

# Open Research Online

---

The Open University's repository of research publications and other research outputs

## Automatic Acquisition Of Knowledge For Solving Analysis Tasks

### Thesis

How to cite:

Patel, Jitendra M (1990). Automatic Acquisition Of Knowledge For Solving Analysis Tasks. PhD thesis. The Open University.

For guidance on citations see [FAQs](#).

© 1989 The Author

Version: Version of Record

---

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's [data policy](#) on reuse of materials please consult the policies page.

---

[oro.open.ac.uk](http://oro.open.ac.uk)

DX 89116  
UNRESTRICTED

**AUTOMATIC ACQUISITION OF KNOWLEDGE  
FOR SOLVING ANALYSIS TASKS**

**JITENDRA M. PATEL**

**THESIS SUBMITTED IN FULFILMENT  
OF THE REQUIREMENTS FOR A  
PHD IN ARTIFICIAL INTELLIGENCE**

**OCTOBER 1989**

**HUMAN COGNITION RESEARCH LABORATORY  
THE OPEN UNIVERSITY  
MILTON KEYNES  
MK7 6AA  
UK**

Date of submission: 30 October 1989

Date of award: 29 January 1990

ProQuest Number: 27758423

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent on the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 27758423

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All Rights Reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 - 1346

## **ABSTRACT**

The extraction of knowledge from domain experts for the purpose of building expert systems has been found to be a difficult and a time-consuming enterprise. Researchers have therefore looked at the possibility of automating this task of knowledge acquisition. The existing knowledge acquisition systems rely on either some task-specific knowledge or general techniques for knowledge elicitation for obtaining problem-solving expertise from domain experts. Generally, systems which employ task-specific knowledge produce better problem-solving expertise than those systems which use elicitation techniques. However, the scope of applicability of former systems is relatively narrow. This is the central problem addressed in this thesis: to design a knowledge acquisition system which can be applied over a wide range of tasks with the purpose of acquiring useful problem-solving knowledge.

The thesis presents a methodology and a system for knowledge acquisition called ASKE. The methodology prescribes that knowledge acquisition should start by defining the task and then use the developed task-model to acquire domain specific knowledge. ASKE is able to support this process by allowing the user to construct task-models and by being able to effectively use them for acquiring domain expertise. The advantage of progressing in this manner is two-fold: firstly, it widens the scope of applicability of the knowledge acquisition system; and, secondly, it makes possible the construction of knowledge-bases that exhibit expert performance.

ASKE contains knowledge engineering expertise which it uses to help domain experts encode their problem-solving expertise directly into a knowledge-base. The system derives its power from the templates which encode knowledge. The templates serve a triple function: [1] they represent knowledge - their normal function; [2] they encode expectations of the kind of knowledge that is to be acquired; and, [3] they serve as a guide for how problem-solving knowledge may be organized so as to facilitate its encapsulation into a knowledge-base.

# CONTENTS

Preface	i
Acknowledgements	ii
List of Figures	iii
Chapter I - Introduction	
1.1 Knowledge is Power	1
1.2 The "Bottleneck"	3
1.3 Aim of this thesis	4
1.4 Previous Works	6
1.5 The System	8
1.6 Scenario	11
1.7 Structure of this thesis	16
Chapter II - Expert Systems and Knowledge Engineering	
2.1 What is an Expert System?	17
2.2 The Knowledge Engineering Process	22
2.3 Knowledge Engineering Methodologies	27
2.4 Problems in Knowledge Engineering	30
2.5 Automation of the Knowledge Engineering Process	34
Chapter III - Knowledge Acquisition	
3.1 Introduction	36
3.2 A Classification of Knowledge Engineering Tools	37
3.3 A Review of Knowledge Acquisition Systems	44
3.4 A Model for Knowledge Acquisition	48
Chapter IV - Fundamental Concepts	
4.1 Introduction	51
4.2 Task characteristics as task-dependent strategies	52
4.3 Previous Cases as Exemplars	59
4.4 Template scheme for representing knowledge	64
4.5 Summary	73
Chapter V - The ASKE System	
5.1 Introduction	74

5.2	ASKE Methodology	75
5.3	Overview of ASKE	77
5.4	Notebook	83
5.5	Other Features	86
5.6	Rulemaker	93
Chapter VI - Processing in ASKE		
6.1	Introduction	97
6.2	Task Characterization	99
6.3	Obtaining the Task Model	110
6.4	Knowledge Elicitation	117
6.5	Rules Editing	130
6.6	Creating an RTEMP	135
Chapter VII - Discussion		
7.1	Introduction	137
7.2	Template driven Knowledge Acquisition	138
7.3	Knowledge Acquisition is a modelling activity	141
7.4	Heuristic Classification Paradigm	143
7.5	Intelligent Interfaces	146
Chapter VIII - Conclusion		
8.1	Recap	149
8.2	Future Research	152
References		155
Appendices		
A	ASKE generated rules for Settlement.Sites	165
B	Final output for Settlement.Sites	170
C	Knowledge Acquisition in the domain of Nursing	173

## **PREFACE**

Parts of Chapters I-VI have been published in Patel 1988, 1989a and 1989b.

The quotes at the start of each Chapter are from Feigenbaum and McCorduck (1984).

## **ACKNOWLEDGEMENTS**

- Marc Eisenstadt supervised this thesis. He was also responsible for getting me a studentship from the Open University which made it possible for me to pursue the research presented in this thesis.
- Arthur Stutt and Mike Brayshaw for many useful discussions on ASKE, Archaeology and cricket. Mike provided useful comments on an earlier draft of this thesis.
- Enrico Motta and Tim Rajan for their critical comments on ASKE and an earlier draft of this thesis.
- Other members of HCRL.
- Steve Shennan and the Department of Archaeology, University of Southampton for making available their facilities so that I could gain a better understanding of the field of archaeology.
- Andy Boddington, Adrian Hopgood, Jenny Jones and Tony Willoughby. Their domain expertise was valuable in the development of the methodology for knowledge acquisition.
- My parents for giving me a free rein on my destiny.

---

This research was supported by an Open University Studentship. The research was carried out using KEE on an Explorer 1 which were given to the Human Cognition Research Laboratory by Unisys (UK). Both are gratefully acknowledged.

---



## LIST OF FIGURES

1-1	Template types	9
1-2	A two stage model of knowledge acquisition	11
2-1	The structure of an expert system	19
2-2	The knowledge engineering process	22
3-1	Overview of knowledge engineering	36
3-2	Classification of knowledge engineering tools	37
3-3	A model for bridging the breadth-depth problem	48
3-4	Merging the KNACK and PROTEGE systems	49
3-5	ASKE's model for knowledge acquisition	50
4-1	Heuristic classification	52
4-2	Task characteristics for selection	54
4-3	Task characteristics for debugging	56
4-4	Task characteristics for diagnosis	56
4-5	Task characteristics for diagnosis and debugging	57
4-6	Task characteristics for interpretation	58
4-7	ROGET dialogue showing use of previous cases	61
4-8	ASKE session showing use of previous cases	63
4-9	Structure of a template	64
4-10	General template	66
4-11	Concept categories for analysis tasks	67
4-12	Acquisition template	68
4-13	Reference template	70
4-14	Working template	71
5-1	Stages in the development of an application with ASKE	75
5-2	Overview of ASKE	78
5-3	The Aske Interface	80
5-4	The Rulemaker Interface	81

5-5	Central Concepts Window	85
5-6	Sketch-Pad	87
5-7	Moving attributes	90
5-8	Relations Window	92
5-9	Editor Window in Rulemaker	96
6-1	Templates for Archaeology	98
6-2	Inferencing in ASKE	99
6-3	Obtaining personal details	101
6-4	Identifying the domain	102
6-5	Identifying the task type	104
6-6	Identifying the area of specialization	105
6-7	Defining the project goals	107
6-8	Selecting an RTEMP	109
6-9	What is required in a task model	112
6-10	An example of a task model	113
6-11	Obtaining the data categories	114
6-12	Obtaining the solution categories	115
6-13	The main concept categories	116
6-14	Entering concepts and attributes	120
6-15	Editing concept attributes	122
6-16	Adding a new attribute	123
6-17	Types of relationships	124
6-18	Same category concepts	127
6-19	Informing about unaccounted concepts	129
6-20	Merging rules	133
6-21	Editing the premise	134
6-22	The new RTEMP	136
7-1	Linking tasks to heuristic classification	144
7-2	How ASKE differs from ETS and MOLE	145

*Alice said, "Would you tell me, please, which way I ought to go from here?"  
"That depends a good deal on where you want to get to," said the Cat. "I don't know where...," said Alice. "Then it doesn't matter which way you go," said the Cat.*

*From Lewis Carroll's  
Alice's Adventures in Wonderland*

*"The ultimate design goal for knowledge acquisition is to allow the expert to encode his own knowledge directly into the computer, removing the role of the knowledge engineer from the knowledge acquisition phase." (Weiss & Kulikowski, 1984:62)*

# Chapter 1

## INTRODUCTION

*Excellent performance requires knowing so much.*

### 1.1 Knowledge is Power

The foundations of the new science of Artificial Intelligence (AI) were laid in the summer of 1956 at the Dartmouth Conference. Ten scientists, who were to lead this endeavour, convened to draft the directions that AI was to follow. The central goal of this new field was to develop "smart" computer systems. In other words, the AI systems were expected to solve complex problems, which, if solved by humans, would be characterized as intelligent behaviour (Barr and Feigenbaum, 1981).

The initial work in AI, until about the mid 60's, was guided by the basic tenet that intelligent behaviour is based on clever reasoning. The research efforts were thus aimed at discovering smart problem-solving techniques, for example, finding general methods that could solve broad classes of problems. These problem-solving methods were then implemented in general-purpose programs such as GPS (Newell and Simon, 1972). The general-purpose algorithms were however found to be insufficient, on their own, in solving complex problems. Furthermore, the programs showed generality-performance tradeoffs: the more classes of problems a single program could handle, the more poorly it seemed to perform on any individual problem.

While the early research produced no breakthroughs, it pointed out the possible ingredients for producing intelligent systems. Besides general purpose techniques, the programs had to have more efficient ways for storing data and for searching the space of possible solutions. So, during the 70's, AI scientists concentrated on the techniques of representation (e.g., Minsky, 1975) - how to formulate the problem so that it would be easy to solve - and search (e.g., Knuth and Moore, 1975) - how to cleverly control the search for a solution so it wouldn't take too long or use too much of the computer's memory capacity. Once again, the research failed to produce the expected results. The new techniques, supported by greater computing power, were not enough to solve real-world problems.

