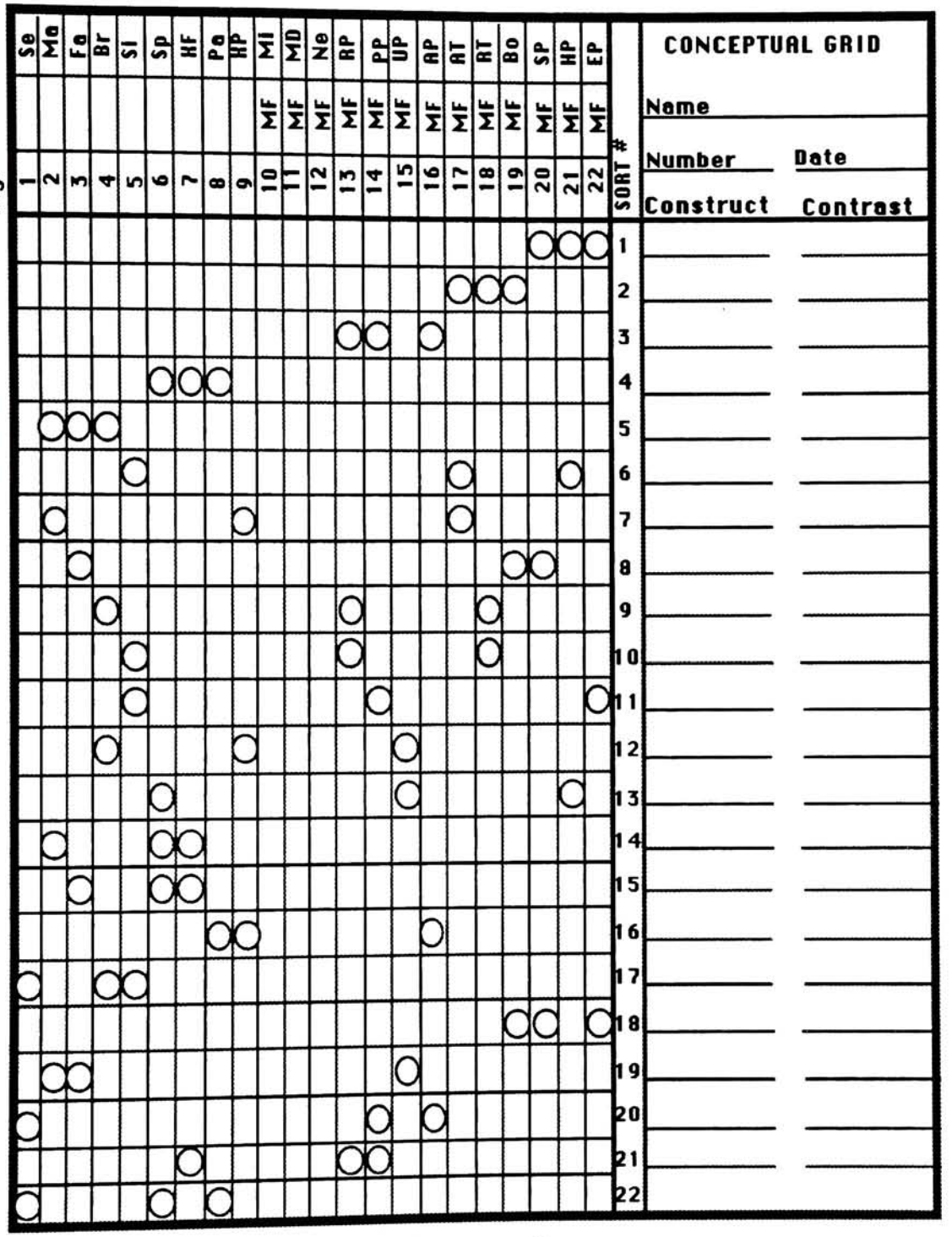


Figure A.2

CONCEPTUAL GRID AND OVERLAY SHEET - FIGURE LIST

1. Write your name in the first blank here
2. Write your mother's first name here. If you grew up with a stepmother, write her name instead.
3. Write your father's first name here. If you grew up with a stepfather, write his name instead.
4. Write the name of your brother who is nearest your own age. If you had no brother, write the name of a boy near your age who was most like a brother during your early teens.
5. Write the name of your sister who is nearest your own age. If you had no sister, writethe name of a girl near your own age who was most like a sister to you during your early teens.

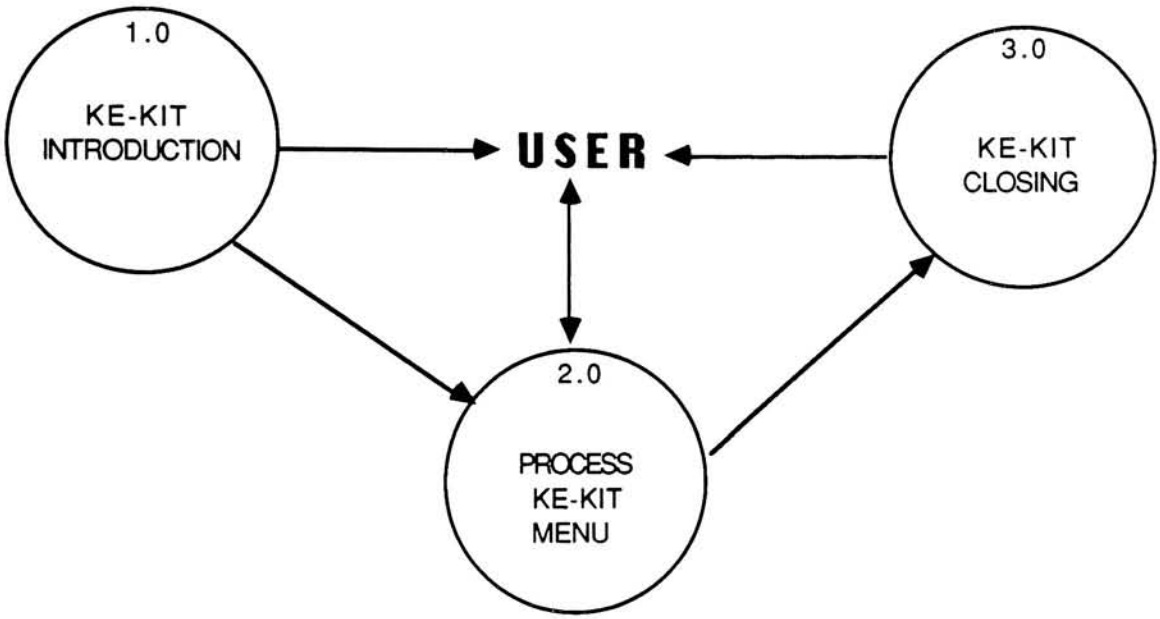
FROM THIS POINT ON, DO NOT REPEAT ANY NAMES. IF A PERSON HAS ALREADY BEEN LISTED, SIMPLY MAKE A SECOND CHOICE

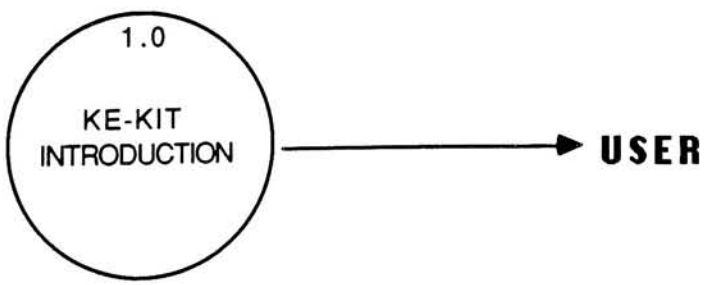
6. Your wife (or husband) or, if you are not married, your closest present girl (boy) friend.
7. Your closest girl (boy) friend immediately preceding the person mentioned above.
8. Your closest present friend of the same sex as yourself.
9. A person of the same sex as yourself whom you once thought was a close friend but in whom you were badly disappointed later.
10. The minister, priest, or rabbi with whom you would be most willing to talk over your personal feelings about religion.
11. Your physician.
12. The present neighbor you know best.
13. A person with whom you have been associated who, for some unexplained reason, appeared to dislike you.
14. A person whom you would most like to help or for whom you feel sorry.
15. A person with whom you usually feel most uncomfortable.
16. A person whom you recently met whom you would like to know better.
17. The teacher who influenced you most when you were in your teens.
18. The teacher whose point of view you have found most objectionable.
19. An employer, supervisor, or officer under whom you served during a period of great stress.
20. The most successful person whom you know personally.
21. The happiest person whom you know personally.
22. The person known to you personally who appears to meet the highest ethical standards.

Figure A.3

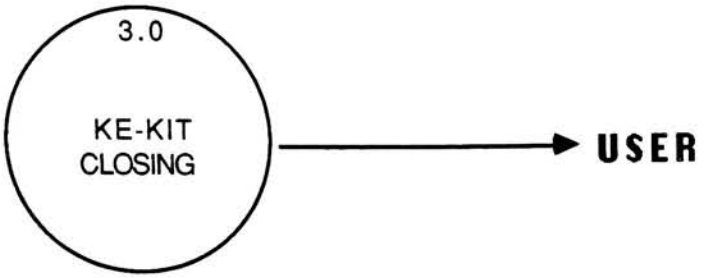
APPENDIX B

KE-KIT Data Flow Diagrams

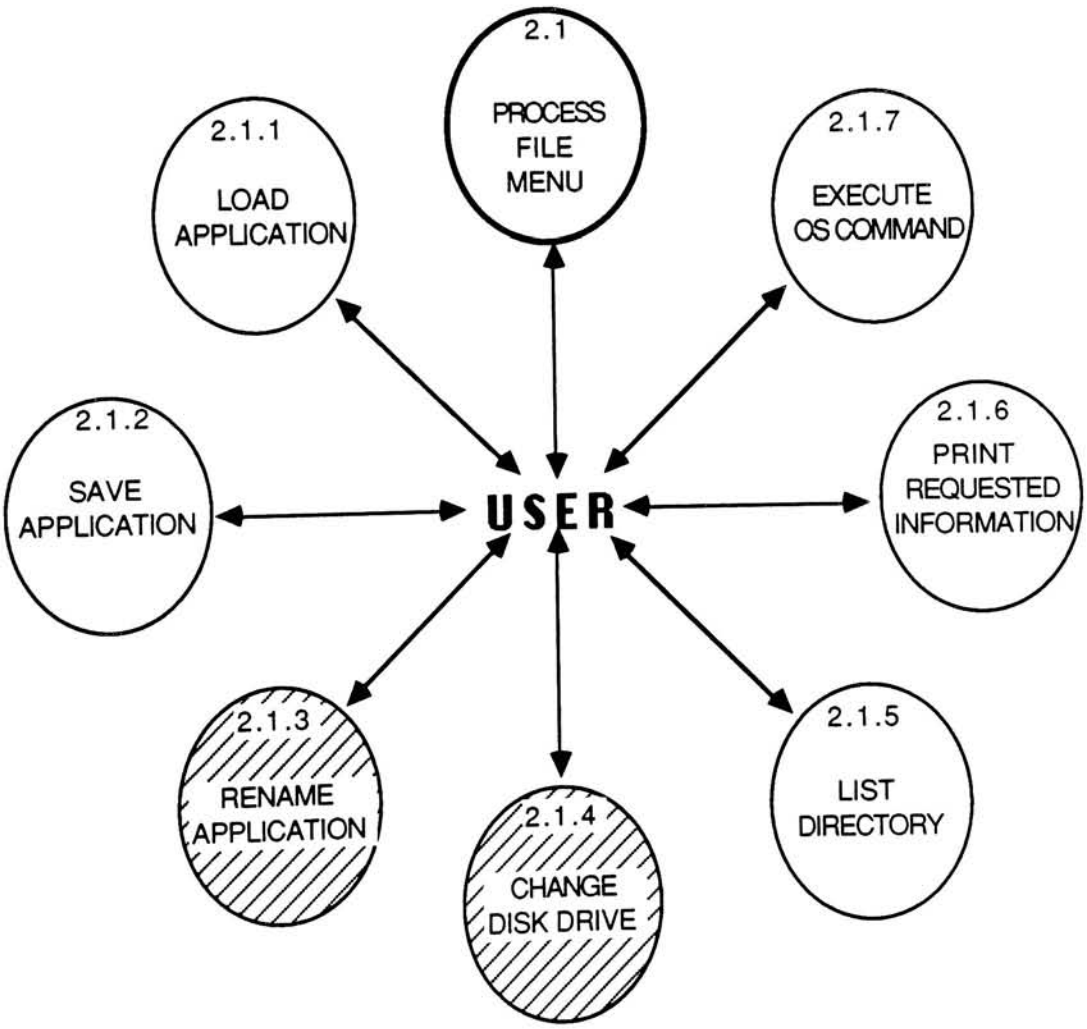


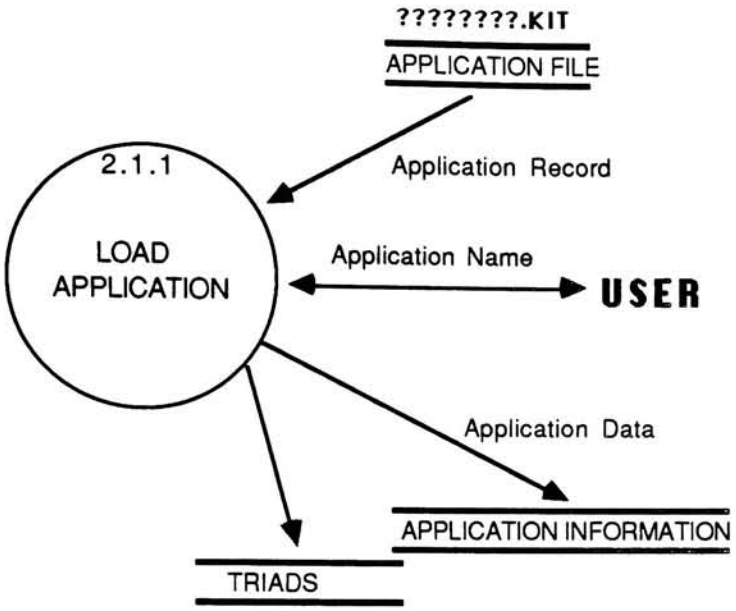


- Introduction to KE-KIT
- Initialize system

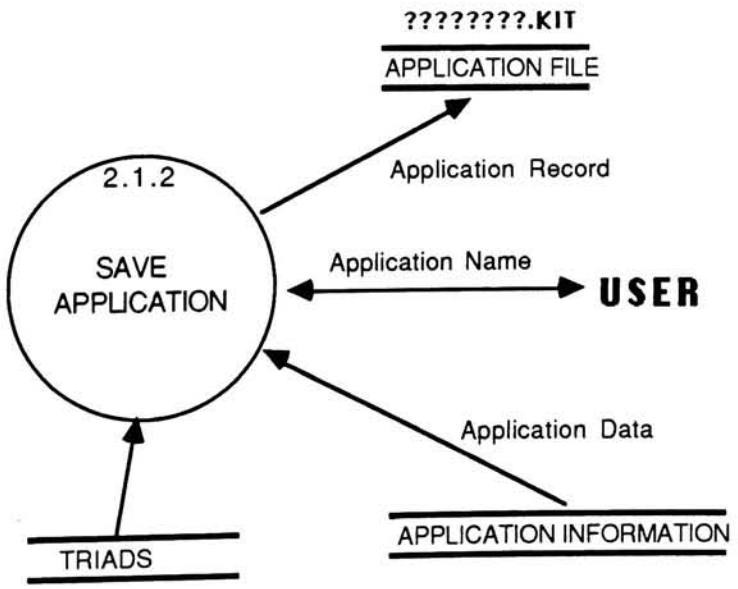


- Closing to KE-KIT
- Free resources

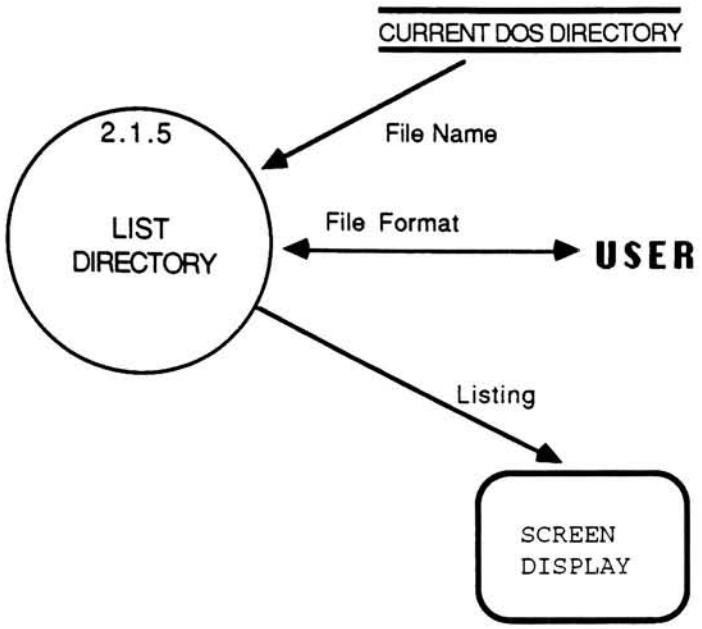




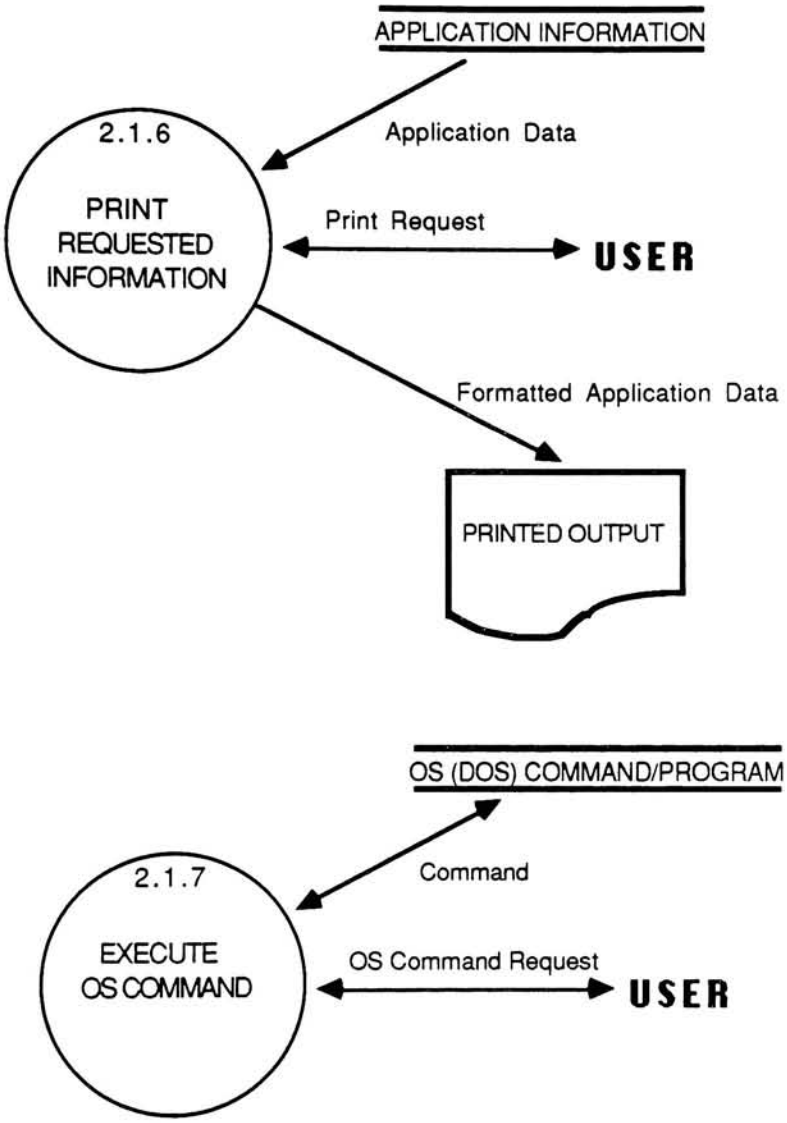
- APPLICATION INFORMATION CONSISTS OF ELEMENTS, CONSTRUCTS, RATING GRID, and IMPLICATIONS

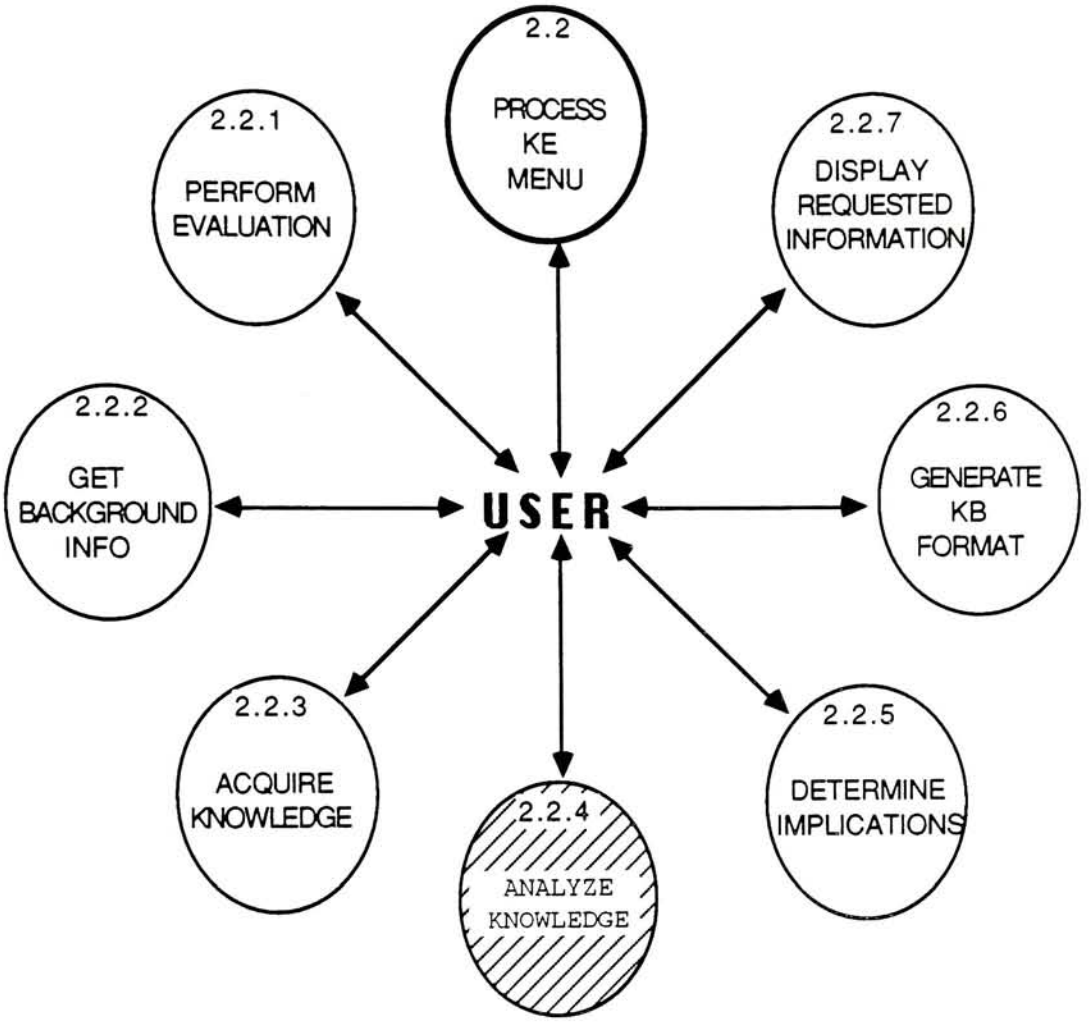


- APPLICATION INFORMATION CONSISTS OF ELEMENTS, CONSTRUCTS, RATING GRID, and IMPLICATIONS

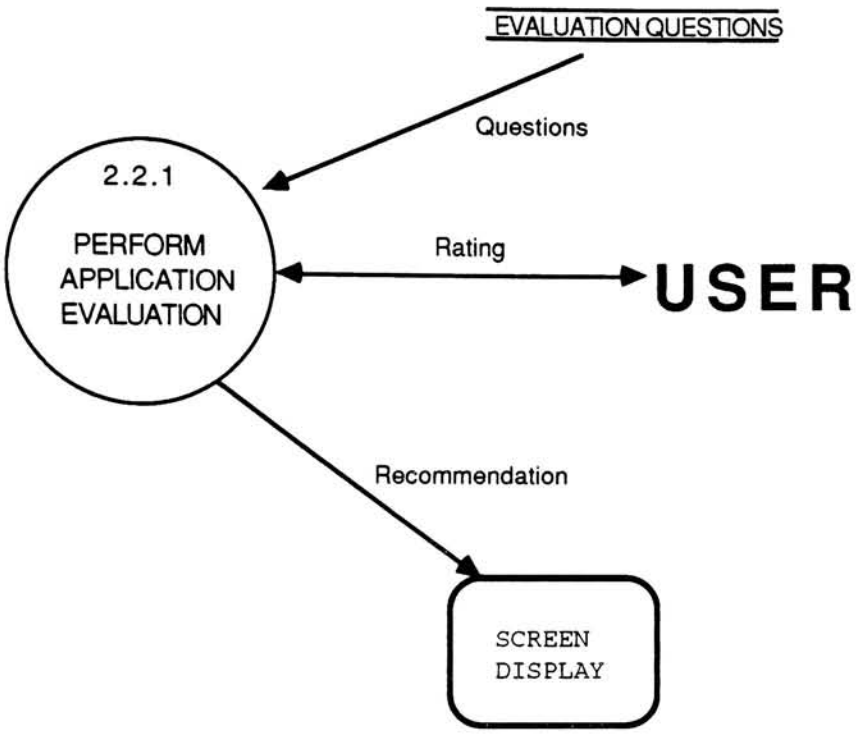

CURRENT DOS DIRECTORY
2.1.5
LIST
DIRECTORY
File Name
File Format
USER
Listing
SCREEN
DISPLAY

- APPLICATION INFORMATION CONSISTS OF ELEMENTS, CONSTRUCTS, RATING GRID, and IMPLICATIONS

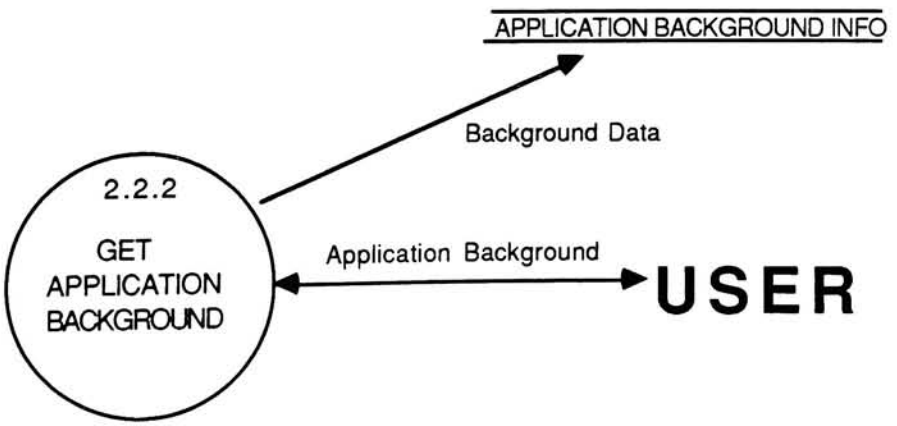


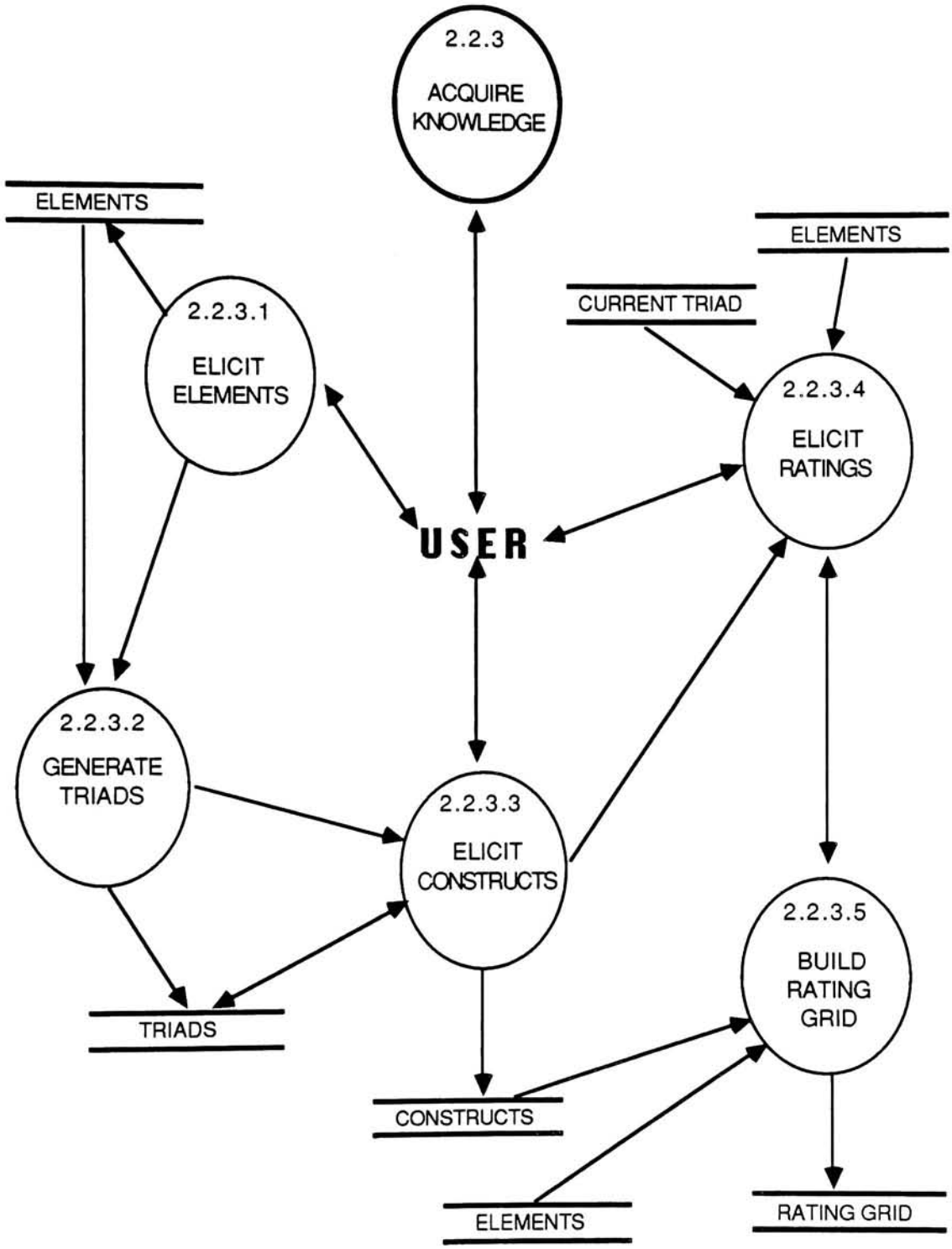

2.2
PROCESS KE MENU
2.2.1
PERFORM EVALUATION
2.2.7
DISPLAY REQUESTED INFORMATION
2.2.2
GET BACKGROUND INFO
USER
2.2.6
GENERATE KB FORMAT
2.2.3
ACQUIRE KNOWLEDGE
2.2.4
ANALYZE KNOWLEDGE
2.2.5
DETERMINE IMPLICATIONS