Though initial work started in mid 60's (e.g., Buchanan et al., 1969), it wasn't until the late 70's<sup>1</sup> that AI scientists accepted the fact that the problem-solving power of a program comes as much from the knowledge it possesses as from the formalisms and inference schemes it employs. The conceptual breakthrough was made and can be quite simply stated: *intelligence comes with knowledge.*

The realization that knowledge is the key to intelligence has led to the subfield of *Knowledge Engineering* associated with the building of *Expert Systems*, which are computer programs designed to capture and utilize the expertise of a human expert in a narrow domain (such as computer configuration, medical diagnosis, signal interpretation, and weather forecasting). In the 80's, expert systems have proliferated and AI has gone commercial. With the success of this new breed of computer systems, knowledge engineering has been recognized as an important field of research. The new motto guiding the work in this area is "knowledge is power" (Feigenbaum and McCorduck, 1983). It changes the emphasis from

---

<sup>1</sup> Knowledge based systems were for the first time introduced at the 5th Joint International Conference on Artificial Intelligence held in Cambridge, Massachusetts, in 1977.

a search for general mechanisms of intelligence to the development of techniques for knowledge acquisition and representation.

## 1.2 The "Bottleneck"

The DENDRAL project was the first AI enterprise which broke with the early tradition and identified the need for large volumes of special-purpose knowledge to permit programs to work effectively in real-world domains. The acquisition of the problem-solving knowledge is however a non-trivial task. For instance, from his experience of building the DENDRAL system, Buchanan (1969:256) writes:

".. one of the greatest bottlenecks in our total system of chemists, programmers and program has been eliciting and programming new pieces of information about mass spectrometry."

For effective performance, an expert system requires substantial domain-specific knowledge, the main source of which is a human expert. The process of transferring the problem-solving knowledge from a domain expert to a computer program is called *Knowledge Acquisition* (KA). Feigenbaum, a pioneer in expert systems, argues that KA is a major obstacle in the development of expert systems:

"... the power to enhance or amplify the performance of AI programs resides in the specific knowledge of the problem domain that can be brought to bear. This knowledge is currently acquired in a very painstaking way; individual computer scientists work with individual experts to explicate the experts' heuristics - to mine those jewels of knowledge out of their heads one by one. ... Right now the problem of knowledge acquisition is the critical bottleneck in artificial intelligence." (Feigenbaum and McCorduck, 1983:107)

The KA problem reflects, in part, an inability to achieve a direct interaction between the domain expert and knowledge-based system during the process of system development. Typically, a knowledge

engineer acts as an intermediary, translating the domain expert's knowledge into appropriate data structures that can be processed by computer. This requires an expert to explain domain concepts to a knowledge engineer, and to describe explicitly a decision-making process of which s/he may not normally be conscious. Consequently, the translation of knowledge into a form suitable for computer processing has proven costly and inefficient, thereby impeding the production and dissemination of functional systems.

### 1.3 Aim of this thesis

We need to remove the bottleneck. According to Buchanan (1969), there are three approaches open to us.

- 1 Educate the knowledge engineer in the domain of application.
- 2 Educate the domain expert in programming.
- 3 Replace the knowledge engineer with a program designed to perform KA to the same level as him/her.

It must be pointed out that there is also a fourth approach, with a lot of research interest, which aims at developing tools, which provide automated help to the knowledge engineer at various stages of expert system development.

While the first two approaches are theoretically feasible, they are not very practical. For example, the first one suggests that the knowledge engineer become an expert which takes years of learning and experience. A characteristic of most human experts is that they do not have a lot of time to spare. So asking them to learn to program, as the second approach requires, may not be such a good solution of the problem.

The third approach has the potential of minimising, if not removing, the bottleneck. Indeed, it is one of the dreams of the expert-system



community: to have knowledge-bases created and maintained by the domain experts themselves rather than by knowledge engineers. Basically, the approach involves developing computer programs which, without the intervention of the knowledge engineer, could interact with domain experts to develop expert systems. In other words, the research aims at developing an expert system which contain the expertise of the knowledge engineer.

This thesis takes the third approach and is a contribution to the field of research with the ultimate aim of removing the knowledge engineer from the expert system building cycle. The aim of this thesis is to present an automatic system for KA called ASKE (Automatic System for Knowledge Engineering). ASKE has been designed with the objective of providing:

- 1] a system which can be used by domain experts to encode their problem-solving expertise; and,
- 2] a system which can be used for developing prototype systems for analysis tasks, i.e. any application task for which solutions can be enumerated.

The development of a system which can be used by humans is indeed an ambitious project. My aim, in this thesis, is not to address the human-computer interface issues per se, but to provide a methodology and a tool-kit for automatic KA.

In the remainder of this Chapter, I will briefly describe the ASKE system. I will start with a look at previous works with the aim of locating the research presented in this thesis. The methodology and the general architecture of ASKE is presented next. This is followed by a scenario of how ASKE develops prototype systems.

## 1.4 Previous Works

Research towards the removal of the KA bottleneck is concentrated mainly in two directions, corresponding to the third and the fourth approaches above, and involves the development of: [1] tools, called KA systems, which interactively acquire knowledge directly from the domain experts; and [2] tools which assist the knowledge engineers at various stages of KA. The second trend does not bear directly on the research presented in this thesis so I will confine the discussion to the mention of two of its more well-known exemplars. This will be followed by a brief introduction to Clancey's (1985) model of heuristic classification which has strongly influenced research in KA systems. Finally, I will describe the main ideas underlying the research in KA systems, and show where the current work fits in.

The second research trend involves providing assistance to the knowledge engineer in the form of a KA methodology and some automated support during knowledge-base development. Two of the most influential works in this area are the KADS (Breuker and Wielinga, 1985) and the KEATS (Motta, et al., 1989a) methodologies for KA. The KADS methodology contains descriptions of techniques for data collection and data analysis some of which are supported by tools in the KADS system (Anjewierden, 1987). The KEATS methodology is imbedded in the KEATS toolkit, which provides semi-automated assistance at all stages of knowledge-base development.

### 1.4.1 The Heuristic Classification Model

Clancey (1985) proposed a model of problem-solving which has been extensively used by researchers in KA. The model provides a precise set of terms and relations by which problem-solving tasks can be characterized. According to Clancey, there are two main types of problems: [1] those whose solution space is known to the problem solver as a set of explicit

alternatives, and problem solving involves proving that one of them is best; and [2] those whose solution space is not known, and the problem solving involves construction of a solution. The former is called heuristic classification problem solving and the latter constructive problem solving. ASKE uses heuristic classification model hence we will concentrate on it.

One of the main characteristics of the heuristic classification model is that all possible solutions of the problem can be specified a priori. The heuristic classification problem solving involves the selection of one among a predetermined set of possibilities as the appropriate description of a situation. The inference structure of heuristic classification consists of data statements and solution features at various levels of abstractions and are mapped heuristically by different kinds of relations. This model has far reaching implications for KA as it precisely describes the kind of knowledge that is necessary for solving problems by heuristic classification.

#### **1.4.2 Automatic Knowledge Acquisition**

Clancey's analysis of problem-solving types has had a strong impact on the designers of KA systems (i.e., tools which interactively acquire knowledge directly from the domain expert without the intervention of the knowledge engineer). One of the insights has been that KA power can be resulted from specializing in a particular performance task to which the acquired knowledge will be applied. For example, McDermott (1988) has argued that task-specific knowledge in the form of well-specified roles in performing a specific class of tasks can provide an effective method for acquiring particular kinds of knowledge. The role-limiting method provides a strategy for knowledge elicitation. This idea has been realized by a number of KA systems, such as MOLE (Eshelman and McDermott, 1986) and SALT (Marcus et al., 1985). These systems specialize in some problem-solving method, an algorithm for applying domain knowledge to perform a task.

Bylander and Chandrasekaran (1987) have, in their notion of "generic tasks", proposed a similar approach to that of McDermott's role-limiting method. A generic task, like Clancey's problem-solving method, describes a problem. It consists of knowledge structures and inference strategies for dealing with the problem. Basically, a generic task is "an elementary generic combination of a problem, representation, and inference strategy about concepts" (Bylander and Chandrasekaran, 1987:235). The main difference between Clancey's heuristic classification and a generic task is that the former is a heterogeneous and the latter a homogeneous problem-solving method (Chandrasekaran, 1987).

While a KA system can obtain power by specializing in a particular performance task, it does narrow down its scope of applicability. For example, MOLE's cover-and-differentiate method of heuristic classification restricts its range to diagnostic tasks only and SALT can only be used for design tasks. We have a problem. If our aim is to provide a facility for developing any knowledge system, the task-specific approach, as it stands, will be ineffective. This is the problem addressed in this thesis, i.e., how to design a KA system so that it can have a wide scope of applicability. To make the project tractable, the range is set for analysis tasks, which can be solved by heuristic classification problem-solving method. The ASKE system is described in the following Section.

## 1.5 The System

ASKE contains knowledge engineering expertise which it uses to help domain experts encode their problem-solving expertise directly into a knowledge-base. The system derives its power from knowledge representation schemes called "templates". They provide a means for representing and acquiring domain knowledge. The problem-solving method of heuristic classification (Clancey, 1985) is used as the basic inference mechanism.

### 1.5.1 Knowledge Base

ASKE's knowledge base is made up of templates. These are structurally similar to frames (Minsky, 1975). A template is, however, more than just a data structure. Like a script (Schank and Abelson, 1977), it provides a conceptual structure for generating expectancies and guiding plans of actions. Templates serve various roles: [1] as a store for domain knowledge; [2] acquisition of domain knowledge ; and [3] as a guide to the domain experts in organizing their knowledge.

TEMPLATES	KNOWLEDGE HELD	USE
general	meta-knowledge about other templates	select acquisition and reference templates
acquisition	task-specific knowledge	build task model
reference	abstracted knowledge base	an exemplar for deve- loping a task model for the new application
working	new knowledge base	as a future exemplar

Figure 1-1 Template Types

ASKE uses four types of templates: general, acquisition, reference and working. Figure 1-1 summarizes the role played by these templates in KA. The general template contains information about other templates. Its main function is to identify the application task and select acquisition and reference templates for further KA. The acquisition template holds knowledge about task characteristics. It is used in the acquisition of the task model, the main concept categories describing the application task. The reference template consists of knowledge-base abstracted from ASKE-built applications. It acts as an exemplar for the acquisition of the task model. The working template is used for storing the new application.