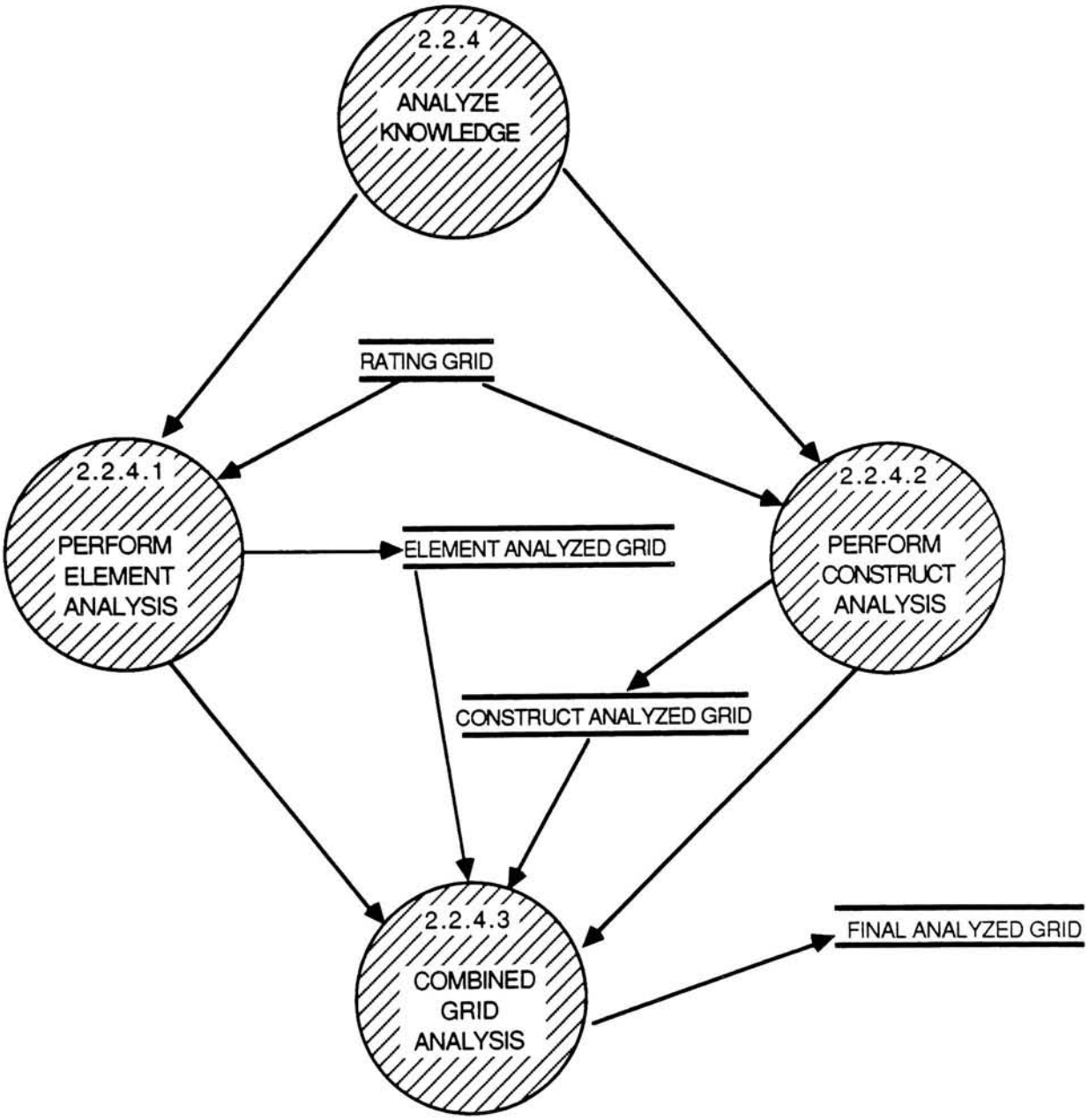
- Perform Application Evaluation
- Give Indication on Success of System to be Developed



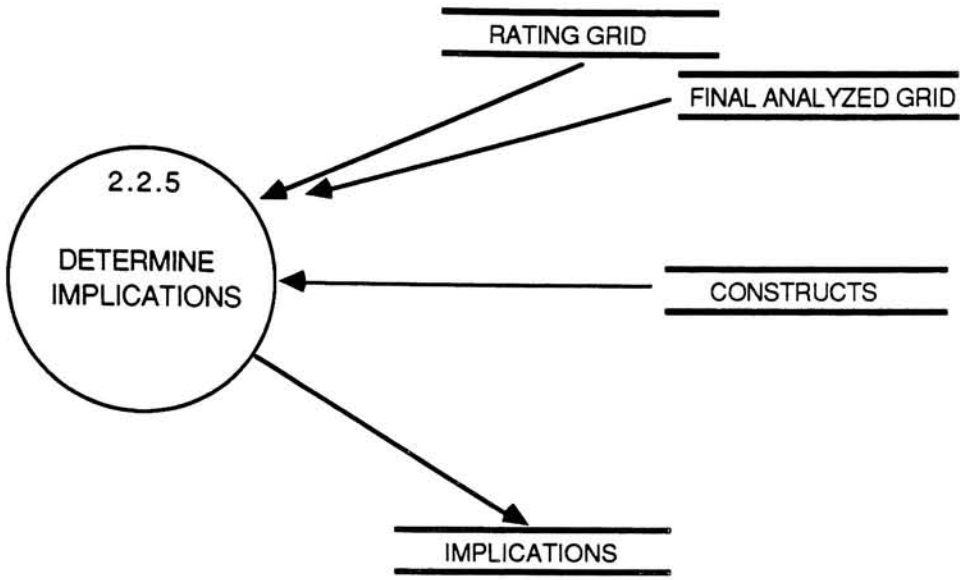
- GATHER BASIC BACKGOUND DATA



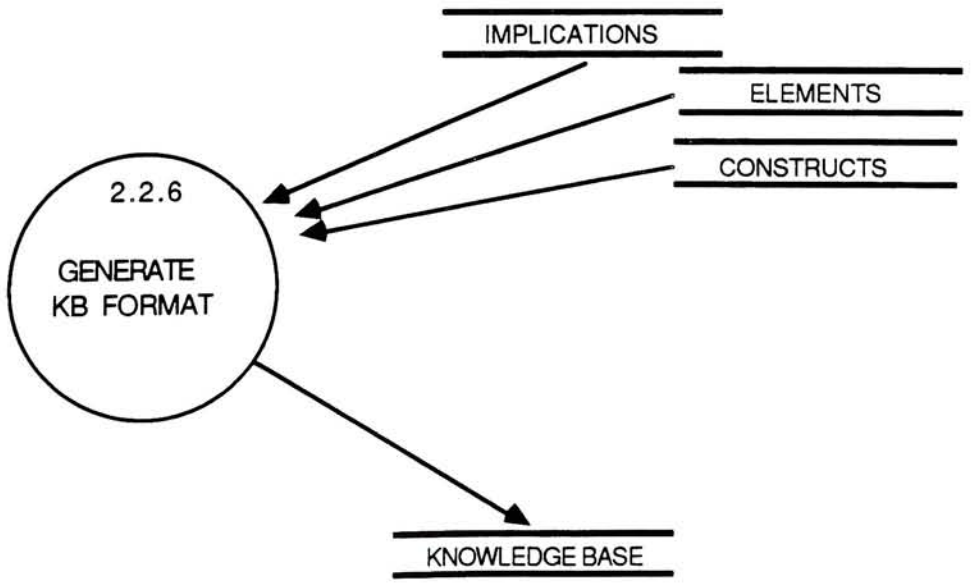

2.2.3
ACQUIRE KNOWLEDGE
ELEMENTS
ELEMENTS
CURRENT TRIAD
2.2.3.1
ELICIT ELEMENTS
2.2.3.4
ELICIT RATINGS
USER
2.2.3.2
GENERATE TRIADS
2.2.3.3
ELICIT CONSTRUCTS
2.2.3.5
BUILD RATING GRID
TRIADS
CONSTRUCTS
ELEMENTS
RATING GRID


2.2.4
ANALYZE KNOWLEDGE
RATING GRID
2.2.4.1
PERFORM ELEMENT ANALYSIS
ELEMENT ANALYZED GRID
2.2.4.2
PERFORM CONSTRUCT ANALYSIS
CONSTRUCT ANALYZED GRID
2.2.4.3
COMBINED GRID ANALYSIS
FINAL ANALYZED GRID

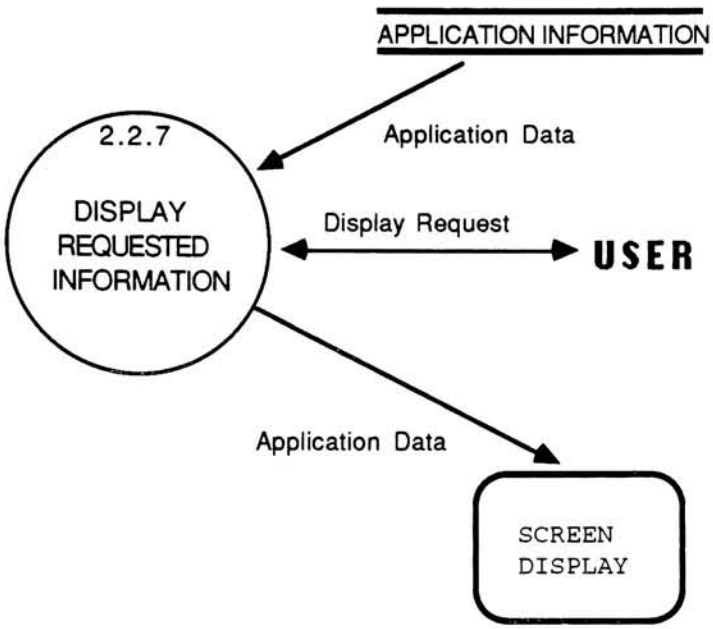
- ANALYZED GRID USED IF ANALYSIS IS PERFORMED



- IMPLICATIONS ARE THE BASIS FOR THE OUTPUT (OTHER INFORMATION USED AS NEEDED)



- APPLICATION INFORMATION CONSISTS OF ELEMENTS, CONSTRUCTS, RATING GRID, and IMPLICATIONS



APPENDIX C

KE-KIT OVERVIEW

KE-KIT OVERVIEW 1990-1991, John Scott Parsons

KE-KIT.OVR

KE-KIT is a set of tools used to automate several aspects of knowledge engineering for intelligent systems (ie. Expert Systems or Knowledge-Based Systems).

The use of KE-KIT is entirely controlled through the menu system. Once a function is initiated, you will be prompted for any needed information. Every function also has some form of help to lead you through the process.

Knowledge Engineering functions within KE-KIT include:

> Application Evaluation
> Knowledge Acquisition
> Knowledge Base Generation

Since KE-KIT is a small scale prototype, all of these functions have limited capabilities.

The "Application Evaluation" conducted is a quick assessment of the application, its domain, and surrounding issues. The objective is to determine the feasibility of using a knowledge-based approach and to identify potential problem areas. It is highly recommended that this feature be used (at least until the analysis becomes second nature).

"Knowledge Acquisition" functions are divided into structured and non-structured approaches.

The non-structured approach is intended to obtain background information concerning the problem and its solution. This data should include the purpose for the system being developed or prototyped. Any data provided will only be used for documentation and reference material.

Structured knowledge acquisition is handled through a psychology-based approach. The approach is based on Personal Construct Theory developed by George Kelly. A tool called a repertory grid (grid for short) is used to explore thinking, experience, and available expertise.

You do not have to know anything about this technique or psychology to use this technique. You will be prompted for information. You must only decide what information to provide and when to stop.

The "knowledge-base generation" function take the information obtained through the knowledge acquisition phase and converts it into a usable form for a knowledge-based system tool. Currently this form is based upon the representation of rules (VP-EXPERT).

KE-KIT processing is largely determined by the menu structure which you have most likely already seen and used. A series of menus, sub-menus, and pop-up windows guide the user through the various knowledge engineering and support activities. More detail can be obtained by referencing the available functions within KE-KIT. In addition to this overview, there
is a HELP facility and a DICTIONARY that gives more detailed information.

The output as a result of the use of KE-KIT should include an overview of the problem domain, the knowledge or expertise, and prototype knowledge base using a rule structure. The target format is based on the KBS tool VP-EXPERT.

Information used by and created by KE-KIT use the following files:

KEKIT.DCT - KE-KIT Dictionary
KEKIT.EXE - The KE-KIT program
KEKIT.HLP - KE-KIT HELP
KEKIT.IMP - Temporary file of implications created by KE-KIT
KEKIT.OVR - KE-KIT Overview

?????.KIT - A saved application created by KE-KIT
?????.VPX - A prototype VP-EXPERT system created by KE-KIT

WARNING: KE-KIT is a basic system without analysis techniques and advanced features to handle such things as consistency, uncertainty, multiple points of view, optimization, explanations, debugging, and so on. The tool, however, is meant to be flexible and capable of eliciting enough expertise to create a prototype system.

APPENDIX D

KE-KIT HELP Function

KE-KIT HELP 1990-1991, John Scott Parsons

KEKIT.HLP

KEKIT MENU

The main menu is considered the "control center" for all functions executed through the KE-KIT program.

Valid movement/selection keys are:

```
Arrow Keys - movement of selection bar
[Enter]    - selects menu item under bar
[Esc]      - backup 1 menu (or exit from highest level)
```

You may also select a menu item by pressing its first letter if it is unique to that option.

Cursor Movement - BROWSE

```
ESCAPE      - exit the browse mode.

Right Arrow - move the cursor position one column to the
              right
Left Arrow  - move the cursor position one column to the
              left

Up Arrow    - move the cursor position one row up
Down Arrow  - move the cursor position one row down

Page Up     - move the cursor position up a window at a
              time
Page Down   - move the cursor position down a window at a
              time

Ctrl-END    - move to the first column on the last row
Ctrl-HOME   - move to the first column in the first row

RETURN      - move to the first column in the next row

Alt-F       - find or search for a text string
```

Cursor Movement - INPUT

BACKSPACE	- delete characters to the left
DELETE	- delete character under cursor
ESCAPE	- clear buffer
INSERT	- insert a space under the cursor
Left Arrow	- move 1 position to the left
Right Arrow	- move 1 position to the right
END	- move cursor to end of string
HOME	- move cursor to beginning of string

Up Arrow	- terminate input field & move to next string
Down Arrow	- terminate input field & move to next string
Return	- terminate input field & move to next string
PageUp	- terminate input field & move to next string
PageDown	- terminate input field & move to next string
Tab	- terminate input field & move to next string
Shift Tab	- terminate input field & move to next string

NOTE: Fields are defined such that there is an automatic return when the maximum length of the input string is reached.

FILE Menu

This pull-down menu contains various file and system commands. See each option for more details.

Load

This function is used to "load" an file related to a previously started system using KE-KIT. A list of available application files are displayed to choose from. Choosing a file will load that application.

Change Dir

This function is used to change the current working disk drive to another specified drive.

Print

This function will allow you to print various portions of the application being developed. For example, background data, elements,

constructs, implications, and rules. Each must be entered or generated in some form before it can be printed. Otherwise only header information is printed.

Save

This function is used to save all files related to the current system being built using KE-KIT.

Directory

This function will allow you to display a list of all files in the current directory or those matching the indicated pattern. You must indicate a desired file specification (mask).

OS Shell

This function will allow you to exit to DOS. At the DOS environment, most valid commands or programs can be executed. When finished in DOS, type 'EXIT' to return to KE-KIT and continue processing.

Rename

This function is used to rename the current KE-KIT application files to another valid name.

KNOWLEDGE ENGINEERING

The Knowledge Engineering functions are the "core" of KE-KIT. They are the various routines used to gather application knowledge and subsequently transform it into a usable form for a Knowledge-Based System or an Expert System.

The functions included are:

EVALUATOR - takes a quick look at the application and determines a likelyhood of success
HISTORIAN - captures domain background data for the current application
BUILDER - performs the specific knowledge or expertise acquisition
ANALYZER - does analysis on the current set of knowledge (NOT IMPLEMENTED)
DETECTIVE - derives simple implications based upon the "knowledge" gathered

GENERATOR - generates a "starting" knowledge base to be used by a specified knowledge based system or expert system tool

Also see: EVALUATOR HISTORIAN BUILDER
ANALYZER DETECTIVE GENERATOR

APPLICATION EVALUATION --> EVALUATOR

This function takes a quick look at the application domain at hand as it applies to four different success areas. These areas include: justification, expertise, problem characteristics, and organizational or political characteristics.

The recommendation and/or advice given is a result of the answers given to the previous four sets of application related questions. It may be possible that more information is need to make a more accurate assessment.

It is now your decision. Is the application worth the risk? Proceed or abandon the application at your own discretion.

This portion of KE-KIT is also known as EVALUATOR.

BACKGROUND INFORMATION --> HISTORIAN

This function will collect a minimal amount of background data about the application domain. This system will only process data asked for here. However, the file that gets created from this function can be expanded with any text editor at some later point.

Information requested consists of:

- application title
  (1st 8 positions used for files created)
- developer's name
  > more than likely the user of this system
- expert list
  > the expert(s) in solving this problem
- user list

> those who will use the system in some way

- application purpose
  > short statement of the objective of the system

- domain problem
  > short description of the problem being addressed

This portion of KE-KIT is also known as HISTORIAN.

KNOWLEDGE ACQUISITION --> BUILDER

This is the set of functions used to elicit domain knowledge for the application being built. The process is done in several steps. First, elements are collected. Secondly, constructs are elicited based on these elements. Finally, all elements are rated based upon the current construct. This process is continued until constructs can no longer be entered or the system limitations of 15 is reached.

In theory one should be able to add, change, or delete elements and constructs. Also, ratings can be adjusted as appropriate.

The technique of knowledge acquisition used is based on a theory originally developed in the field of psychology.

This portion of KE-KIT is also known as BUILDER.

Also see: CONSTRUCTS ELEMENTS RATINGS

ELEMENTS

Elements represent specific things of the same type. For example, they could be people, events, pictures, ideas, objects, activities, or whatever you choose. In this system, they will represent some type of conclusion, recommendation, or action to be taken.

Each element should be drawn from the same category and able to provide representative coverage of the domain. Do not include an element which is a subset of another.

In this context, the elements given should represent some kind of action or the result of some other actions. An element then may represent the action required to solve a problem, to fix a specific situation, or some other means of categorizing other bits of information (problem-solving or otherwise).

Also see: CONSTRUCTS RATINGS BUILDER

CONSTRUCTS

A construct can be thought of as an attribute or trait. Constructs can be used to reach conclusions or anticipate events. They are used to discriminate between elements of the domain and thus identify differences between elements. This is usually done by looking at a triad (set of 3) of elements.

Each construct involves two poles. Each represents one end of a dichotomy. One may be thought of the contrast or opposite of the other, but complete opposition is not a requirement. The left pole (as used here) represents that which holds true for most of the immediately perceived context (2 of the 3 elements). The right pole is the "contrasting" pole.

A series of random triads will be presented and as many constructs as possible for each triad should be entered.

Also see: ELEMENTS RATINGS BUILDER

RATINGS

A rating is used to identify which "pole" an element most represents. The rating scale is based upon the fact that emphasis may be placed upon the similarity or left pole. There may be several different rating scales used in the "real" world. The one used for KE-KIT is defined below.

Valid ratings are as follows:

```
1   --- the element represents the "left" pole
0   --- the element represents the "right" pole
B/b -- the element represents both poles with
       an equal degree
```

N/n -- the element represents neither pole

The implementation of KE-KIT ignores any rating of B(b) or N(n). These ratings should not occur if the elements and constructs are defined with sufficient detail. If this has been done and these ratings still apply, the rating scale used is not defined well enough and another scale should be sought.

Also see: CONSTRUCTS ELEMENTS BUILDER

KNOWLEDGE ANALYSIS --> ANALYZER

This function is used to analyze the current set of knowledge for among other things, consistency, completeness, and other validation and optimization related issues.

This function is NOT currently implemented and is considered beyond the scope of this prototype project.

This portion of KE-KIT is also known as ANALYZER.

IMPLICATION DERIVATION --> DETECTIVE

This function is used to "generate" implications based upon the current set of knowledge. These implications, if approved, can then be used to produce an initial Knowledge-Based System (KBS) that can be used by a shell.

This portion of KE-KIT is also known as DETECTIVE.

GENERATE KNOWLEDGE BASE --> GENERATOR

This function is used to "generate" a starting knowledge base to be used by an expert system shell. It must be emphasized that the knowledge base created is far from a final product and must be enhanced using the KBS tool.

The set of implications must be derived before this function can be executed; otherwise, there will be nothing to generate the knowledge base with.

This portion of KE-KIT is also known as GENERATOR.

Also see: DETECTIVE

DISPLAY CAPTURED KNOWLEDGE --> REVIEWER

This function will allow you to display various portions of the application being developed. For example, background data, elements, constructs, implications, and rules. Each must be entered or generated in some form before it can be displayed. Otherwise only header information is presented.

Also see: Print

KE-KIT MISCELLANEOUS FEATURES

This list of features is made up of functions not directly related to the processing of an application. These features exist more guide a user on the use of KE-KIT and help them better understand the processing of KE-KIT.

These features include:

- KE-KIT Overview

  This feature gives a short introduction KE-KIT.

- KE-KIT Help Facility

  HELP information is contained within this file.

- KE-KIT Dictionary Facility

  This facility tries to clarify some terminology used.

APPENDIX E

KE-KIT DICTIONARY / TERMS

KE-KIT DICTIONARY 1990-1991, John Scott Parsons

KEKIT.DCT

The following list is in alphabetical order. Each definition may in turn reference another listing for further details or comparison. Please keep in mind that this is a limited list. Feel free to add other definitions to this list at any time.

BACKWARD CHAINING

Backward chaining is one of several control strategies that regulate the order in which inferences are drawn. In a rule-based system, backward chaining is initiated by a "goal" rule. The system attempts to determine if the goal rule is correct or satisfied. It backs up to the if clauses of the rule and tries to determine if they are correct or satisfied. This in turn leads the system to consider other rules that would confirm the "if" clauses. In this way the system "backs" into its rules. Eventually, the back-chaining sequence ends - either a result is found or a question is asked which will start the process again.

Inference based on a backward-chaining control strategy is goal-directed.

SEE: Forward Chaining, Inference

CERTAINTY

Certainty represents the degree of confidence one has in a fact or relationship.

SEE: Certainty Factor, Probability

CERTAINTY FACTOR

A certainty factor is a weight given to a fact or relationship to indicate the degree of confidence one has in it.

SEE: Certainty, Probability

CONSTRUCTS

A construct can be thought of as an attribute or trait. Constructs can be used to reach conclusions or anticipate events. They are used to discriminate "elements" of the domain and thus identify differences between them. This is usually done by looking at a triad (set of 3) of elements.

Each construct involves two poles. Each represents one end of a dichotomy. One may be thought of the contrast or opposite of the other, but complete opposition is not a requirement. The right pole (as used here) represents that which holds true for most of the immediately perceived context (2 of the 3 elements). The left pole is the "contrasting" pole.

A series of random triads will be presented and as many constructs as possible for each triad should be entered.

DEEP KNOWLEDGE

Deep knowledge consists of basic theories, basic/first principles, axioms, and facts about the domain.

SEE: Surface Knowledge

DOMAIN

The domain for an application is considered to be the topical area or region of knowledge being acquired and analyzed for the application at hand.

ELEMENTS

Elements represent specific things of the same type. For example, they could be people, events, pictures, ideas, objects, activities, or whatever you choose.

Each element should be drawn from the same category and able to provide representative coverage of the domain. Do not include an element which is a subset of another.

In this context, the elements given should represent some kind of action to take or conclusions that can be made. An

element then may represent the action required to solve a problem, to fix a specific situation, or some other means of categorizing other bits of information (problem-solving or otherwise).

EXPERT SYSTEM

An 'Expert System' is a computer system which attains expert-level performance in solving real-world problems.

It is a system which provides timely problem solving capabilities of a MODEL EXPERT usable by a non-expert, who might not have that resource readily available.

EXPERTISE

Expertise consists of the skill and knowledge possessed by some humans that result in performance far above what is considered the norm. The make up of expertise may include large amounts of information combined with rules-of-thumb, simplifications, rare facts, and wise procedures in such a way that one can analyze specific types of problems in an efficient manner.

EXPERTS

WHAT IS AN EXPERT?

- a person who because of training and EXPERIENCE is able to do things the rest of us can not
- a person who is both PROFICIENT and EFFICIENT in their actions
- a person WIDELY RECOGNIZED as being able to solve a particular type of problem others can not solve nearly as efficient or effectively
- a person others use as a RESOURCE when solving their own problems

EXPERTS:

- know a great many things and have tricks for applying what they know to problems and tasks
- are good at plowing through irrelevant information in order to get at basic issues
- are good at recognizing problems they face as instances of types which they are familiar

EXPLANATION

An explanation represents information presented to justify a particular course of action. This information is usually presented to a user of a knowledge system so that the reasoning strategy can be better understood.

FACT

A fact is most simply a statement whose validity is widely accepted.

FRAME

A "frame" is a knowledge representation scheme that associates an object with a collection of features (eg. facts, rules, defaults, values). Each feature is stored in a slot. A frame is a set of slots related to a specific object. In traditional programming terms, a frame may be thought of as a property list or a record.

FORWARD CHAINING

Forward chaining is one of several control strategies that regulate the order in which inferences are drawn. In a rule-based system, forward chaining begins by asserting/executing all of the rules whose "if" clauses are true. It then checks to determine what additional rules might be true, given the facts that have already been established. This processed is repeated until the program reaches a goal or runs out of new possibilities.

Inference based on a forward-chaining control strategy is data-directed.

SEE: Backward Chaining, Inference

HEURISTIC

A heuristic is usually thought of as a rule-of-thumb or some other simplification that reduces/limits the search required for a large problem space. Unlike algorithms, heuristics do not guarantee correct solutions.

HEURISTIC RULE

A heuristic rule is one that has been written to capture the heuristics which an expert uses to solve a problem. The expert's original heuristics may not have taken the form of if-then rules. One of the problems involved in building a knowledge system is converting and expert's heuristic knowledge into rules.

SEE: Heuristic, Rules

IF-THEN RULE

An if-then rule is a statement of a relationship among a set of facts. The relationships may be definitional (ie. IF female AND married THEN wife) or heuristic (ie. IF cloudy THEN take umbrella)

SEE: Rules, Heuristic, Heuristic Rule,

IMPLICATIONS

Implications provide new information or facts based on given information. They are derived rather than given as fact. They are also not necessarily verified as being true; whereas a RULE is given to be true.

Adding or removing given information may result in changes to existing implications or new ones may result.

INDUCTION SYSTEM

An induction system is a knowledge system which has a knowledge base consisting of examples. An induction algorithm builds a decision tree from the examples and the system goes on to deliver advice or a solution to a problem.

INFERENCE

The process by which new facts are derived from known facts is know as inference. A rule combined with a rule of inference or control strategy and a known fact results in a new fact.

SEE: Rule, Forward Chaining, Backward Chaining,

INFERENCE ENGINE

The portion of a knowledge system that contains the inference and control strategies is known as the inference engine. More broadly, the inference engine may also include various knowledge acquisition, explanation, and user-interface subsystems.

SEE: Inference

INTERFACE

An interface is any link between one computer program/system and any thing else external to that program or system. Knowledge system typically have interfaces for development (knowledge acquisition) and for users. In addition, some systems have interfaces that pass information to and from other programs, data bases, display devices, or sensors.

KNOWLEDGE

Knowledge consists of any collection of facts and relationships which, when exercised, produces competent performance. The quantity and quality of knowledge possessed by a person of a system can be judged by the variety of situations in which successful results can be obtained by the person or system.

KNOWLEDGE ACQUISITION

The process of Knowledge Acquisition is part of the Knowledge Engineering set of activities. It may use one or many techniques activities to obtain information about problem solving. It is more than just fact gathering - it also involves determining how to use the information. The activities of collecting, locating, and refining are used during this process.

A person acting as a knowledge engineer usually spends time with an expert in order to acquire knowledge for a Knowledge-Based System or an Expert System. This process is iterative and at times very difficult. As thus it is very time consuming.

KNOWLEDGE BASE

The portion of a knowledge system that consists of the facts and heuristics about a domain is known as a knowledge base.