### 1.5.2 Inference Structure

ASKE uses the problem-solving method of heuristic classification to drive the KA process. The main presupposition here is that a task can be represented as a classification problem: some solution object is selected from a set of enumerable candidates (e.g., diseases, components) on the basis of evidential considerations (e.g., symptoms, requirements). Like in the MOLE (Eshelman and McDermott, 1986) and ETS (Boose, 1985) systems, the inference structure is hard-wired into ASKE (it is implicitly recorded in the acquisition templates).

### 1.5.3 Methodology

The methodology is based on the assumption that the most difficult part for the experts, when trying to encode their problem-solving expertise into a computer program, is the initial organization of the knowledge so that it can be mapped into a target representation. The design of ASKE has been specifically guided by this issue: how to help the expert organize his/her domain knowledge, so that it is more conducive to the development of an expert system. The system provides the expert with an interface and a step-by-step procedure for encoding the knowledge.

The KA takes place in two stages, as shown in Figure 1-2. In the first stage, a task model (i.e., the main concept categories and the interrelationships between them that the domain expert uses to perform the task) of the domain is developed. This activity is guided by two bodies of knowledge: [1] the knowledge of possible application tasks; and, [2] exemplars abstracted from the knowledge-bases that were built using ASKE. The task types provide information about the kind of knowledge that is to be obtained from the expert. For example, the main data and solution categories for the new application. Often experts find it difficult to think in this top-down fashion. To assist them, an example of a similar, already built task model is presented.

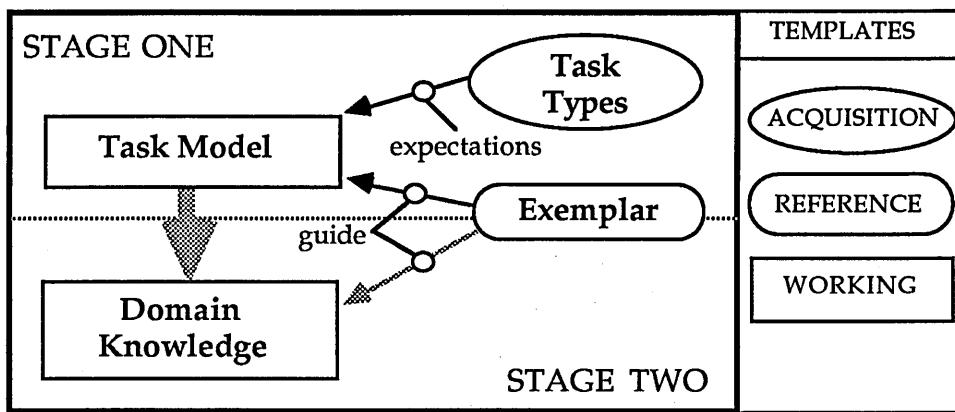


Figure 1-2 A two stage model for automatic knowledge acquisition

The Figure shows how the templates are used in KA. The general template, not shown here, contains information about the various templates in ASKE. This knowledge, which is built up at run-time, forms the basis for selecting the task type and the exemplar used for developing a task model.

In the second stage, the task model is used to guide the acquisition of the concept hierarchies and heuristic associations between hierarchies. Once again, the exemplar has a role to play. The knowledge of how concept hierarchies were constructed in a similar domain can be very useful. For example, AIDS infections can be classified in terms of the site of infection or categorized by the type of agents. The expert can be aided in decision making from the knowledge of how infections were categorized in a similar system which diagnosed infectious diseases.

At the end of stage two, ASKE generates an if-then rule for every heuristic association. The rules are quite simple in structure and not suitable for automatic testing and evaluation. The expert is, hence, provided with facilities for editing the rules.

## 1.6 Scenario

To illustrate ASKE's processing of information, I will present a scenario of a hypothetical session. It concerns the acquisition of knowledge for interpreting settlement sites from an expert archaeologist. To summarize

the archaeological reasoning involved, the archaeologist analyzes the evidence provided by material relics found at a site previously occupied by past cultures, and produces a plausible description of activities that took place in the site. Further conclusions may be drawn regarding the population of the site, the trade contacts, period of occupation, social organization, etc.

### 1.6.1 Stage One

The goal of the first stage is to build a model for interpreting settlement sites. This is achieved in three steps. Initially, ASKE interacts with the expert, guided by the general template, to identify the nature of the new application. On the basis of this, acquisition and reference templates are selected. The acquisition template provides expectations of the kind of knowledge required in the new model. The reference template acts as an exemplar. Finally, a model is developed. (This model is held in the working template).

#### Step One

*(The following questions are generated from the general template.)*

[1] What is the domain of your expertise?

=> archaeology

[2] What task will the new application perform? [Choose one from: selection, interpretation, diagnosis, debugging]

=> interpretation

[3] What is your area of specialization?

=> settlement site

#### Step Two

From [2] above, the acquisition template for the task of interpretation is



selected. The reference template is selected from existing knowledge-bases in ASKE. The selection strategy is based on the closeness of the new application to previous cases. If more than one is found, the user is asked to choose from the list. In this case there is only one knowledge-base, that for interpreting burial sites from archaeological data, hence it is selected automatically.

### Step Three

*(The acquisition template provides the following questions. The statements in bracket comes from the reference template.)*

[4] What are the main categories of observed data for interpreting settlement sites? (The main categories of observed data for interpreting burial sites were: artifacts, ecofacts, features.)

=> artifacts, features

[5] What are the main solution categories for interpreting settlement sites? (The main solution categories for interpreting burial sites were: hypotheses)

=> activities, site profile

### 1.6.2 Stage Two

For the interpretation of settlement sites, the important facts are the different categories of observed data and solution. The actual expertise in interpreting sites lies in the ability to map data to interpretations. The rules therefore depict transformations from data to interpretation. These rules are of the form:

```
IF      data and data and ....
THEN   interpretation and .....
```

The following dialogue shows how the model for interpreting settlement sites is used to interrogate the expert. (NOTE: Most of the interaction, from

hereon, is through ASKE's graphical interface.)

*(Obtain classification of different data types.)*

[7] What are the different types of artifacts ?

=> pottery, stone, metal

[8] What are the different types of features ?

=> pit, ditch, hearth

*(Obtain classification of different solution types.)*

[9] What are the different types of activities?

=> cooking, pottery-making, butchering, storage

[10] What are the different types of site profile?

=> exchange contacts, social status, occupation

*(The attributes of all data types are obtained.)*

[11] What are the important attributes of pottery that may contribute towards the interpretation of this settlement site?

=> (attribute) fabric

(possible values) coarse, fine

(attribute) decoration

(possible values) plain, ornamented

*(The relationship between data and solution is identified. This is used to automatically generate rules.)*

[12] What can you conclude from the fabric of pottery?

=> (activity) cooking, storage

(site profile) exchange contacts

[13] What can you conclude from the decoration of pottery?

=> (activity) cooking

(site profile) exchange contacts, ritual practices

Next, ASKE generates a rule for every association between concepts. The concept from the data category is made the premise and the one from solution category the conclusion of the rule. For example, "Rule1" and "Rule2", below, are generated from [11], [12] and [13]. Both rules have the

same premise but different conclusions.

<u>Rule1</u>		
IF	artifact is a pottery	
	fabric is (coarse, fine)	
	decoration is (plain, ornamented)	
THEN	activity is cooking	
<u>Rule2</u>		
IF	artifact is a pottery	
	fabric is (coarse, fine)	
	decoration is (plain, ornamented)	
THEN	site profile is exchange contacts	

The automatically generated rules are very general and further editing is required. The rules are, hence, displayed in the Rules Editor facility of ASKE. The expert is then asked to edit the rules. The following shows the edited version of the two rules.

<u>Rule1</u>		
IF	artifact is a pottery	
	fabric is coarse	
	decoration is plain	
THEN	activity is cooking	
<u>Rule2</u>		
IF	artifact is a pottery	
	fabric is fine	
	decoration is ornamented	
THEN	site profile is exchange contacts	

The ASKE session ends when the rules are edited. These are then output to a file. Finally, ASKE creates a new reference template which will contain knowledge abstracted from the present session. This template is stored for future sessions.

## 1.7 Structure of the thesis

Chapter II presents a short resume of the expert system technology. Next, the process of knowledge engineering is described and some of the major problems in the traditional KA methods are pointed out. The advantages of automating the knowledge engineering process are presented.

Chapter III presents a classification of knowledge engineering tools. The ASKE system is categorized according to this scheme and some of the other systems within the category are reviewed. Finally, a model for automatic KA is presented.

Chapter IV discusses the three main features of ASKE. The characteristics of analysis problems, from which the task models are derived, are described. Next, a case for using previous examples to guide KA is presented. Finally, the representational scheme of templates is introduced.

Chapter V presents the ASKE system. The presentation is focussed on: the ASKE methodology for KA; how it is implemented; and, the various interface facilities provided for the user to encode his/her domain expertise.

Chapter VI presents an example of how ASKE processes information. ASKE is used for developing an initial prototype for interpreting archaeological settlement sites from material remains.

Chapter VII presents a discussion of the ASKE system. The issues that arise from the research are discussed.

Chapter VIII presents the conclusion. A summary is given as well as future research directions.

# Chapter 11

## EXPERT SYSTEMS AND KNOWLEDGE ENGINEERING

*Knowledge is an artifact, worthy of design.*

### 2.1 What is an Expert System?

Expert systems are computer programs which solve problems by applying substantial knowledge of specific areas of expertise. They are basically "a high-level intelligent support for the human expert" (Feigenbaum and McCorduck, 1983:86). This is because they do not have any general mechanism for common sense reasoning, but contain knowledge of highly circumscribed domains. Expert systems thus act as "intelligent assistants" by providing quick solutions to problems which may or may not be definitive and the final decision rests with the user of the system.

Knowledge is the major factor in the performance of an expert system. It is held in the knowledge-base module of the expert system (Figure 2-1) and is typically represented in one of two main formalisms: production rules and structured objects (e.g., frames and semantic nets). Depending on whether rules or objects are used for codifying the problem-solving know-how of a human expert, the system is called rule-based (e.g., MYCIN) or model-based (e.g., INTERNIST), respectively. Of the two, rule-based expert systems are more in vogue. The KA research presented in this thesis is geared to produce knowledge-bases for rule-based expert systems only.

### 2.1.1 Rule-Based Expert Systems

The methodology for many contemporary expert systems is derived from the MYCIN system, which was developed at Stanford in the mid-1970's. It uses expert medical knowledge to diagnose and prescribe treatment for spinal meningitis and bacterial infections of the blood. Two of the most important legacies of the MYCIN project were: a prototype for rule-based systems and an architecture for expert systems.

The guiding principle for rule-based systems is that "the knowledge critical for decision-making *can be encoded* in the form of highly modular rules" (Weiss and Kulikowski, 1984:4). The domain knowledge of rule-based systems is represented as sets of rules that are checked against a collection of facts or knowledge about the current situation. Rules are expressed as IF-THEN statements. The IF part of the rule is usually referred to as premise or condition and the THEN part as conclusion or action. Typically, knowledge in rule-based systems is represented as situation-action rules of the following form:

IF        There is evidence that A and B and C are true,  
THEN    Conclude there is evidence that D is true (0.9).

Rules often have certainty factors, which numerically indicate the strength of the rule, associated with them. In the above rule, the number 0.9 indicates that the evidence is strongly indicative (0.9 of 1) but not absolutely certain. Certainty factors provides a means of drawing inferences from uncertain or incomplete data. The ability to reason under uncertainty is often cited as one of the main characteristics of expert systems distinguishing it from conventional programs.

In expert systems, the main sources of uncertainty are the use of abductive inference and the reasoning with missing or unreliable data. In abductive reasoning, one reasons from premise to conclusion: if P then Q. The

uncertainty in this is that there might be other premises when Q is true, for example: if Z then Q. However, within a narrowly defined problem area it is possible to specify all premises when a certain conclusion is true. For instance, when configuring a computer, the components either go together or they do not. In this case, there is no need for certainty factors.

Another of the features that distinguishes expert systems from conventional computer programs is that there is a rigid separation between the problem-solving knowledge (i.e., knowledge-base) and methods for utilizing this knowledge (i.e, inference engine). In MYCIN, for instance, domain knowledge is encoded as rules. This knowledge is separated from the mechanism of interpreting and applying the rules (Figure 2-1). MYCIN has provided a prototype for current expert systems.

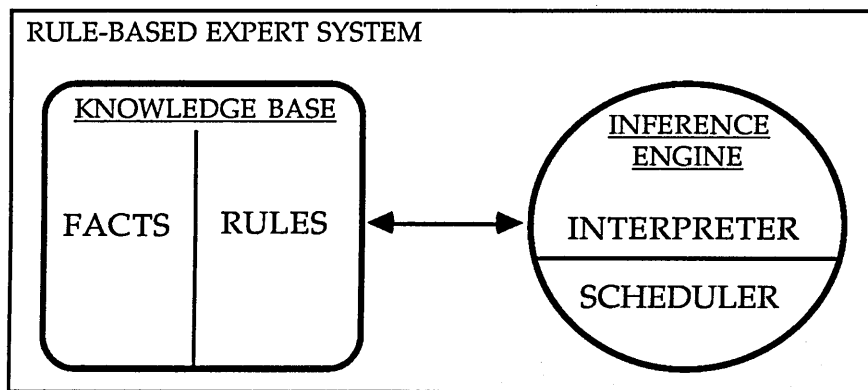


Figure 2-1 The structure of an expert system.

Two main elements of an expert system are: a knowledge-base consisting of facts (data) and rules that use those facts for decision making; and an inference engine consisting of a rules scheduler and interpreter which select and operate on the rules in the knowledge-base to produce solution/s of the problem. Another element of an expert system, not shown in the Figure, is that of user interface through which the user accesses the expert system.

The main advantage of separating knowledge and control in an expert system is that the same inference engine can be utilized to drive different

knowledge-bases. An expert system stripped of its knowledge (i.e., an inference engine with a knowledge representation language) is called a "shell". One of the best examples of a shell is EMYCIN (Buchanan and Shortliffe, 1984), which was produced by removing specific domain knowledge from MYCIN. EMYCIN has been used to effectively drive the knowledge-base of several expert systems.

### 2.1.2 Knowledge-Base

The knowledge-base consists of facts (or data) and production rules which reason about the facts. A rule is expressed as a conditional statement with an antecedent (premise or condition) and a consequent (conclusion or action) component. The rule defines that if the antecedent condition can be satisfied, the consequent can be too. When the consequent is an action, the effect of satisfying the antecedent is to schedule the action for execution. When the consequent is a conclusion, the effect is to infer the conclusion. Facts, in contrast to rules, are static and inactive. They represent concepts, properties and relations. These are utilized by the rules, as the following example shows.

<u>Rule1</u>	<u>FACTS</u>
IF X is-an activity-area the content of X is-a firepit	activity-area
THEN activity of X is sleeping-room	kiva
	content: squash
	location: subterranean
<u>Rule2</u>	plaza
IF X is-an activity-area	content: firepit
the location of X is subterranean	location: ground
THEN activity of X is ritual	

The facts knowledge-base consists of two activity-areas: kiva and plaza, which are described by two properties of content and location. The two rules infer the activity of an area from the given facts. X in the rule is a variable and could be substituted by either of the activity-areas. For



example, if X is assigned the value "kiva", the result would be: "activity of kiva is ritual".

The knowledge of expert systems is normally derived from human experts, rather than from other knowledge sources such as text books and manuals. Experts are people who are very good at solving specific types of problems. Their skill usually comes from extensive experience, and detailed specialized knowledge of the problems they handle. The applications of expert system are now so extensive that an accurate list of domains of applications would be much too long to include here.

### 2.1.3 Inference Engine

The inference engine contains knowledge for deciding: [1] how to apply the domain knowledge, and [2] when and in what order to apply different pieces of domain knowledge. The first provides the global regime for controlling the behaviour of the system. This knowledge is domain-independent and tends to be hard-wired into the interpreter. At the global level of control the main decision is made regarding whether the rules should be driven backward or forward. In the backward driven interpreter, the chaining starts from the conclusion to be established to satisfying the conditions necessary for its truth. In the forward driven system, the chaining progresses from the conditions that are known to be true towards the conclusions to be established.

The second control regime is explicitly coded into the scheduler, which controls the system behaviour at the local level. The knowledge at this level is domain-dependent and includes methods of conflict resolution (e.g., refractoriness, recency and specificity) which determine which and when rules are fired. These resolution mechanisms vary from system to system.

## 2.2 The Knowledge Engineering Process

The term knowledge engineering (KE) was coined in the mid-1970's by Feigenbaum (1977) and refers to "the process of mapping an expert's knowledge into a program's knowledge-base" (Buchanan and Shortliffe, 1984:5). The KE process consists of three stages: elicitation, formalization and implementation, as represented in Figure 2-2<sup>1</sup>.

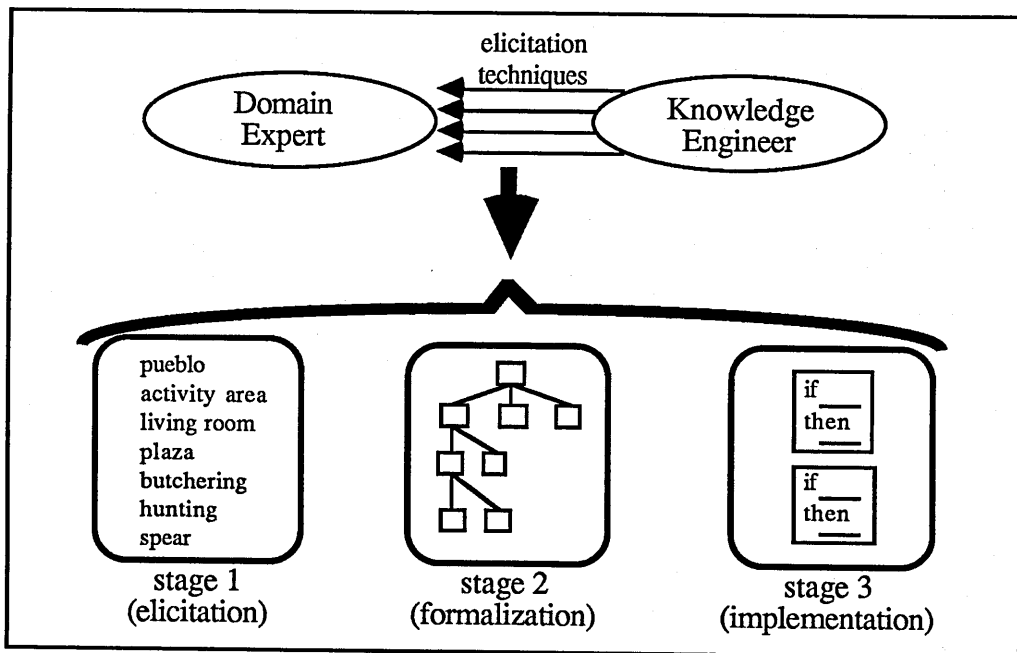


Figure 2-2 The Knowledge Engineering Process

Constructing an expert system involves selecting an appropriate problem, finding available expert(s), and possessing an appropriate system building tool. The next major task is KA, which involves elicitation, formalization and implementation of problem-solving knowledge.

Within the context of automatic KA, KE and KA are synonymous, they connote the encoding of an expert's knowledge within a computer

<sup>1</sup> Note that though the three stages are described as occurring in isolation from one another, in reality, they are carried out concurrently. For instance, a knowledge engineer while eliciting knowledge would also be thinking about formalization. Moreover, the representational scheme that is adopted is often strongly influenced by the knowledge engineer's favourite tool for building expert systems; although this depends on the choice of methodology used for expert system development.

program. However, KE subsumes KA in the normal usage of the term (i.e., when manual methods are employed). For instance, in the KEATS methodology, KE is characterized as consisting of "acquisition, representation, implementation and debugging of (a model of) the expert reasoning and phenomenology for a chosen target domain" (Motta, et al., 1989a:298). The acquisition stage includes knowledge elicitation and interpretation of data.

The KEATS methodology is one of the most comprehensive description of the KE process, however, its applicability is restricted to semi-automatic (see Section 3.2 for definition) and manual methods of KE. From the perspective of KA systems, and the one which this thesis presents, the KA process also includes representation and implementation stages. This is implicit in the design of KA systems: they interact directly with a domain expert to produce a prototype system. And their KA activity include the three stages depicted in Figure 2-2. In the rest of this subsection, I will briefly describe the three stages of KA.

### **2.2.1 Knowledge Elicitation**

The elicitation stage involves extracting problem-solving expertise from the domain expert. This is rather a difficult stage of KA mainly because there are no definite guide-lines for performing knowledge elicitation (Forsythe and Buchanan, 1988). What is available is an arsenal of techniques, for example, interviewing, verbal protocol, and scaling methods. There has been some research on the mapping between techniques and types of knowledge (e.g., Burton et al., 1987, 1988; Gammack and Young, 1985). For example, Burton et al., from their comparative study of techniques over experts, conclude that the ladder grid technique is particularly suitable for classification domains. However, they also note that "techniques are differentially suitable for different experts" (1988:89).