SEE: Knowledge, Facts, Heuristics, Domain

KNOWLEDGE-BASED SYSTEM

A KNOWLEDGE-BASED system is a computer system which models the human reasoning or thought process to solve complex or demanding problems.

The difference between an Expert System and a Knowledge-Based System is based upon either the source of the acquired knowledge or the level of knowledge attained and used by the system. Your point of view will define the difference.

KNOWLEDGE ENGINEER

A knowledge engineer is an individual whose speciality is assessing problems, acquiring knowledge, and building knowledge systems. This usually implies training in cognitive science, computer science, and artificial intelligence.

SEE: Knowledge Engineering, Knowledge Acquisition

KNOWLEDGE ENGINEERING

Knowledge Engineering is the process of building and Expert System or a Knowledge-Based System(KBS). It is the act and all associated activities of transferring knowledge from an Expert to a computer program. In many respects, knowledge engineering is similar to traditional software engineering.

KNOWLEDGE REPRESENTATION

Knowledge representation defines the method used to encode and store facts and relationships in a knowledge base. Examples include production rules, semantic networks, frames, objects, and logical expressions.

SEE: Production rules, Semantic networks, Frames, Object

KNOWLEDGE SOURCE

The source of expert knowledge or needed information may take many forms. These include the following:

- human experts
- written material (books, journals, reports, etc.)
- databases
- video recordings
- case studies

KNOWLEDGE SYSTEM

A computer program that uses knowledge and inference procedures to solve difficult can be called a knowledge system.

SEE: Expert system, Knowledge-Based system

MODUS PONENS

Modus ponens is a basic rule of logic that asserts that if we know that A implies B and we know for a fact that A is the case, we can assume B.

SEE: Rules

MONOTONIC REASONING

A reasoning system based on the assumption that once a fact is determined it can not be altered during the course of the reasoning process is known as monotonic reasoning.

OBJECT

Broadly, an object refers to physical or conceptual entities that have many attributes.

PROBABILITY

Probability in this context refers to various statistical approaches used to determine the likelyhood of a particular relationship. Some systems use a modified version of Bayesian probability theory to calculate the likelyhood of various outcomes.

SEE: Certainty, Uncertainty

PROBLEM SOLVING

Problem solving is a process in which one starts from an initial state and proceeds to search through a problem space in order to identify the sequence of operations or actions that will lead to a desired goal. Successful problem solving depends upon knowing the initial state, knowing what an acceptable outcome will be, and knowing the elements and operators that define the problem space. If the elements or operators are very large in number or if they are poorly defined, one is faced with a huge or unbounded problem space and an exhaustive search can become impossible.

SEE: Problem Space

PROBLEM SPACE

A problem space is a conceptual or formal area defined by all of the possible states that could occur as a result of interactions between the elements and operators that are considered when a particular problem is being studied.

PRODUCTION RULE

A production or production rule is the term used to describe an if-then rule.

SEE: Rules

PRODUCTION RULE SYSTEM

A production system is a human or computer system that has a data base of production rules and some control mechanism that selects applicable rules in an effort to reach some goal state.

SEE: Production Rule, Rules

PROTOTYPE

An initial version of a knowledge/expert system is called a prototype. A small version of a system is developed to test effectiveness of the overall knowledge representation and inference strategies being employed to solve a particular problem. Also, a prototype is often built as a "proof-of-

concept."

SEE: Inference, Knowledge System, Knowledge-Based System, Expert System, Knowledge Representation

REASONING

Reasoning is the process of drawing inferences or conclusions.

SEE: Inference

REPRESENTATION

A representation is the way in which a system stores knowledge about the domain.

SEE: Knowledge Representation, Knowledge, Domain

RULES

Rules are one of the most simplest forms of knowledge representation. Knowledge in this form often uses the IF-THEN syntax. In this case a consequence/conclusion holds true if the condition exists or a stated fact is known. A rule then defines a relationship between other bits of information.

RULE-BASED SYSTEM

A rule-based system is a computer program that represents knowledge by means of rules.

SEE: Rules, Production Rule, Production Rule System

SEARCH SPACE

SEE: Problem Solving, Problem Space

SEMANTIC NETWORKS

A semantic network is a type of knowledge representation that formalizes objects and values as nodes and connects the nodes with arcs or links that indicate the relationships between the various nodes.

SEE: Knowledge Representation

SHELL/TOOL

Expert System tools or system shells are used to help facilitate the development of an application. By using these tools, the developer can concentrate on the domain specifics rather than the common system requirements.

The alternative is to use a programming language and develop most of the required functions from ground zero for each and every application.

SOFTWARE ENGINEER

An individual who designs conventional computer software is commonly known as a software engineer.

SEE: Knowledge Engineer

TECHNOLOGY TRANSFER

In the context of expert systems, technology transfer is the process by which knowledge engineers turn over an expert system to a user group. Since expert systems need to be updated, knowledge engineers need to train users to maintain a system before it is deployed to the user environment. In effect, some users must learn to do some knowledge engineering.

SEE: Knowledge Engineering, Knowledge Engineer

UNCERTAINTY

Uncertainty refers to a value that can not be determined during a consultation.

SEE: Certainty

APPENDIX F

ANIMALS APPLICATION IMPLICATIONS

Application Title: ANIMALS

IMPLICATIONS DERIVED

```
>>  IF      only a few variations
      AND not known to be quick
      AND large
        AND not part of "cat family"
        AND lives in wild
    THEN  bear

>>  IF      only a few variations
      AND not known to be quick
      AND large
        AND not part of "cat family"
        AND lives in jungle
        AND lives in wild
    THEN  elephant

>>  IF      many variations exist
      AND known for speed
        AND relatively small
        AND belongs to "cat family"
      AND house pet
      AND house pet
    THEN  cat

>>  IF      many variations exist
      AND known for speed
        AND relatively small
        AND not part of "cat family"
      AND house pet
      AND house pet
    THEN  dog
```

```
>>  IF      only a few variations
    AND known for speed
      AND relatively small
      AND belongs to "cat family"
     AND lives in jungle
     AND lives in wild
  THEN  tiger

>>  IF      only a few variations
    AND known for speed
      AND relatively small
      AND belongs to "cat family"
     AND lives in jungle
     AND lives in wild
  THEN  lion
```

ANIMALS APPLICATION
KE-KIT CREATED VP-EXPERT FILE

```
ACTIONS

    DISPLAY "Welcome to the  ANIMALS system.
         Press any key to begin. ~"

FIND ANSWER

END;

AUTOQUERY

ASK  lives in wild: "lives in wild - Does this apply to the problem?"
ASK  house pet: "house pet - Does this apply to the problem?"
ASK  lives in jungle: "lives in jungle - Does this apply to the problem?"
ASK  house pet: "house pet - Does this apply to the problem?"
ASK  belongs to "cat family": "belongs to "cat family"  Does this apply to the problem?"
ASK  not part of "cat family": "not part of "cat family" - Does this apply to the problem?"
ASK  relatively small: "relatively small - Does this apply to the problem?"
ASK  large: "large - Does this apply to the problem?"
ASK  known for speed: "known for speed - Does this apply to the problem?"
ASK  not known to be quick: "not known to be quick - Does this apply to the problem?"
ASK  many variations exist: "many variations exist - Does this apply to the problem?"
ASK  only a few variations: "only a few variations - Does this apply to the problem?"

CHOICES lives in wild,house pet:YES, NO;
CHOICES lives in jungle,house pet:YES, NO;
CHOICES belongs to "cat family",not part of "cat family":YES, NO;
CHOICES relatively small,large:YES, NO;
CHOICES known for speed,not known to be quick:YES, NO;
CHOICES many variations exist,only a few variations:YES, NO;

RULE 1
IF      only a few variations = YES
  AND   known for speed = YES
  AND   relatively small = YES
  AND   belongs to "cat family" = YES
  AND   lives in jungle = YES
  AND   lives in wild = YES
THEN    ANSWER  = lion ;
```

```
RULE 2
IF      only a few variations = YES
  AND   known for speed = YES
  AND   relatively small = YES
  AND   belongs to "cat family" = YES
  AND   lives in jungle = YES
  AND   lives in wild = YES
THEN    ANSWER  = tiger;

RULE 3
IF      many variations exist = YES
  AND   known for speed = YES
  AND   relatively small = YES
  AND   not part of "cat family" = YES
  AND   house pet = YES
  AND   house pet = YES
THEN    ANSWER  = dog;

RULE 4
IF      many variations exist = YES
  AND   known for speed = YES
  AND   relatively small = YES
  AND   belongs to "cat family" = YES
  AND   house pet = YES
  AND   house pet = YES
THEN    ANSWER  = cat;

RULE 5
IF      only a few variations = YES
  AND   not known to be quick = YES
  AND   large = YES
  AND   not part of "cat family" = YES
  AND   lives in jungle = YES
  AND   lives in wild = YES
THEN    ANSWER  = elephant;

RULE 6
IF      only a few variations = YES
  AND   not known to be quick = YES
  AND   large = YES
  AND   not part of "cat family" = YES
  AND   lives in wild = YES
THEN    ANSWER  = bear;
```

ANIMALS APPLICATION
KE-KIT SAVED KNOWLEDGE

B: ANIMALS*John S. Parsons*01/08/92*ZooKeeper*People interested in animals
B: It often may be difficult to determine animal types
B: This system is a start in classifying animals

E: 1*6*elephant*an animal
E: 1*5*dog*an animal
E: 1*4*tiger*an animal
E: 1*3*lion *an animal
E: 1*2*cat*an animal
E: 1*1*bear*an animal

C: 1*0*2*6*variations of the animal*relative number of variations for the animal*many variations exist*only a few variations
C: 1*0*2*5*speed of movement*how fast is the animal perceived to be*known for speed*not known to be quick
C: 1*0*2*4*size of animal*what is the general size of the animal*relatively small*large
C: 1*0*2*3*family of animal*what family does the animal belong to*belongs to "cat family"*not part of "cat family"
C: 1*0*2*2*habitat location*general habitat location for the animal*lives in jungle*house pet
C: 1*0*2*1*general habitat*what is the general living habitat of the animal*lives in wild*house pet

G:1*0*2*lion *3*1*1*2*1*3*1*4*1*5*1*6*0*
G:1*0*2*tiger*4*1*1*2*1*3*1*4*1*5*1*6*0*
G:1*0*2*dog*5*1*0*2*0*3*0*4*1*5*1*6*1*
G:1*0*2*cat*2*1*0*2*0*3*1*4*1*5*1*6*1*
G:1*0*2*elephant*6*1*1*2*1*3*0*4*0*5*0*6*0*
G:1*0*2*bear*1*1*1*2*n*3*0*4*0*5*0*6*0*

T: 1*3*2*5
T: 1*2*5*6
T: 1*6*4*2
T: 1*3*1*2
T: 1*1*4*5
T: 1*2*3*6
T: 1*6*4*5
T: 1*3*1*6
T: 1*5*1*3
T: 1*1*3*4
T: 1*4*3*5
T: 1*6*1*4
T: 1*1*6*5
T: 1*5*1*2

SPORTS APPLICATION IMPLICATIONS

Application Title: SPORTS

IMPLICATIONS DERIVED

>> IF constant playing field
AND contact sport
AND played indoors
AND team play
AND Winter activity
AND ball used
THEN basketball

>> IF constant playing field
AND contact sport
AND played indoors
AND team play
AND Winter activity
AND puck used
THEN hockey

>> IF constant playing field
AND non-contact sport
AND played outdoors
AND team play
AND Summer activity
AND ball used
THEN baseball

>> IF constant playing field
AND contact sport
AND played outdoors
AND team play
AND Summer activity
AND ball used
THEN soccer

```
>>  IF      playing field is variable
     AND non-contact sport
    AND played outdoors
     AND individual play
     AND Summer activity
   AND ball used
  THEN  golf

>>  IF      constant playing field
     AND contact sport
    AND played outdoors
    AND team play
    AND ball used
  THEN  football
```

SPORTS APPLICATION
KE-KIT CREATED VP-EXPERT FILE

```
ACTIONS

    DISPLAY "Welcome to the SPORTS system.
        Press any key to begin. ~"

FIND ANSWER

END;

AUTOQUERY

ASK playing field is variable: "playing field is variable - Does this apply to the problem?"
ASK constant playing field: "constant playing field - Does this apply to the problem?"
ASK contact sport: "contact sport - Does this apply to the problem?"
ASK non-contact sport: "non-contact sport - Does this apply to the problem?"
ASK played outdoors: "played outdoors - Does this apply to the problem?"
ASK played indoors: "played indoors - Does this apply to the problem?"
ASK team play: "team play - Does this apply to the problem?"
ASK individual play: "individual play - Does this apply to the problem?"
ASK Winter activity: "Winter activity - Does this apply to the problem?"
ASK Summer activity: "Summer activity - Does this apply to the problem?"
ASK ball used: "ball used - Does this apply to the problem?"
ASK puck used: "puck used - Does this apply to the problem?"

CHOICES playing field is variable,constant playing field:YES, NO;
CHOICES contact sport,non-contact sport:YES, NO;
CHOICES played outdoors,played indoors:YES, NO;
CHOICES team play,individual play:YES, NO;
CHOICES Winter activity,Summer activity:YES, NO;
CHOICES ball used,puck used:YES, NO;

RULE 1
IF      constant playing field = YES
  AND   contact sport = YES
  AND   played indoors = YES
  AND   team play = YES
  AND   Winter activity = YES
  AND   ball used = YES
THEN    ANSWER  = basketball;
```

```
RULE 2
IF    constant playing field = YES
  AND  contact sport = YES
  AND  played indoors = YES
  AND  team play = YES
  AND  Winter activity = YES
  AND  puck used = YES
THEN   ANSWER = hockey;

RULE 3
IF    constant playing field = YES
  AND  non-contact sport = YES
  AND  played outdoors = YES
  AND  team play = YES
  AND  Summer activity = YES
  AND  ball used = YES
THEN   ANSWER = baseball;

RULE 4
IF    constant playing field = YES
  AND  contact sport = YES
  AND  played outdoors = YES
  AND  team play = YES
  AND  Summer activity = YES
  AND  ball used = YES
THEN   ANSWER = soccer;

RULE 5
IF    playing field is variable = YES
  AND  non-contact sport = YES
  AND  played outdoors = YES
  AND  individual play = YES
  AND  Summer activity = YES
  AND  ball used = YES
THEN   ANSWER = golf;

RULE 6
IF    constant playing field = YES
  AND  contact sport = YES
  AND  played outdoors = YES
  AND  team play = YES
  AND  ball used = YES
THEN   ANSWER = football;
```

SPORTS APPLICATION
KE-KIT SAVED KNOWLEDGE

B: SPORTS*John Scott Parsons*01/08/92*Sport Club members*women and those not familiar with sports
B: Many people do not know very much about sports
B: This system will help familiarize people with sports

E: 1*6*baseball*type of sport
E: 1*5*soccer*type of sport
E: 1*4*hockey*type of sport
E: 1*3*golf*type of sport
E: 1*2*football*type of sport
E: 1*1*basketball*type of sport