In spite of a large number of techniques available for knowledge elicitation, knowledge engineers, to a large extent, use interviews and verbal protocols only. Unfortunately, neither of these techniques are structured enough to be automated. Psychometric techniques such as repertory grid, on the other hand, have been automated (e.g., Boose, 1985, Shaw and Gaines, 1987). I will briefly describe each of these techniques. (For a more comprehensive survey of techniques see: Neale, 1988; Burton and Shadbolt, 1987; and Welbank, 1983).

### Interviews

Interviewing is the most commonly used technique for acquiring problem-solving expertise from the domain expert. There are many ways of structuring an interview, but they all suffer from the fact that the knowledge elicited may not be the same as that utilized in practice (e.g. Welbank 1983). Furthermore, depending on the way the questions are phrased, the interviewer may, quite unintentionally, introduce bias and error into the interviewing process (LaFrance, 1987). The attraction of this technique is that the knowledge engineer can have control over how the session progresses, and the product of the session is normally an easily analysable transcript.

### Protocol Analysis

The protocol analysis method has the merit of deriving a much more true-to-life task situation than the interview method, but the final transcripts are much harder to analyse (Shadbolt and Burton, 1989). Like interviewing, there are a number of ways of performing protocol analysis. Typically, an expert is given a problem and asked to solve it while thinking aloud. This is tape recorded, transcribed into protocols, and then analyzed - the final product is often production rules. The technique of verbal protocols is not without problems. For example, Nisbett and Wilson (1977) have argued that protocols reflect the subject's tacit

knowledge about plausible causes for his/her responses. Thus, they have questioned the accuracy and consistency of verbal reports. Ericcson and Simon however suggest that "verbal reports, elicited with care and interpreted with full understanding of the circumstances under which they were obtained, are a valuable and thoroughly reliable source of information about cognitive processes" (1980:247).

There are a number of other problems with the technique of protocol analysis besides their accuracy. For example,

"Protocol analyses share with the unstructured interview the problem that they may deliver unstructured transcripts which are hard to analyse. Moreover, they focus on particular problem cases and so the scope of the knowledge produced may be very restricted. It is difficult to derive general domain principles from a limited number of protocols." (Shadbolt and Burton, 1989:5)

### Repertory Grid

Interviewing and verbal protocol techniques are seldom used in isolation. Quite often, knowledge engineers also use formal techniques derived from psychological testing. Repertory grid technique, devised by George Kelly with reference to his cognitive theory of personality, is one of the most popular. In the grid method, first a list of elements (i.e., domain concepts) are obtained from the expert. Next, constructs, which indicate the dimensions upon which the sets of elements show similarities or differences, are elicited. Finally, each of the elements is rated along this dimension, usually on a numerical scale such as 1-5. The resultant grid is then used to derive rules. While grid method is particularly good at eliciting domain concepts, it is not suitable for eliciting causal, procedural or strategic knowledge (e.g., Boose, 1985; Gammack and Young, 1985). Furthermore, the technique can be demanding on the expert if the number of elements to be compared gets too large.

### Automatic Knowledge Elicitation

KA systems perform knowledge elicitation in one of two ways: [1] They implement a knowledge elicitation technique, for example, ETS (Boose, 1985) uses repertory grid technique. [2] They use the generic structure of the task or domain to drive knowledge elicitation; for example, MOLE (Eshelman and McDermott, 1986) uses a heuristic classification model (Clancey, 1985) to elicit knowledge from domain experts.

#### 2.2.2 Knowledge Formalization

The formalization stage includes domain conceptualization and knowledge representation. At this stage, the raw knowledge from the elicitation stage is given structure by mapping it onto a suitable representational scheme; the concepts, sub-problems and control features are formalized into representations such as frames and rules. In MYCIN, for instance, the formalization stage involves mapping the facts into attribute-object-value triples. Consider the facts about the organism bacteriodes. It has three important identifying features: gram stain, morphology and aerobicity with the values gramneg, rod and anaerobic respectively. This would be formalized as follows:

IF	the gram stain of the organism is gramneg, and	[1]
	the morphology of the organism is rod, and	[2]
	the aerobicity of the organism is anaerobic	[3]
THEN	there is suggestive evidence (.6) that the identity of the organism is bacteriodes	[4]

The conceptualization and representation stages are viewed here as intertwined: the raw knowledge is given structure by mapping it directly onto the chosen representational scheme. In the non-automatic methodologies of KA, however, there is a fine line drawn between the two stages. Motta et al. (1989b:9), for example, argue:

"At the domain conceptualization stage, the knowledge engineer attempts to impose a global structure upon the data collected so far in order to produce an abstract model of the problem in terms of taxonomic hierarchies, causal networks, tables, flow diagrams, or whatever organization s/he finds convenient for modelling the domain and the problem-solving structure of the problem. The important element that differentiates this level from the knowledge representation one is that the representation at this level doesn't need to be runnable, but is meant to be a semi-formal characterization of the structure of the task."

### 2.2.3 Implementation

At the implementation stage the knowledge from stage two is mapped into a representational formalism which is associated with the tool chosen for implementing the knowledge-base. It is however often the case that the representational scheme adopted in the previous stage is strongly influenced by the representational and reasoning facilities afforded by the implementation toolkit. For example, the above MYCIN rule is directly translated into a runnable form:

PREMISE: (\$and (same cntxt gram gramneg)	1
(same cntxt morph rod)	2
(same cntxt air anaerobic))	3
ACTION: (conclude cntxt identity bacteriodes tally .6)	4

In automatic KA, the implementation and formalization stages are closely related. They are best viewed as two levels, internal and external. Implementation is the data represented internally in the machine and hidden from the user. Formalization is the external representation of the same data, presented to the user.

## 2.3 Knowledge Engineering Methodologies

There are three main approaches to building knowledge-bases: rapid prototyping, knowledge-analysis and task-specific (Woodward, 1989). In

rapid prototyping, the knowledge engineer attempts to implement a prototype expert system as soon as sufficient knowledge is extracted. In the knowledge-analysis approach, the knowledge engineer only attempts the first two stages of KA. The implementation stage is delayed until *all* of the expert knowledge is elicited. In the task-specific approach, models of task characteristics are used to drive the KA process.

### 2.3.1 Rapid Prototyping

For this approach, the process of KA is as depicted in Figure 2-2; the knowledge engineer would elicit knowledge, map this into a suitable representation scheme and implement a prototype system which is then tested and built upon. Usually, the final representational formalism is selected prior to the elicitation procedures. The underlying assumption for this approach is that knowledge-base construction is an inherently experimental process (Hayes-Roth et al., 1983).

### 2.3.2 Knowledge-analysis

The basic assumption underlying this approach is that "knowledge is a multi-level phenomenon" (Woodward, 1989:155). Static knowledge, according to Brachman (1978) is represented at five different levels: linguistic, conceptual, epistemological, logical and implementational. Breuker and Wielinga (1985) argue that knowledge should go through multi-level analysis for an effective knowledge-base. They have proposed a methodology, based on knowledge-analysis, for developing expert systems called KADS. The starting point of the KADS methodology is that there should be a fairly complete conceptual model of a future knowledge-based system before any serious effort towards design and implementation is spent. This conceptual model is represented at the epistemological level.

At the heart of the KADS approach is the notion of "Interpretation Model", a template structure which contains a generic task model:



"An interpretation model is a generic model of the problem-solving process for a class or prototype of task [e.g., diagnosis, monitoring and planning]. It looks like a catalogue of task-specific ingredients from which selections can be made that appear to match the knowledge structures of the domain." (Breuker and Wielinga, 1987:31)

Interpretation models provide expectations of the kind of knowledge required for solving application tasks. Their main role, however, is in the analysis of data.

### 2.3.3 Task-Specific

While the knowledge-analysis approach is characterized by analysis of knowledge at multiple levels, the task-specific approach relies on only single level analysis of the task. The underlying assumption in this approach is that functional models of tasks and problem-solving methods can be used to guide KA. A problem-solving method contains task-specific knowledge, such as the different roles knowledge plays in the task, and control knowledge which define the order in which subtasks have to be solved to perform the task. The task-specific approach is exemplified in McDermott's (1988) role-limiting method and Chandrasekaran's (1987) generic tasks. These methods provide improvements on the rapid prototyping approach. Both advocate that the elicitation and representation of knowledge is guided by the generic structure of the task which is initially identified. There are however a number of important differences between knowledge-analysis and task-specific approaches.

[1] In the knowledge-analysis approach there is an emphasize placed on explicitly separating the different types of knowledge found in expertise. There are four main levels of knowledge: domain, task, inference and strategic (e.g., Breuker and Wielinga, 1985; Shadbolt and Burton, 1989). The domain and the task level knowledge are sometimes called declarative and procedural knowledge, respectively. Declarative knowledge is "knowing that": the static aspects of knowledge such as facts about objects,

events and their relations; procedural knowledge is "knowing how": how to find relevant facts and make inferences (Winograd, 1975). The inference level knowledge describes the overall system behaviour, about how expertise is organized and used. The strategic knowledge describes the strategy used in problem-solving.

In the task-specific approach such a fine-grained distinction between the various knowledge types is not made. The inference and strategic knowledge tend to be implicit in the system and used with some task level knowledge to drive the KA process.

[2] The task-specific approach clearly identifies the importance of building KA tools specific to the task type. The emphasis is to limit the role of the expert to the provision of domain- or task-specific knowledge. The knowledge-analysis approach tends to involve the expert to a much greater degree so that the resulting knowledge-base reflects the benefits of the multi-level analysis of the knowledge. However, the emphasis is more on the analysis of knowledge to the extent that the importance of knowledge elicitation is undermined.

[3] The knowledge-analysis approach provides a general KE methodology. The approach presumes that KA will be done by a knowledge engineer, with or without some automated assistance. The task-specific approach, on the other hand, specifically addresses the question of how to automate the KA process. The methodology assumes that KA will be ultimately carried out by a KA tool, without the intervention of a knowledge engineer.

## 2.4 Problems in Knowledge Acquisition

Techniques for KA are still at an early stage of development. The process of transferring knowledge from an expert's head to a computer program is still labour-intensive and time-consuming, making it the most expensive

---

component of the construction of an expert system (Duda and Gaschnig, 1981). The real problem seems to be the fact that there is no well defined methodology for doing KA. Guide-lines abound (e.g., Bobrow et al., 1986; Grover, 1983; Hayes-Roth et al., 1983), however they are just adhoc recipes for the knowledge engineer. The lack of any formal theory of knowledge-engineering has meant that knowledge engineers have had to employ trial and error to build expert-level prototypes which take as long as 6-24 man-months (McDermott, 1982; Smith and Baker, 1983). Besides the lack of methodology, further obstacles to KA stem from the agents involved in the process: the expert and the knowledge engineer.