C: 1*0*2*6*state of playing field*what type of playing field is used*playing field is variable*constant playing field
C: 1*0*2*5*contact type*does the sport have physical contact*contact sport*non-contact sport
C: 1*0*2*4*where normally played*place where sport is normally played*played outdoors*played indoors
C: 1*0*2*3*type of play*sport involves team or individual play*team play*individual play
C: 1*0*2*2*season normally played*when is the sport normally played*Winter activity*Summer activity
C: 1*0*2*1*main object of play*what object is used in the game*ball used*puck used

G:1*0*2*football*2*1*1*2*n*3*1*4*1*5*1*6*0*
G:1*0*2*golf*3*1*1*2*0*3*0*4*1*5*0*6*1*
G:1*0*2*soccer*5*1*1*2*0*3*1*4*1*5*1*6*0*
G:1*0*2*baseball*6*1*1*2*0*3*1*4*1*5*0*6*0*
G:1*0*2*hockey*4*1*0*2*1*3*1*4*0*5*1*6*0*
G:1*0*2*basketball*1*1*1*2*1*3*1*4*0*5*1*6*0*

T: 1*2*1*4
T: 1*3*4*5
T: 1*3*6*4
T: 1*2*5*4
T: 1*4*3*2
T: 1*2*1*6
T: 1*3*2*5
T: 1*5*6*1
T: 1*5*2*1
T: 1*6*4*5
T: 1*5*4*1
T: 1*1*6*3
T: 1*3*6*5
T: 1*2*5*6
T: 1*6*2*3

INVEST APPLICATION IMPLICATIONS

Application Title: INVEST

IMPLICATIONS DERIVED

>> IF non-experienced investor
AND does not require a broker
AND rising interest rates exist
AND low to medium risk
THEN money market fund

>> IF non-experienced investor
AND does not require a broker
AND low to medium risk
THEN conservative mutual funds

>> IF does not require a broker
AND falling interest rates exist
AND medium to high risk
THEN aggressive fund stocks

>> IF experienced investor
AND requires a broker
AND falling interest rates exist
AND medium to high risk
THEN stocks

>> IF non-experienced investor
AND does not require a broker
AND rising interest rates exist
AND low to medium risk
THEN regular passbook savings

>> IF non-experienced investor
AND does not require a broker
AND rising interest rates exist
AND low to medium risk
THEN certificate of deposit

INVEST APPLICATION
KE-KIT CREATED VP-EXPERT FILE

```
ACTIONS

    DISPLAY "Welcome to the  INVEST system.
         Press any key to begin. ~"

FIND ANSWER

END;

AUTOQUERY

ASK  experienced investor: "experienced investor - Does this apply to the problem?"
ASK  non-experienced investor: "non-experienced investor - Does this apply to the problem?"
ASK  does not require a broker: "does not require a broker - Does this apply to the problem?"
ASK  requires a broker: "requires a broker - Does this apply to the problem?"
ASK  falling interest rates exist: "falling interest rates exist - Does this apply to the problem?"
ASK  rising interest rates exist: "rising interest rates exist - Does this apply to the problem?"
ASK  low to medium risk: "low to medium risk - Does this apply to the problem?"
ASK  medium to high risk: "medium to high risk - Does this apply to the problem?"

CHOICES experienced investor,non-experienced investor:YES, NO;
CHOICES does not require a broker,requires a broker:YES, NO;
CHOICES falling interest rates exist,rising interest rates exist:YES, NO;
CHOICES low to medium risk,medium to high risk:YES, NO;

RULE 1
IF      non-experienced investor = YES
  AND   does not require a broker = YES
  AND   rising interest rates exist = YES
  AND   low to medium risk = YES
THEN    ANSWER  = money market fund;


RULE 2
IF      non-experienced investor = YES
  AND   does not require a broker = YES
  AND   low to medium risk = YES
THEN    ANSWER  = conservative mutual funds;
```

RULE 3
IF does not require a broker = YES
AND falling interest rates exist = YES
AND medium to high risk = YES
THEN ANSWER = aggressive fund stocks;

RULE 4
IF experienced investor = YES
AND requires a broker = YES
AND falling interest rates exist = YES
AND medium to high risk = YES
THEN ANSWER = stocks;

RULE 5
IF non-experienced investor = YES
AND does not require a broker = YES
AND rising interest rates exist = YES
AND low to medium risk = YES
THEN ANSWER = regular passbook savings;

RULE 6
IF non-experienced investor = YES
AND does not require a broker = YES
AND rising interest rates exist = YES
AND low to medium risk = YES
THEN ANSWER = certificate of deposit;

INVEST APPLICATION
KE-KIT SAVED KNOWLEDGE

B: INVEST*John Scott Parsons*01/10/92*Investment Counselors*Non-professional investors
B: Many people do not understand how to properly invest
B: This is an attempt to assist unskilled investors

E: 1*6*stocks*investment type
E: 1*5*regular passbook savings*investment type
E: 1*4*money market fund*investment type
E: 1*3*aggressive fund stocks*investment type
E: 1*2*certificate of deposit*investment type
E: 1*1*conservative mutual funds*investment type

C: 1*0*2*4*investor experience*level of experience that the investor has*experienced investor*non-experienced investor
C: 1*0*2*3*need for a broker*investment may or may not be handled by a broker*does not require a broker*requires a broker
C: 1*0*2*2*state of interest rates*direction of interest rates (rising or falling)*falling interest rates exist*rising interest rates exist
C: 1*0*2*1*risk of investment*what level of risk is involved*low to medium risk*medium to high risk

G:1*0*2*certificate of deposit*2*1*1*2*0*3*1*4*0*
G:1*0*2*regular passbook savings*5*1*1*2*0*3*1*4*0*
G:1*0*2*stocks*6*1*0*2*1*3*0*4*1*
G:1*0*2*aggressive fund stocks*3*1*0*2*1*3*1*4*b*
G:1*0*2*conservative mutual funds*1*1*1*2*b*3*1*4*0*
G:1*0*2*money market fund*4*1*1*2*0*3*1*4*0*

T: 1*2*1*3
T: 1*2*4*1
T: 1*4*1*5
T: 1*4*5*2
T: 1*4*2*6
T: 1*5*1*6
T: 1*3*5*6
T: 1*2*5*1
T: 1*3*2*4
T: 1*6*4*1
T: 1*2*1*6
T: 1*3*1*6
T: 1*5*4*3
T: 1*5*1*3
T: 1*5*2*3
T: 1*4*6*3

TRIP APPLICATION IMPLICATIONS

Application Title: TRIP

IMPLICATIONS DERIVED

>> IF no problems for allergies
AND mountain climbing available
AND southern USA
AND southern USA
AND tours not available
AND boating available
AND western USA
THEN Grand Canyon Park

>> IF caution if allergies exist
AND mountain climbing available
AND southern USA
AND southern USA
AND tours available
AND boating available
AND eastern USA
THEN Everglades Park

>> IF no problems for allergies
AND mountain climbing available
AND northern USA
AND northern USA
AND tours not available
AND no boating
AND eastern USA
THEN Shenandoah

```
>>  IF      no problems for allergies
      AND no mountain climbing
     AND northern USA
     AND northern USA
      AND tours not available
     AND western USA
   THEN  Crater Lake

>>  IF      no problems for allergies
      AND no mountain climbing
     AND southern USA
     AND southern USA
      AND tours not available
     AND no boating
     AND eastern USA
   THEN  Hot Springs

>>  IF      no problems for allergies
       AND mountain climbing available
     AND northern USA
     AND northern USA
      AND tours available
     AND no boating
     AND western USA
  THEN  OLYMPIC Park
```

TRIP APPLICATION
KE-KIT CREATED VP-EXPERT FILE

ACTIONS

DISPLAY "Welcome to the TRIP system.
Press any key to begin. ~"

FIND ANSWER

END;

AUTOQUERY

ASK no problems for allergies: "no problems for allergies - Does this apply to the problem?"
ASK caution if allergies exist: "caution if allergies exist - Does this apply to the problem?"
ASK mountain climbing available: "mountain climbing available - Does this apply to the problem?"
ASK no mountain climbing: "no mountain climbing - Does this apply to the problem?"
ASK southern USA: "southern USA - Does this apply to the problem?"
ASK northern USA: "northern USA - Does this apply to the problem?"
ASK southern USA : "southern USA - Does this apply to the problem?"
ASK northern USA: "northern USA - Does this apply to the problem?"
ASK tours available: "tours available - Does this apply to the problem?"
ASK tours not available: "tours not available - Does this apply to the problem?"
ASK boating available: "boating available - Does this apply to the problem?"
ASK no boating: "no boating - Does this apply to the problem?"
ASK eastern USA: "eastern USA - Does this apply to the problem?"
ASK western USA: "western USA - Does this apply to the problem?"

CHOICES no problems for allergies,caution if allergies exist:YES, NO;
CHOICES mountain climbing available,no mountain climbing:YES, NO;
CHOICES southern USA,northern USA:YES, NO;
CHOICES southern USA ,northern USA:YES, NO;
CHOICES tours available,tours not available:YES, NO;
CHOICES boating available,no boating:YES, NO;
CHOICES eastern USA,western USA:YES, NO;

RULE 1
IF no problems for allergies = YES
AND mountain climbing available = YES
AND southern USA = YES
AND southern USA = YES
AND tours not available = YES
AND boating available = YES
AND western USA = YES
THEN ANSWER = Grand Canyon Park;

```
RULE 2
IF     caution if allergies exist = YES
  AND  mountain climbing available = YES
  AND  southern USA = YES
  AND  southern USA  = YES
  AND  tours available = YES
  AND  boating available = YES
  AND  eastern USA = YES
THEN   ANSWER  = Everglades Park;

RULE 3
IF     no problems for allergies = YES
  AND  mountain climbing available = YES
  AND  northern USA = YES
  AND  northern USA = YES
  AND  tours not available = YES
  AND  no boating = YES
  AND  eastern USA = YES
THEN   ANSWER  = Shenandoah;

RULE 4
IF     no problems for allergies = YES
  AND  no mountain climbing = YES
  AND  northern USA = YES
  AND  northern USA = YES
  AND  tours not available = YES
  AND  western USA = YES
THEN   ANSWER  = Crater Lake;

RULE 5
IF     no problems for allergies = YES
  AND  no mountain climbing = YES
  AND  southern USA = YES
  AND  southern USA  = YES
  AND  tours not available = YES
  AND  no boating = YES
  AND  eastern USA = YES
THEN   ANSWER  = Hot Springs;

RULE 6
IF     no problems for allergies = YES
  AND  mountain climbing available = YES
  AND  northern USA = YES
  AND  northern USA = YES
  AND  tours available = YES
  AND  no boating = YES
  AND  western USA = YES
THEN   ANSWER  = OLYMPIC Park;
```

TRIP APPLICATION
KE-KIT SAVED KNOWLEDGE

B: TRIP*John S. Parsons*01/10/92*Travel Agent*people planning a vacation
B: Most vacationers usually don't know much about some vacation spots
B: This system will assist those planning vacations

E: 1*6*Shenandoah*national park
E: 1*5*Crater Lake*national park
E: 1*4*Hot Springs*national park
E: 1*3*Everglades Park*national park
E: 1*2*Grand Canyon Park*national park
E: 1*1*OLYMPIC Park*national park

C: 1*0*2*7*allergy sufferer*caution may be needed for those with allergies*no problems for allergies*caution if allergies exist
C: 1*0*2*6*recreation-mountain climbing*mountain climbing may or not be available*mountain climbing available*no mountain climbing
C: 1*0*2*5*location n/s*general location of park in USA*southern USA*northern USA
C: 1*0*2*4*n/s location*general location of national park*southern USA *northern USA
C: 1*0*2*3*recreation - tours*tours may or may not be available*tours available*tours not available
C: 1*0*2*2*recreation-boating*recreation type of boating may or may not exist*boating available*no boating
C: 1*0*2*1*location of park*general part of country park is located*eastern USA*western USA

G:1*0*2*OLYMPIC Park*1*1*0*2*0*3*1*4*0*5*0*6*1*7*1*
G:1*0*2*Hot Springs*4*1*1*2*0*3*0*4*1*5*1*6*0*7*1*
G:1*0*2*Crater Lake*5*1*0*2*n*3*0*4*0*5*0*6*0*7*1*
G:1*0*2*Shenandoah*6*1*1*2*0*3*0*4*0*5*0*6*1*7*1*
G:1*0*2*Everglades Park*3*1*1*2*1*3*1*4*1*5*1*6*1*7*0*
G:1*0*2*Grand Canyon Park*2*1*0*2*1*3*0*4*1*5*1*6*1*7*1*

T: 1*4*3*6
T: 1*1*3*6
T: 1*1*6*2
T: 1*4*2*6
T: 1*4*1*2
T: 1*4*5*6
T: 1*4*5*2
T: 1*6*4*1
T: 1*3*5*1
T: 1*2*1*3
T: 1*4*5*1
T: 1*1*5*2
T: 1*3*5*6
T: 1*4*3*5
T: 1*6*5*1

APPENDIX G

KE-KIT EVALUATION SYSTEM

The application evaluation function within KE-KIT covers four main areas critical to the success of a knowledge-based system. The following is the list of questions used for each section.

JUSTIFICATION

Is there a need to make scarce/unique knowledge widely available?

Will expertise be lost if it is not captured?

Will there be significant savings or payoffs from a KBS?

Is there a need to increase the efficiency of the decision process?

Is training for handling this problem area expensive?

Will building a KBS help future development?

Can the proposed KBS be integrated with other services or products to increase their value?

Will an improved understanding of the problem, gained through development, be valuable?

EXPERTISE

Is there an expert available who solves problems better than the majority of the intended users?

Is the expertise to be used accurate and correct?

Are there a few key people with specialized knowledge or expertise spending excessive time helping others?

Are the experts willing to work with KEs or KE tools?

Will the expert have time to complete the development process?

If multiple experts contribute, is one the final authority?

Can the experts sufficiently explain their methods so that they can be easily understood?

PROBLEM CHARACTERISTICS

Does the problem require mainly experience-based reasoning?

Is the problem solution dependant on common-sense reasoning?

Does the problem require small amounts of time for the expert to solve (less than 2 hours) or can it be sub-divided?

Do the users require all possibilities known in advance?

GROUP/POLITICAL CHARACTERISTICS

Is there adequate managerial commitment for the effort?

Are the users committed to using the expert system/KBS?

Will the introduction of an expert system cause political or control repercussions either from its use, contents, or recommendations?

Will the system handle a REAL and NECESSARY business need?

Is it acceptable to complete the system in phases?

APPENDIX H

UNCERTAINTY

INTRODUCTION

An important factor in the design and implementation of a Knowledge Based System (KBS) and the knowledge acquisition process revolves around uncertainty handling. Without uncertainty handling, all knowledge is believed to be completely true with no exceptions. Everybody experiences UNCERTAINTY within every lifetime event (we all live in an uncertain world), but it does not prevent decision making. People live with uncertainty with the help of their INTUITION. Intuition is an inner feeling that guides the decision-making process. It is a personal tool that is used without asking how it really works. Intuition comes from one's life experiences; thus it varies from person to person. The intuition process is generally much simpler than the best known methods for treating uncertainty.

Most domains have some level of uncertainty that arises from the environment. Inexact reasoning is common in the sciences. Only a small portion of natural science can be termed exact. Uncertainty may emanate from sources such as limited or inaccurate measured precision, fuzziness (a degree of belonging), lack of (or incomplete) knowledge, and sensitivity to initial conditions. Inexact information may come from: 1) human fuzzy concepts, 2) unreliable information, 3) matching of similar information rather than identical experiences, 4) incomplete information, or 5) differing opinions. Putting this all together, uncertainty may occur when one is not absolutely certain about a piece of information, the boundary of a piece of information is not clear-cut, or when uncertainty and fuzziness both exist.

The amount of certainty depends upon the number of possible answers, and the strength of preference one has for each answer as compared to the other known possibilities. The amount of uncertainty fluctuates and depends upon the known possibilities and the available evidence for each alternative. Certainty is felt when complete belief or complete confidence exists for a single answer. Available knowledge and experiences may fail when new developments or evidence is encountered. This failure leads to several possible answers with a PARTIAL BELIEF for each or even complete uncertainty. *General Uncertainty* exists when it is believed that no certain answer exists. On the other hand, *Personal Uncertainty* exists

when uncertainty is felt, but it is realized that certainty can be reached by consulting a more knowledgeable source.

The meaning of terms is important in situations involving uncertainty. Vague words convey only a general meaning that is made more specific with the help of past experiences and the context in which they are used. When used properly and in the right context, vague terms have their benefits. However, vague terms can cause misunderstandings unless proper definitions are agreed upon. Also, it is often the case that different words are used to represent the same degree of belief. This can cause misinterpretations and in turn complicate the decision process.

One way of representing degrees of belief (other than words) is through a numerical representation (e.g. a percentage of confidence). However, numbers can be very subjective and their use are ill defined and often inconsistent. The alternative is to use a set of terms which are defined with a degree of belief (i.e. good, very good, etc.) Any observer's ability to make precise but significantly certain statements about complex matters decreases as the level of complexity increases. This seems to give the impression that precision and certainty seem to be incompatible. The use of precision may not be essential in all cases and in fact may NOT be wanted. The idea is to correctly represent the situation and not force fit the data to the model.

Knowledge Based Systems (KBS) or expert systems attempt to mimic the deductive and inductive decision process. Given this objective, these systems should operate in real-world situations and reason with uncertain information. The purpose of inexact reasoning in a KBS is to come up with the most likely conclusion or explanation on the basis of the available facts or evidence. Some domains have more uncertainty than others. Coping with this uncertainty in reasoning is a complex issue. These systems try to capture expertise for others to use. Expertise is usually based on experience rather than text-book facts. It may follow then that uncertainty may come from several sources including the expert, the data presented, or the problem itself. In developing a KBS, uncertainty can be mistakenly captured. Although an expert may be very good using his/her knowledge, there may be a problem in easily formulating or verbalizing the process of using the knowledge. In addition, once the thought process is *made* explicit, the consequences may conflict prior understanding. In some domains (where acknowledged experts are considered as such simply because they know more than others) experts may doubt their own ability and thus interject some degree of uncertainty. The data used or available may itself be incomplete and thus be vague or ambiguous.

In dealing with inexactness, there are two basic steps: determining the uncertainty of a basic set of events and combining the values obtained in the first step to arrive at the uncertainty value of compound or complex events. In developing a system that uses uncertainty handling, the system will almost never exhibit the desired results the first time. There then are three choices: change the uncertainty values supplied, change some rules in the system, or change the theory used for combining uncertainty values. In most cases, changing the underlying theory is not possible or practical. Thus, a system that uses uncertainty should allow experts to adjust uncertainties without difficulties and allow different methods of combining uncertainties for dealing with various problems or subproblems.

The most familiar form of inference is based on the two-valued Boolean logic. First-order predicate calculus, a generalization of Boolean logic, and modes ponens provide the language and rule of inference used in an overwhelming number of AI systems (especially forward chaining production rules). First-order logic has many shortcomings that uncertainty handling schemes have tried to address with varying degrees of success. Crisp values are not always known. Implications may not be crisp. Production rules can encourage incomplete specification of domain knowledge. Logic provides no simple way to draw tentative conclusions from incomplete information. Inherent modularity in production rules limits their utility for managing uncertain information.

Uncertain inference can be implemented in many different ways and as such many uncertainty management systems (UMS) have been proposed. Four basic, well-known UMSs are based on 1) Bayesian inference, 2) Certainty Factors, 3) Fuzzy Logic, 4) Dempster-Shafer evidential reasoning. The selection of the appropriate UMS for developing a particular application is often the key to the project's success. Using the wrong UMS may compromise the system's performance, effectiveness, robustness, and reliability. The key in using any UMS is that acceptable results must be seen.

BAYESIAN PROBABILITY

The Bayesian approach to probability relies on the concept that one should incorporate the *prior probability* of an event into the interpretation of a situation. Bayes' theorem provides a mathematical model where *prior beliefs* are combined with *evidence* to form estimates of uncertainty. In general a posterior belief is given as the likelihood of some event multiplied by the prior-belief. When a full specification is not available, some use approximate methods for completing the model - models that match common patterns of human reasoning.

The idea of using subjectivity with Bayes' theorem appears to violate the mathematical model. However a non-subjective approach makes it more difficult for use in expert systems since all relevant prior and conditional probabilities are required, all possible outcomes should be disjoint, and as a part of maintenance, it is very difficult to change one probability without causing a ripple effect that may incorrectly change other probabilities. However, one may view the Bayesian probabilities as simply another method of combining inexact values, rather than a probabilistic approach to uncertainty.

The term probability is usually used rather loosely in everyday conversation. Probabilities are a way of turning opinion or expectation into numbers. People often express uncertain information in terms of likelihood or odds. Probability is a measure of certainty between 0 and 1. The extreme values denote impossibility and certainty. Positive degrees of Belief indicate increasing probability and vice-versa. These are subjective probabilities. Probabilities also depend on whether associated events are mutually exclusive or not.

CERTAINTY FACTORS

Certainty factors (CFs) express or represent the strength of belief in the conclusion of a rule given that the premises are true. A CF can be regarded as updating the belief in a hypothesis based on evidence. A certainty factor is given by the difference between the degrees of belief and disbelief. Each fact or rule has a *degree of truth* or a *confidence factor*. CFs can also be regarded as approximate representations for subjective reasoning with facts or implications (or rules). They are also a means of representing conditional probabilities in a way such that non-experts can use them.

The definition and use of certainty factors began with the work of Buchanan and Shortliffe on the MYCIN (a medical diagnosis expert system) project at Stanford University. Most systems using CFs now are based on the original MYCIN work. The MYCIN model of inexact reasoning permits the co-existence of several plausible values for a single attribute with different degrees of belief.

Each fact and each rule has an associated CF. The CF interval [-1,1] indicates the certainty with which each fact or rule (hypothesis) is believed. Positive and negative CFs indicate a predominance of confirming or opposing evidence. Thus, the larger the CF the greater the degree of belief. A CF of 1 or -1 indicate absolute knowledge or confidence. On the other hand a CF of 0 indicates no evidence regrading the hypothesis.

In other words, the supporting evidence is equivalent to negative evidence. Often the default value is given as complete certainty (CF = 1).

NON-YES-NO parameters		YES-NO parameters	
KNOWN	CF > 0.2	KNOWN	|CF| > 0.2
NOTKNOWN	CF <= 0.2	NOTKNOWN	|CF| <= 0.2
DEFINITE	CF = 1	DEFINITE	|CF| = 1
NOTDEFININTE	CF < 1	NOTDEFININTE	|CF| < 1

HIGHER-LEVEL predicates

SAME	CF > 0.2
THOUGHTNOT	CF < -0.2
NOTSAME	CF <= 0.2
MIGHTBE	CF >= -0.2
VNOTKNOWN	|CF| <= 0.2
DEFIS	CF = +1
DEFNOT	CF = -1
NOTDEFIS	0.2 < CF < 1
NOTDEFNOT	-1 < CF < 0.2

FUZZY LOGIC

Lotfi Zadeh introduced the concept of Fuzzy Sets as an extension of classical (crisp) set theory to model vague facts. Alternative forms of logic emerge once binary, black and white distinctions are dismissed. Non-standard logics arise when the *uncertainty* of distinctions is accepted and definiteness in distinction is not available. In logic this occurs classically as the problem of the *borderline case*. Under uncertainty it may not be possible to make clear-cut assignments of truth or falsity to statements. In handling complex information, there must be a compromise between precise information and uncertainty.

It has been shown that people naturally form fuzzy sets when categorizing objects. While people comprehend vague concepts as if the concepts are represented internally as fuzzy sets, they do not always manipulate these concepts in the same fashion. People reason with words NOT numbers. Everyone uses "fuzzy" words. Also, most natural concepts in the world are not crisp, they are fuzzy. The question as to whether a concept is fuzzy or not may be resolved by examining the applicability of a simple modifier such as *very* to the concept in question. If the the concept makes sense with the addition of the modifier, then it is *fuzzy*. Fuzziness occurs when an interval has no sharp interval.

DEMPSTER-SHAFER THEORY of EVIDENCE

The Dempster-Shafer (D-S) theory of evidence was first set forth in 1960s by Dempster and extended by Glen Shafer in 1976. The fundamental difference between the Dempster-Shafer calculus and the probabilistic approach is the relaxation of the constraint that P(x) + P(x') = 1. The sum of the basic probabilities over all possible variable values must equal 1. Reasoning is done with measures of belief rather than basic probabilities. Measures of belief are combined according to Dempster's rule of combination. Dempster-Shafer reasoning provides a mechanism for reasoning about plausibility and belief separately. The D-S approach uses belief functions and plausibility functions to attach numerical lower and upper bounds on the likelihoods of events. This allows the uncertainty management systems to perform more robustly in complex situations.

Dempster-Shafer theory accepts an incomplete probabilistic model when some parameters are missing. The probabilistic information that is available (like the strength of evidence) is interpreted not as likelihood ratios, but rather as random events that impose truth values to various propositions for a certain fraction of time. The theory does not pretend to provide full answers to probabilistic queries, but rather assigns itself to providing partial answers. It estimates how close the evidence is to forcing the truth of the hypothesis instead of estimating how close the hypothesis is to being true.

SUMMARY

There is no single accepted method of handling uncertainty and researchers should be aware of their goals when tackling a project. The domain, human reasoning process, type of uncertainty, and available UMSs must be considered before making a decision on what UMS to use. One must be careful since the use of a KBS in a practical environment poses practical, legal, and moral issues. Somebody must take responsibility for actions taken based on a KBS. This involves the handling of errors, maintenance, and so forth. Uncertainty in the domain can only compound existing problems.

Uncertainty causes problems in at least two main aspects of KBS development: during knowledge elicitation and knowledge representation. No knowledge acquisition (KA) process can eliminate uncertainty, but techniques can be used to minimize it. The source of uncertainty is not always clear when gathering knowledge. Conflict resolution is handled rather easily by the method being used. The exception is when more than one method is being used with the same data. The

technique of attaching beliefs to rules and evidence and how to combine beliefs are major research issues.

Most knowledge is "probable" rather than certain and attempts to encapsulate such knowledge can meet with severe difficulties during elicitation and subsequent implementation. The acquisition of a numeric value associated with knowledge can be a tedious process unless the information can be extracted from available data. The advantage of using uncertainty methods is the established calculations for combining the certainty values. The problem is in determining the contextual meaning of the numbers. Another school of thought believes that human decision analysis is essentially linguistic and the notion is often overlooked when a technical discussion of mathematical systems takes place and thus the use of linguistic terms in uncertainty handling is often dismissed. The exception appears to be the use of fuzzy sets, fuzzy logic, and linguistic variables.

Psychological research has revealed that human performance in the face of uncertainty is spotty at best. This is the case of novices and experts alike. It has been shown that human estimates tend to be heavily on the conservative side. Several explanations have been offered for this phenomenon. They relate to bias through inappropriate hypothesis formation, mis-assessment of subjective probabilities, misapplication of prior odds inappropriate or incomplete assessment of likelihood ratios, mis-aggregation (calculation error), incomplete search for evidence, or mis-interpretation of analysis.

There are many things that can affect the estimates made by humans in uncertain situations, thus they can be considered subjective probabilities. Often frequent events are underestimated and infrequent events are overestimated. Experts typically do not have predetermined numbers representing the uncertainty of a particular fact or rule at hand. Instead, they guess or estimate such numbers in some way. In these situations, there are a number of factors that may affect the accuracy of their judgement. People will generally assign a higher probability or certainty to information that is easy to remember. The best thing for the knowledge engineer (KE) to do is to avoid using probabilistic or statistical judgments by the domain expert as much as possible. The goal is to minimize mis-calibrations.

It is often harder to quantify one's thoughts and impressions alone than when assisted by another person. In such cases, the expert may benefit from trying to explain ideas to someone else. When knowledge engineers deal with experts, one approach to deal with cognitive biases is to use carefully worded questions that will help the expert think about the implications of an answer. Another useful approach is to

force the expert to compare different uncertainties against each other. Other approaches involve the use of visual or graphical representations. Still another method of helping experts better express uncertainty is *ranking* (selection, insertion, exchanges). This approach requires the expert to order a set of facts in terms of their uncertainty.

Even after a KE enters knowledge into a KBS with the help of an expert, the use of the system can still be hampered by uncertainty. An end-user may not use the same "certainty factors" or "measures of belief" that another user or expert may use. Also, the user may not use certainty values in a consistent manner. So, in affect, even though knowledge is entered into a system in a consistent, cohesive way to handle uncertain situations, the knowledge may not be used the same way in the real world. As a result, information may not be combined in the correct way and 'execution' paths may go astray.

The search for better tools for decision analysis is unending. Inductive methods of "generating" knowledge is ineffective in handling uncertainty. The lack of an adequate formalism for modeling uncertainty in KBSs is an identified obstacle to progress of not only KBS methods, but artificial intelligence in general. In fact, many application developers avoid using uncertainty in KBSs for several reasons. Mathematical details may be intimidating. Some domains are not driven by uncertainty. Confusion ranges between reasoning with uncertain knowledge and providing uncertain conclusions. In fact, some very useful systems have been developed without complex inference mechanisms and models of uncertainty.

APPENDIX I

GLOSSARY

Demons - are a program or routine that is waiting to execute at anytime that some previously defined condition becomes true (or in some cases false).

Layouts - are verbal descriptions that show how the knowledge provider conceives of tasks and how current information is organized in light of prior knowledge and the present context. Each one helps make sense of the facts and heuristics used in the task by specifying the goals for which they are to be used. Layouts provide a characterization of the task in terms of its boundaries, organization, and basic classifications.

Metaphors - describe one thing by reference to another dissimilar thing so that the first is understood more completely than if the comparison had not been made.

Rating Grids - are grids in which solutions or elements are represented on one vertex of a table and traits or constructs of these solutions are represented on the other vertex and each possibility is given a rating on a trait scale.