#### 2.4.1 Expert

Expert systems are founded on the premise that human expertise can be codified and replicated by rule-following machines. The elicitation of expertise is however a non-trivial task. The main reason for this is that human expertise lies in laid-down experience, gathered over a number of years. As an individual progresses from the status of a novice to that of an expert his/her knowledge is built up incrementally<sup>1</sup>. Facts, once unrelated, get integrated through occurrence in the same episodes. With the increase in expertise at problem-solving, chunks of knowledge are integrated together into higher order chunks. An expert thus has more and better organized chunks of knowledge than a non-expert (e.g., Chase and Simon, 1973). This knowledge, however, cannot be easily extracted as has been so often claimed to be. For example, according to Feigenbaum and McCorduck (1983) knowledge is some tangible in the heads of experts and can be mined. This is rather a misguided view because expert knowledge is so routinized that experts no longer know how they solve problems. The knowledge is tacit and is not available to conscious awareness (Johnson, 1983).

---

<sup>1</sup> For a discussion of how expertise is acquired see Section 4.3.1.

The nature of the expert knowledge makes the task of KA difficult: experts find it difficult to describe exactly how they do what they do, especially with respect to their use of judgement, experience, and intuition (Duda and Gaschnig, 1981). Furthermore, some of the expertise is made up of unarticulated understanding of the world and task at hand (e. g., Collins et al., 1985), and it has never before explicitly acquired but obtained through experience (Berry, 1987). And even when knowledge is forthcoming, there is always the possibility that this knowledge may be incomplete or even incorrect (Gaines, 1987).

Very often, KA is carried out in a room, away from the domain expert's place of work. This is especially true when KA is done by a system. However, this can be a problem. For example, Godden and Baddeley (1975) found that recall is best in the environment in which information is encoded. If the expert's performance varies with the context, it will be necessary to do KA in the expert's normal place of work.

A more serious problem than the context-dependency of memory is that of compiled knowledge. According to Anderson's (1987) model of skill acquisition, knowledge which was once represented explicitly through repeated use becomes "compiled-down" to become implicit. A result of this is that the expert's performance becomes more efficient. For the knowledge engineer, however, eliciting expertise becomes difficult because the human, on becoming an expert, loses access to the problem-solving knowledge.

The personality of the expert him/herself is a further source of complication. The attitude of the domain expert towards the building of an expert system, for instance, can create problems (Burton and Shadbolt, 1987). Obviously, an uncooperative expert can make the task of knowledge extraction very difficult.

As pointed out earlier, expertise is made up of different types of knowledge. Gammack and Young (1985) have argued that different techniques differentially tap different knowledge types. A formal evaluation of knowledge elicitation techniques by Burton et al. (1987) has revealed that the efficacy of each technique is also differentially related to experts' individual characteristics. For example, they found that over 50% of the variance in elicitation time for interviewing technique was accounted for by personality factors.

#### 2.4.2 Knowledge Engineer

Because the required format for expressing knowledge is complex and not easy to learn, busy experts may be unwilling to devote the time to mastering these techniques and find it easier and quicker to communicate with a knowledge engineer. The knowledge engineer's task is to explain the program's framework to the expert and to translate the expert's problem-solving knowledge into the framework. However, the latter task requires the knowledge engineer to have a deep knowledge of the application domain. This adds precious time to the knowledge-base development cycle.

The biggest problem however is lack of communication between expert and knowledge engineer. Because the knowledge engineer is really a layman, the expert is forced to provide simplified explanations and in the process s/he leaves out relations or concepts which very often turn out to be important for the performance of the program (Buchanan, 1969). An added complication is that "both the expert and the programmer are simultaneously developing representations of the domain that they believe are appropriate for the task and for the program that performs that task" (Buchanan, 1979:418). The success of the project will therefore depend on the degree of concurrence between the two models.

## 2.5 Automation of the Knowledge Engineering process

One trend in KA research is to eliminate the knowledge engineer and get the expert working directly with a shell. Shaw and Gaines (1987) identify several reasons to doubt that human labour is the appropriate solution for "the knowledge engineering problem". They suggest that using a human intermediary may be less effective:

"knowledge may be lost through the intermediary and the expert's lack of knowledge of the technology may be less of a detriment than the knowledge engineer's lack of domain knowledge" (111).

The automation of the knowledge engineering process involves the development of a software tool which would enable domain experts to encode their expertise directly into the system. This would have the following advantages over the traditional practice of knowledge engineering.

- 1] The knowledge engineer is removed from the knowledge-base development cycle. This eliminates all problems related to the knowledge engineer.
- 2] The problems related to the expert's personality are eliminated. There is an implicit assumption that if the expert decides to develop an expert system then s/he will be self determined and motivated.
- 3] The communication barrier is removed. The expert does not have any need to simplify his/her explanations assuming the tool can understand him/her.

In replacing a knowledge engineer with a KA tool one makes two assumptions. First, the domain expert is computer literate. This is probably a safe assumption to make in the current era of microelectronic revolution. Second, the domain expert knows what expert systems are

and the role they play in decision making. Normally, a knowledge engineer would brief the expert on the subject. However this is not such a crucial assumption as the basic principles of expert systems technology are not hard to grasp and can be understood in a matter of hours.

By replacing the knowledge engineer, there will be an added onus on the system developer to provide an "intelligent" interface to the system. Smith and Baker (1983), for example, have emphasized the importance of good interface for user acceptance. Their system for interpreting oil-well logs, Dipmeter Advisor, has 42% of the code devoted to interface. Kitto (1988), after examining the KNACK (Klinker et al., 1987) and AQUINAS (Boose and Bradshaw, 1987) tools, found that a knowledge engineer was required for an efficient use of the facilities provided by these systems.

# Chapter 11

## KNOWLEDGE ACQUISITION

*In part, they were correct.*

### 3.1 Introduction

Once it had been realized that KA was a major obstacle in expert system development, researchers started looking towards automating the KA process. The search for automated solutions to the problem can be traced back to TEIRESIAS (Davis, 1979). TEIRESIAS was the first system to conduct a dialogue with the expert in order to expand the knowledge-base. Since these initial efforts, a large variety of tools have been created which assist in the building of expert systems. Their position in the development cycle is depicted in Figure 3-1.

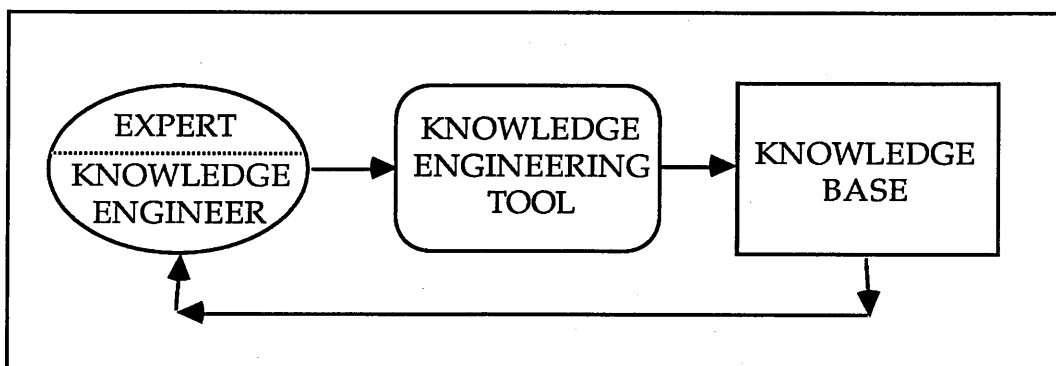


Figure 3-1 Overview of Knowledge Engineering

Either an expert or a knowledge engineer uses a KE tool to create a knowledge-base for the application.



### 3.2 A Classification of Knowledge Engineering Tools

Automatic aids to KA may be categorized by the degree to which the elicitation stage is automated. Accordingly, the KE tools can be divided into three groups of: KE environments, KE support-tools and KA systems. Figure 3-2 represents this classification.

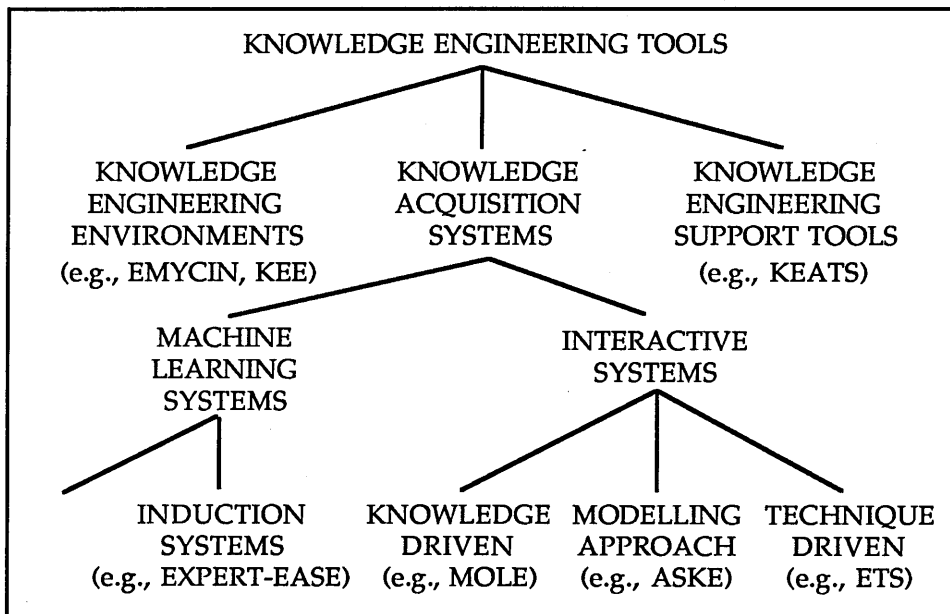


Figure 3-2 Classification of Knowledge Engineering Tools

There are three major classes of KE tools: environments, which do not play any role in knowledge elicitation; support-tools, which provide semi-automatic help to the knowledge engineer in encoding and analyzing data; and KA systems, which automatically elicit knowledge from domain experts.

Before going on to describe the various KE tools I would like to summarise the level of refinement that is to be found in the three groups. The tool types are compared along four dimensions: [1] the robustness of implementation, i.e., whether commercial or research; [2] the tools contain some epistemological model which is used to guide KA; [3] the intermediate knowledge representation (KR) facilities provided; and [4] the support for carrying out knowledge elicitation.