Repertory - is taken from repertoire (as in repertoire of constructs) [BANNISTER68].

Rules-of-thumb - are concrete, implementable strategies of minor to moderate scope which can identify and define issues meeting conditions which will serve as the most complete strategy.

Scripts - give the knowledge provider's sequential and procedural knowledge of the domain. Learning an expert's scripts provides a temporal chart of the actions and enables understanding of each action in terms of the prior knowledge required to perform it.

Stories - are accounts of previous experience, often case studies or talk-aloud protocols, which can be oriented in different ways.

BIBLIOGRAPHY

[ABRETT88] Abrett, H. and Burstein, M. H.(1988). "The KREME Knowledge Editing Environment"; Knowledge Acquisition Tools for Expert Systems; Knowledge-Based Systems Volume 2; pp.1-24; Academic Press

[ALPAR90] Alpar, P. (1990); "Toward Structured Expert Systems Development"; Expert Systems with Applications; Volume 1; Number 1; pp 63-70; Pergamon Press

[ALPERIN87] Alperin, L.B. & Kedzierski, B.I.(1987). "AI-Based Software Maintenance", COMPUTER, IEEE Computer Society, IEEE, March 1987, pp. 321-326

[BAHRAMI88] Baharami, A. (1988); Designing Artificial Intelligence Based Software; pp 263-271; Sigma Press

[BANNISTER68] Bannister, D. & Mair, J.M.M. (1968), The Evaluation of Personal Constructs; pp 209-216; Academic Press

[BANNISTER70] Bannister, D. (1970), "A Brief Introduction to Personal Construct Theory", Perspectives in Personal Construct Theory; Bannister,D.(ed); pp 47-62; Academic Press

[BANNISTER71] Bannister, D. & Fransella, F (1971), Inquiring Man: The Theory of Personal Constructs; Penguin Education

[BELL81] Bell, R. C. & Keen, T. R. (1981), "A Satistical Aid for the Grid Administrator", Recent Advances in Personal Construct Technology; pp 209-216; Academic Press

[BELL89] Bell, J. & Hardiman, R.J. (1989); "The Third Role - The Naturlistic Knowledge Engineer"; Knowledge Elicitation: Principles, Techniques, and Applications; Diaper (ed); pp 49-85; Ellis Horwood Limited

[BENFER89] Benfer, R. A. & Furbee, L. (1989), "Knowledge Acquisition in the Peruvian Andes", AI EXPERT; November 1989; pp. 22-29; Miller Freeman Publications

[BENNETT85] Bennett, J. S. (1985); "ROGET: A Knowledge-Based System for Acquiring the Conceptual Structure of a Diagnostic Expert System"; Journal of Automated Reasoning; Volume 1, 1985; pp 49-74; D. Reidel Publishing Company

[BOOSE85] Boose, J. H. (1985) "A Knowledge Acquisition Program for Expert Systems Based on Personal Construct Psychology"; International Journal of Man-Machine Studies; Volume 23, Number 5, pp 495-525; Academic Press

[BOOSE86] Boose, J.H. (1987); Expertise Transfer for Expert System Design; Artificial Intelligence; Elsevier

[BOOSE87] Boose, J.H. & Bradshaw, J.M. (1987), "AQUINAS: A Knowledge Acquisition Workbench for Building Knowledge-Based Systems"; Proceedings of the Second AAAI Knowledge Acquisition for Knowledge Based Systems Workshop; pp. 6.1-6.6

[BOOSE88] Boose, J. H. (1988) "Uses of Repertory Gid-Centered Knowledge Acquisition Tools for Knowledge-Based Systems"; International Journal of Man-Machine Studies; Volume 29, Number 3, pp 287-310; Academic Press

[BOOSE88a] Boose, J. and Gaines, B. (1988), Knowledge Acquisition Tools for Expert Systems; Knowledge-Based Systems Volume 2; pp. vii-xvi; Academic Press

[BOOSE88b] Boose, J.H. & Bradshaw, J.M.(1988), "Expertise Transfer and Complex Problems: Using AQUINAS as a Knowledge-Acquisition Workbench for Knowledge-Based Systems"; Knowledge Acquisition Tools for Expert Systems; Knowledge-Based Systems Volume 2; pp.39-64; Academic Press

[BOOSE89] Boose, J. & Shaw, M. (1989), "Knowledge Acquisition for Knowledge-Based Systems", a tutorial from the Eleventh International Joint Conference on Artificial Intelligence, Detroit Michigan, August 1989

[BREUKER87] Breuker, J. & Wielinga, B. (1987), "Use of Models in the Interpretation of Verbal Data"; Knowledge Acquisition for Expert Systems; Kidd, A.L. (ed) pp 17-44; Plenum Press

[CARR89] Carr, C. (1989). "Spreadsheet Sketches for Rule-Based Systems"; AI EXPERT; November 1989; pp. 30-34; Miller Freeman Publications;

[CHARNIAK85] Charniak, E. & McDermott D. (1985), Introduction to Artificial Intelligence; pp. 453-484,609-662; Addison-Wesley

[CHECKLAND81] Checkland, P. (1981), Systems Thinking, Systems Practice; pp 189-191; John Wiley and Sons

[CORDINGLEY89] Cordingley, E. S. (1989); "Knowledge Elicitation Techniques for Knowledge-Based Systems"; Knowledge Elicitation: Principles, Techniques, and Applications; Diaper (ed); pp 89-175; Ellis Horwood Limited

[DAVIS82] Davis, R. & Lenat, D. B. (1982); Knowledge-Based Systems in Artificial Intelligence; pp 227-285; McGraw-Hill

[DIEDERICH88] Diederich,J., Ruhmann, I. & May (1988), "KRITON: A Knowledge-Acquisition Tool for Expert Systems"; Knowledge Acquisition Tools for Expert Systems: Knowledge-Based Systems Volume 2; pp. 83-94; Academic Press

[EASTERBY-SMITH81] Easterby-Smith, M. (1981), "The Design, Analysis and Interpretation of Repertory Grids", Recent Advances in Personal Construct Technology; pp 9-30; Academic Press

[ESHRAGH81] Eshragh, F. (1981), "Subjective Multi-Criteria Decision Making", Recent Advances in Construct Technology; pp 183-208; Academic Press

[EVANSON88] Evanson, S. E. (1988). "How to Talk to an Expert"; AI EXPERT; February 1988; pp. 36-42; Miller Freeman Publications

[FRANSELLA77] Fransella, F. & Bannister, D. (1977), A Manual for Repertory Grid Technique; pp 1-119; Academic Press

[GAINES81] Gaines, B.R. & Shaw, M.L.G. (1981), "New Directions in the Analysis and Interactive Elicitation of Personal Construct Systems", Recent Advances in Construct Technology; pp 147-182; Academic Press

[GAINES88] Gaines, B.R. (1988). "Knowledge Acquisition: Developments and Advances"; Expert Systems and Intelligent Manufacturing; pp. 410-434; Elsevier Science Publishing

[GAMMACK87] Gammack, J. G. (1987), "Different Techniques and Different Aspects on Declarative Knowledge"; Knowledge Acquisition for Expert Systems; Kidd, A.L. (ed) pp 137-163; Plenum Press

[GARG-JANARDAN88] Garg-Janardan, C. & Salvendy G. (1988), "A Conceptual Framework for Knowledge Elicitation"; Knowledge Acquisition Tools for Expert Systems; Knowledge-Based Systems Volume 2; pp. 119-130; Academic Press

[GEISSMAN88] Geissman, J.R. & Schultz, R.D. (1988), "Verification and Validation of Expert Systems", AI EXPERT; February 1988; pp. 26-33; Miller Freeman Publications

[GORDON88] Gordon, S.E. (1988). "The Human Factor in Expert Systems", AI EXPERT; January 1988; pp. 55-59; Miller Freeman Publications

[GRUBER87] Gruber, T. & Cohen, P. (1987), "Principles of Design for Knowledge Acquisition", COMPUTER; IEEE Computer Society; IEEE, March 1987; pp 9-15

[GRUBER89] Gruber, T. R. (1989), The Acquisition of Strategic Knowledge; Academic Press

[HARRI-AUGSTEIN85] Harri-Augstein, S. (1985), "Learning-to-Learn Languages: New Perspectives for the Personal Observer"; Issues and Approaches in Personal Construct Theory; (ed) Bannister, D. (1985); pp 47-66; Academic Press

[HART86] Hart, A. (1986), Knowledge Acquisition for Expert Systems; McGraw Hill;

[HINKLE70] Hinkle, D. (1970), "The Game of Personal Constucts", Perspectives in Personal Construct Theory; Bannister, D. (ed); pp 91-110; Academic Press

[KEEN81] Keen, T. R. & Bell, R. C. (1981), "One Thing Leads to Another: A New Approach to Elicitation in the Repertory Grid Technique", Recent Advances in Personal Construct Technology; pp 81-94; Academic Press

[KELLY55] Kelly, G.A. (1955), The Psychology of Personal Constructs: Volume One - A Theory of Personality; pp 46-318; W.W Norton and Company

[KELLY70] Kelly, G. (1970), "A Brief Introduction to Personal Construct Theory", Perspectives in Personal Construct Theory; Bannister, D(ed); pp 1-30; Academic Press

[KELLY77] Kelly, G. (1977), "The Psychology of the Unknown", New Perspectives in Personal Construct Theory; Bannister, D. (ed); pp 1-20; Academic Press

[KIDD87] Kidd, A. L. (1987), "Knowledge Acquisition - An Introductory Framework"; Knowledge Acquisition for Expert Systems; Kidd, A.L.(ed); pp 1-16; Plenum Press

[KITTO87] Kitto, C. M. (1987); "Knowledge Acquisition Tools for Different Problem-Solving Paradigms: Research at Boeing Computer Services"; Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems; (ed) Slavendy, G. (1987); pp 515-522; Elsevier Science Publishers

[KITTO89] Kitto, C. M. & Boose, J H. (1989); "Selecting Knowledge Acquisition Tools and Strategies based on Application Characteristics"; International Journal of Man-Machine Studies; Volume 31, Number 2; pp. 149-160; Academic Press

[LEACH81] Leach, C. (1981), "Direct Analysis of a Repertory Grid", Recent Advances in Personal Construct Technology; pp 217-232; Academic Press

[MAIR70] Mair, J.M.M. (1970), "Psychologists are Human Too", Perspectives in Personal Construct Theory; Bannister, D. (ed); pp 157-184; Academic Press

[MATTHEWS87] Matthews, M.H. (1987). "Maintenance and Language Choice", AI EXPERT; September 1987; pp. 42-48; Miller Freeman Publications

[MOORE88] Moore, E.A. & Agogino, A.M. (1988). "INFORM: An Architecture for Expert-Directed Knowledge Acquisition"; Knowledge Acquisition Tools for Expert Systems; Knowledge-Based Systems Volume 2, pp. 227-244; Academic Press

[MORIK88] Morik, K. (1988). "Acquiring Domain Models"; Knowledge Acquisition Tools for Expert Systems; Knowledge-Based Systems Volume 2, pp. 245-256; Academic Press

[MUSEN89] Musen,M.A. (1989); Automated Generation of Model-Based Knowledge-Acquisition Tools; pp. 1-79; Morgan Kaufmann Publishers

[NEIMEYER85] Neimeyer, R.A. (1985); "Problems and Prospects in Personal Construct Theory"; Issues and Approaches in Personal Construct Theory; (ed) Bannister, D. (1985); pp 143-172; Academic Press

[PARSAYE88] Parsaye, K. (1988). "Acquiring and Verifying Knowledge Automatically", AI EXPERT; May 1988; pp. 48-63; Miller Freeman Publications

[RINGLAND88] Ringland, G.A. & Duce D.A. (1988), Approaches to Knowledge Representation: An Introduction; pp 263-271; Research Studies Press/John Wiley and Sons

[SCHUTZER90] Schutzer, D. (1990); "Business Expert Systems: The Competitive Edge"; Expert Systems with Applications"; Volume 1; Number 1; pp 17-21; Pergamon Press

[SHACHTER87] Shachter, R. D. & Heckerman, D. E. (1987); "Thinking Backward for Knowledge Acquisition", AI Magazine; Fall 1987; Volume 8, Number 3; pp 55-61; American Association for Artificial Intelligence

[SHAW80] Shaw, M.L.G. (1980), On Becoming A Personal Scientist: Interactive Computer Elicitation of Personal Models of the World; pp 147-163; Academic Press

[SHAW81] Shaw, M.L.G. (1981); "Conversational Heuristics for Eliciting Shared Understanding", Recent Advances in Personal Construct Technology; pp 31-44; Academic Press

[SHAW82] Shaw, M. L. G. (1982) "PLANET: Some Experience in Creating an Integrated System for Repertory Grid Applications on a Microcomputer"; International Journal of Man-Machine Studies; Volume 17, Number 3; pp 345-360; Academic Press

[SHAW86a] Shaw, M.L.G. & Gaines B.R. (1986), "Knowledge Engineering Tools for Expert Systems"; Computer Assisted Decision Making; pp 147-163; Elsevier Science Publishers

[SHAW86b] Shaw, M.L.G. & Gaines, B.R (1986); "Interactive Elicitation of Knowledge from Experts", Future Computing Systems; Volume 1, Number 2, 1986; pp 151-189; Oxford University Press & Maruzen Company Limited

[SHAW87] Shaw, M.L.G. & Gaines, B. R. (1987), "An Interactive Knowledge-Elicitation Technique Using Personal Construct Technology"; Knowledge Acquisition for Expert Systems; Kidd, A.L.(ed); pp 109-136; Plenum Press

[SHEMA88] Shema, D. B. & Boose, J. H. (1988), "Refining Problem-Solving Knowledge in Repertory Grids Using a Consultation Mechanism"; International Journal of Man-Machine Studies; Volume 29, Number 4; pp 447-460; Academic Press

[SOWA84] Sowa, J.F.(1984); Conceptual Structures: Information Processing in Mind and Machine; pp 318-328; Addison-Wesley

[STEFAN77] Stefan, C. (1977), "Core Structure Theory and Implications", New Perspectives in Personal Construct Theory; Bannister, D. (ed); pp 281-298; Academic Press

[STEWART81] Stewart, V. & Stewart, A. (1981), Business Applications of Repertory Grid; pp 3-71; McGraw-Hill

[STRINGER77] Stringer, P. (1977), "Participating in Personal Construct Theory", New Perspectives in Personal Construct Theory; Bannister, D. (ed); pp 299-320; Academic Press

[WATANABE89] Wantanabe, M. & Yamanouchi, T. & Iwamoto, M. & Ushioda, Y. (1989), "CL: A Flexible and Efficient Tool for Constructing Knowledge-Based Expert Systems"; IEEE Expert; Fall 1989; pp 41-50; IEEE

[WATERMAN86] Waterman, D.A. (1986). A Guide to Expert Systems; Addison Wesley

[WILSON89] Wilson, M. (1989); "Task Models for Knowledge Elicitation"; Knowledge Elicitation: Principles, Techniques, and Applications; Diaper (ed); pp 197-219; Ellis Horwood Limited