Level of Refinement	Environments	Support-tools	KA Systems
Implementation	commercial (KEE, ART)	commercial (KEATS)	research (MOLE, ASKE)
Epistemological Model	none	sometimes (task-specific)	yes
Intermediate KR	many	many	one
Elicitation	none	data analysis only	elicitation is automated

While both KE Environments and Support-tools are available as commercial systems, KA systems are still at the research stage of development. One of the main characteristics of KA systems is that carry out knowledge elicitation, which is guided by some epistemological model. Epistemological models are used by Support-tools to provide facilities for data analysis, for example, analysing protocols. Environments contain no epistemological models and they do not normally provide any facility for knowledge elicitation. Most of the Environments, however, provide multiple schemes for representing knowledge (e.g., frames, objects, rules). Support-tools often contain more than one KR formalism. KA systems are more restrictive, they tend to have only one representational scheme.

### 3.2.1 Knowledge Engineering Environments

A KE environment provides the knowledge engineer with a workbench for implementing a knowledge-based system. Its facilities include one or more formalisms for knowledge representation and a knowledge-base editor. It serves a rather passive role in KA. The knowledge engineer elicits knowledge from the domain expert with virtually no help from the tool. The knowledge engineer then has to decide for him/herself how to encode the elicited knowledge. Basically, the KE environments provide

knowledge encoding facilities but they do not explicitly prescribe any methodologies for carrying out the KA process.

Waterman(1986) categorizes these as either skeletal systems (more commonly known as "shells") or general purpose systems (also referred to as "programming environments" or "toolkits"). Shells (e.g., EMYCIN, ROSIE) offer a faster, cheaper development route but they constrain the designer in the limited formalisms they support. Hybrid toolkits (e.g., ART, KEE, Knowledge-Craft) offer the user a choice of knowledge representations and inference methods. The greater choice afforded by the toolkits however does not make KA any easier, but on the contrary, "they provide the knowledge engineer with a bewildering array of possibilities and little, if any, guidance under what circumstances which of these possibilities should be used" (Reichgelt and van Harmelen, 1986:2).

### **3.2.2 Knowledge Engineering Support-Tools**

The KE support-tools go one step further than the previous class of tools: they contain most of the features of the environments and also provide semi-automatic help at the knowledge elicitation stage. This assistance is normally in the form of semi-automatic transcript analysis and some automatic interviewing. Normally, a support-tool will include a methodology for KA. However, the KA process is not automated and the main decisions about which and when to use a particular knowledge elicitation technique is left in the hands of the knowledge engineer. Furthermore, like environments, support-tools are quite complex to use.

An example of this class of tools is KEATS (Motta et al., 1989a,1989b), which is a toolkit that is based on a methodology for building expert systems. KEATS provides semi-automated help to the knowledge

engineer in data analysis and domain conceptualization. It contains a hypertext-based facility called Acquist, which allows the knowledge engineer to analyse transcripts and create conceptual models of the domain. KEATS does not, however, take an active role. All the decisions are taken by the knowledge engineer.

### **3.2.3 Knowledge Acquisition Systems**

The knowledge acquisition systems are completely automatic. They elicit knowledge from a domain expert and generate a prototype. These systems typically interact with domain experts, organize the knowledge they acquire, and generate a knowledge-base. There are two major classes of systems within this group: Machine Learning (ML) and Interactive. The ML systems use techniques such as induction, analogy and case-based reasoning to learn new things. For the purpose of this thesis we will only look at the subclass of ML systems that use induction. The interactive systems use some kind of specialist knowledge and a methodology for KA.

#### **3.2.3.1 Induction Systems**

Induction systems use inductive learning techniques to extract knowledge from experts. Inductive learning connotes the use of inductive inference on specific instances to arrive at general descriptions (Michalski, 1983). Typically, an expert supplies a set of domain examples of different types of decisions, called a training set, together with the attributes which describes the examples, and values s/he assigns to those attributes. From the training set, the system induces a set of rules, which are often constructed in the form of a decision tree. The rationale behind these systems is that experts pass on their knowledge to apprentices, through their ability to identify key concepts of the domain and present them as tutorial examples. They are thus more geared to providing cases or examples of their decisions (Michie, 1986).

For experts unaccustomed to formalizing their expertise, the inductive method of KA may be more convenient (Gruber and Cohen, 1987). Generally, induction is "a viable KA method if the problem domain is sufficiently simple and well-defined" (Michalski and Chilausky, 1980:79). The problem with this methodology is that the expert needs to provide examples to cover all possible cases because an incomplete or inadequate set is likely to result in poor rules (Hart, 1986). Since experts cannot account for all they know, there can be no certain way of telling whether the supplied attributes constitute a sufficient set for the construction of a valid decision tree. Furthermore Quinlan, whose original ideas led to the creation of the EXPERT-EASE system, reports that "finding small but adequate sets of attributes for the chess end game king-rook king-knight problems was a considerable task" (1982:201). If this was found to be a problem in such domains as chess that is well defined and understood, it is very likely to be a serious obstacle to the application of this methodology in real world problems.

### 3.2.3.2 Interactive Systems

Interactive systems extract knowledge by carrying out system-driven dialogue with a domain expert. They differ from inductive systems in that they do not have a learning algorithm, but rather, the interrogation of the expert is guided by some specialist knowledge. A number of interactive systems have been developed within the last 5 years. Most of these can be classified into three groups: technique, knowledge or modelling based. I will describe these three subclasses of interactive systems and compare them along two dimensions: [1] breadth - the scope of applicability of the system; and [2] depth - the quality of the knowledge base produced.

The above classification is, of course, not the most comprehensive one. It however serves the purpose of locating the ASKE system within the range of KE tools. Before we go on to a discussion of the various interactive

system types it would be useful to compare these systems along the four knowledge-level dimensions: inference, strategic, task, and domain. A "knowledge-level" description was proposed by Newell (1982) to differentiate the types of knowledge necessary for solving a problem from the symbols used to represent knowledge. Basically, the analysis of an application task at the knowledge-level consists of a specification of the behaviours necessary for solving a problem; the symbol-level analysis consists of a specification of the computational mechanisms for modelling those behaviours. The comparison of systems on the four dimensions can provide useful insights on the kind of knowledge that they utilize for KA.

Knowledge Types	ASKE	KNACK	PROTEGE	OPAL	MOLE	ETS
Inference	yes	no	no	yes	yes	yes
Strategic	no	yes	yes	yes	no	no
Task	yes	yes	yes	yes	yes	no
Domain	yes	yes	yes	yes	no	no

### Technique-based KA systems

The characteristic of the technique-based systems is their use of psychological elicitation techniques as the basis for KA. Because of the domain- and task-independent nature of the techniques, systems based on them have a wide scope of applicability. The technique-based systems can be used to acquire knowledge for a wide variety of applications. These systems are said to have *breadth*. For example, ETS (Boose, 1985) and KITTEN (Shaw and Gaines, 1987) both use the repertory grid technique to acquire knowledge from an expert. A strength of these systems is that they can be used to acquire knowledge for solving any task for which solutions can be enumerated a priori.

By using a particular elicitation technique, a system also inherits all its

limitations. The repertory grid method, for example, is not very good at obtaining procedural knowledge. An expert system, for effective performance, must have both procedural and declarative knowledge. The knowledge bases produced by the repertory grid systems are, therefore, not deep.

### Knowledge-based KA systems

The knowledge-based KA systems employ task-specific methodologies, and they derive their KA power from the use of task-specific knowledge. This knowledge is often in the form of a problem-solving method, a procedure for solving an application task. The method defines "the roles that the task-specific knowledge it requires must play and the forms in which that knowledge can be represented" (McDermott, 1988:228). For example, SALT (Marcus et al., 1985), a system for developing certain types of constructive tasks, knows that for a design specification it must have lists of constraints and fixes for constraint violations. It obtains this knowledge by using the problem-solving method of propose-and-revise.

MOLE (Eshelman and McDermott, 1986) employs the problem-solving method of cover-and-differentiate, which is suitable for certain types of diagnostic tasks. It starts with a set of symptoms supplied by the user. Then, it iteratively obtains candidates that cover or explain the symptoms and information that will differentiate the candidates. Thus, MOLE assumes that: [1] the user can pre-enumerate the hypotheses or solutions that are to be selected; and, [2] s/he can define the problem in terms of covering knowledge. With such an approach, knowledge-based KA systems have succeeded in developing prototype performance systems. These KA systems are therefore said to have *depth*.

By relying on task-specific knowledge, however, knowledge-based systems are rather constrained in their scope of applicability. One of the reasons for

this is that the knowledge tends to in a compiled form leaving very little room for flexibility. The MOLE system, for example, can be used for diagnostic tasks only.

### Model-based KA system

A model-based system is one which performs KA by eliciting a model of the domain which is then used to obtain problem-solving expertise from the domain expert. This approach to KA allows the incorporation of both the knowledge-analysis and task-specific methodologies of KE. A model not only allows the conceptualization of a problem at an abstract level, but also facilitates knowledge elicitation (Motta et al., 1989b). The modelling approach has been successfully applied in KNACK (Klinker, et al., 1987), which builds a model of its domain and then uses this to gather additional knowledge from the domain expert.

Musen (1988), in his PROTEGE and OPAL systems, provides one of the best examples of model-based approach to KA. His methodology explicitly separates the problems of creating models of application tasks from the encoding of the domain-specific knowledge. PROTEGE interacts with a knowledge engineer to build a task model, which is used to automatically generate KA systems like OPAL. Domain experts can then independently use PROTEGE-generated tools to develop performance systems.

A particular strength of the model-based approach is that it can be implemented to provide the system both breadth of scope and depth in the knowledge-base produced. This characteristic has been exploited in the ASKE system, which is described in the following chapters.

### **3.3 A Review of KA Systems**

In this sub-section, I will present a review of five KA systems: ROGET (Bennet, 1985), ETS, MOLE, KNACK and PROTEGE-OPAL. These systems



have been selected because they influenced the design of ASKE. The systems are described in terms of the following five points:

- 1 Description: a short statement about what the system does.
- 2 Category: the classification of the system in the tool tree depicted in Figure 3-2.
- 3 Scope: what types of problems the system tackles.
- 4 Strength: what is the main strength of the system
- 5 Features: special features of the system.

### 3.3.1 ROGET

Description: ROGET helps a domain expert design a knowledge-base for an EMYCIN-based expert system. It conducts a dialogue with the expert to acquire the expert system's conceptual structure, a representation of the kinds of domain-specific inferences that the consultant will perform and the facts that will support these inferences. Finally, ROGET converts each instance and fact into the EMYCIN's object-attribute-value representation, and the support relationships into rules.

Category: Knowledge-based interview system

Scope: Diagnostic task

Strength: ROGET contains a strong conceptual model of diagnosis, which it uses to obtain the conceptual structure of the new application.

Features: ROGET employs a set of domain-independent expectations, abstracted from the problem-solving organizations of existing diagnostic expert systems, to acquire the conceptual structure of the target consultant. The rationale for this is that the kinds of concepts that diagnostic systems employ and base their inferences upon are essentially the same.

### 3.3.2 Expertise Transfer System (ETS)

Description: ETS employs the theory of personal construct psychology to elicit a domain expert's experiential knowledge. In particular, the repertory grid technique is used to elicit, analyse and refine knowledge. A

typical session involves interactively constructing a rating grid of problem solutions and their traits. The rating grid is then transformed into rules.

Category: Technique-based interview system

Scope: Any classification task

Strength: Domain-independent strategy allows ETS to have a wide scope of applicability.

Features: ETS shows how a formal knowledge elicitation technique could be implemented into an automatic KA system to produce fast prototypes.

### 3.3.3 MOLE

Description: MOLE uses task-specific methodology to interactively acquire problem-solving knowledge from the domain expert. It understands the kinds of knowledge that are significant in diagnosis. This knowledge is in the form of the role-limiting (or problem-solving) method of cover-and-differentiate.

Category: Knowledge-based interview system

Scope: Diagnostic tasks

Strength: With its task-specific method, MOLE is able to acquire and produce diagnostic systems or prototypes.

Features: MOLE contains a performance component which checks the knowledge-base for consistency. The performance system provides a means of comparing MOLE's diagnosis with that of the expert's.

### 3.3.4 KNACK

Description: KNACK uses an acquire-and-present method to generate shells, called WRINGERS, that evaluate designs and produce reports. It first acquires a model of the domain, the concepts and vocabulary that experts use in performing their task, and a sample report, a document that exemplifies the output a WRINGER is expected to produce. The domain model and the sample report is then integrated and used to elicit report outlines, phrases and run-time procedures for filling in reports.

Category: Model-based interview system

Scope: Reporting tasks (e.g., writing proposals and progress reports, documenting design decisions, and defining requirements for a product)

Strength: It be used to develops expert systems which produce reports.

Features: KNACK uses a number of knowledge roles, which are represented as templates. "The knowledge role templates define implementation details for a piece of knowledge, and they define the optional parts of a piece of knowledge" (Klinker et al., 1987:75).

### 3.3.5 PROTEGE-OPAL

Description: PROTEGE uses a skeletal-plan-refinement method to build KA systems such as OPAL (Musen, 1988). At the PROTEGE level, knowledge engineers work with domain experts to build models of tasks that can be solved using the method of skeletal-plan-refinement. PROTEGE uses these task models to generate custom-tailored, graphical KA tools (e.g., OPAL) automatically. At the OPAL level, domain experts instantiate the task models to define new applications. The individual tools then translate the instantiated task models into functional knowledge-bases.

Category: Model-based interview system

Scope: Tasks that can be solved by skeletal-plan refinement

Strength: It uses model-based approach to produce KA systems, which can be used to develop prototype performance systems.

Features: PROTEGE is unique in that its final product is not a knowledge-base, but rather, another KA system. The main user of PROTEGE is the knowledge, however, PROTEGE-generated KA tool (e.g., OPAL) can be used by domain experts.

### 3.4 A model for KA

From the previous review it is clear that:

- 1] for a system to have breadth of scope it must have task independent strategies; and
- 2] for depth in the knowledge-base the system requires task-specific knowledge.

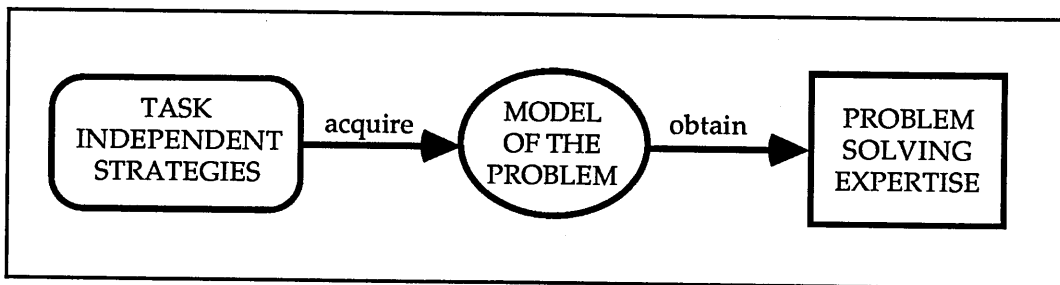


Figure 3-3 A model for bridging the breadth-depth problem

Task-independent strategies such as problem-solving methods and knowledge elicitation techniques are first used to acquire some task-specific knowledge (e.g., task model). The system then acts as a knowledge-based KA system. It uses this knowledge as a guide for interacting with the domain expert to obtain domain-specific knowledge.

It is however possible to integrate the technique-based and knowledge-based KA strategies by adopting a modelling approach, as shown in Figure 3-3. The wider scope of applicability is afforded by the task-independent strategies which are employed to derive a model of the problem. This model provides the necessary task-specific knowledge for acquiring the problem-solving expertise from the domain expert.

The model depicted in Figure 3-3 is not far from the one used in KNACK and PROTEGE. In these systems, the task-independent strategies consist of the problem-solving methods of acquire-and-present and skeletal-plan-refinement, respectively. A neat solution would then be to merge these

two systems into one. The final system would look like Figure 3-4. The task-independent strategies are replaced by the two problem-solving methods. a simple inference strategy can be used to select an appropriate problem-solving method, which can then be used to acquired the task model of the problem.

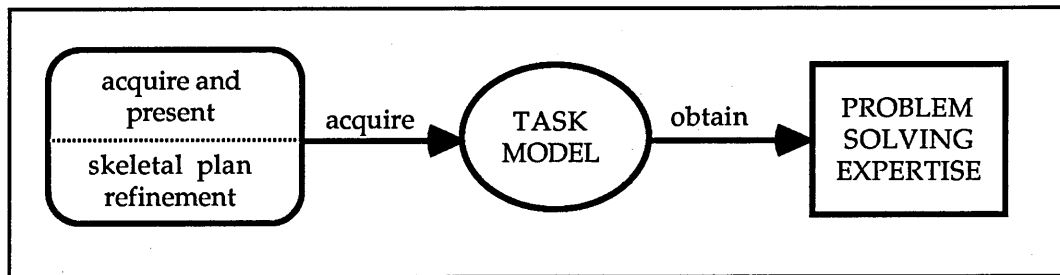


Figure 3-4 Merging KNACK and PROTEGE systems

KNACK uses acquire-and-present method to acquire a domain model. It then interviews the domain expert, guided by the model, to develop a prototype expert system. PROTEGE uses the problem-solving method of skeletal-plan-refinement to acquire a task model of the problem consisting of a graphical tool. This tool can be used by domain experts to build expert systems.

There is a serious flaw in the system depicted in Figure 3-4: the two problem-solving methods produce two incompatible models of the domain. This situation can be rectified by using a same-level description for different problem-solving strategies. This is the course of action taken in ASKE (see Figure 3-5). In its design, the modelling approach has been implemented to give it the power of knowledge-based KA systems and the scope of applicability of technique-based KA systems. The task-independent strategies used are the task characteristics. For example, diagnosis of an "object" is described by the association between "signs of malfunction" it exhibits and "cause of malfunction". Thus, problem-solving knowledge for diagnosis must contain descriptions of these three basic concepts.

The level at which the task characteristics are described is much too general. To help in conceptualizing, an example task model of a similar problem is presented. For example, in medical diagnosis, the expert is shown a previously built task model for diagnosis in medicine. This method of using previous cases to develop new applications has already been successfully employed in ROGET.

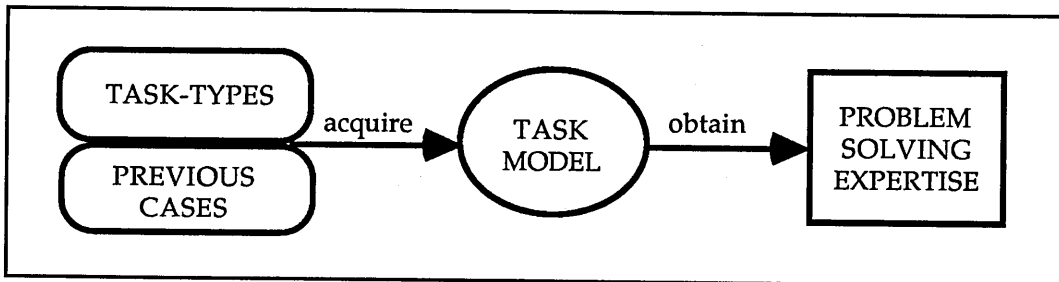


Figure 3-5 ASKE's model for KA

ASKE contains general knowledge about various analysis tasks and a library of abstracted knowledge-bases. An appropriate task-type and a previous case are used to acquire the task model, which is then used as a guide for obtaining the problem-solving expertise from the domain expert. The first half of KA is carried out by question-answering. In the second half, graphical interface of mouse and menus is used.

# Chapter 11

## FUNDAMENTAL CONCEPTS

*One must even know first to be able to know more later.*

### 4.1 Introduction

In ASKE, KA is viewed as a modelling activity and the system devotes its initial efforts to developing a task model for the problem. The framework for the model is derived from the characteristics of the application task. The expert is provided with a closely related task model, from a previous case, to help him/her in filling-in the details. The two main features of the system are, therefore, uniformly described task types and use of previous cases (see Figure 3-5). With these features, ASKE is able to achieve a greater scope of applicability. The task model provides a means of creating rich knowledge-bases. ASKE's real strength, however, is in the representational scheme of templates. All knowledge that ASKE has or obtains, is represented in the template formalism.

In the rest of the chapter, the three central elements of ASKE: templates, task-types and previous cases, are described. First, a rationale for the use of task types as task-dependent strategies is presented, followed by a description of the task characteristics of the various analysis tasks. Second, a case for the use of previous cases as exemplars is made. Third, the template scheme for knowledge representation is described, and various template types are introduced.

## 4.2 Task characteristics as task-dependent strategies

ASKE's expectations of how problem-solving expertise is organized is based on Clancey's (1985) notion of heuristic classification. His model of heuristic classification provides a precise set of terms and relations by which problem-solving tasks can be characterized. According to the model, the heuristic classification method can be applied to solve those tasks for which all possible solutions of the problem can be enumerated a priori. Classification involves the selection of one among a predetermined set of possibilities as the appropriate description of a situation.

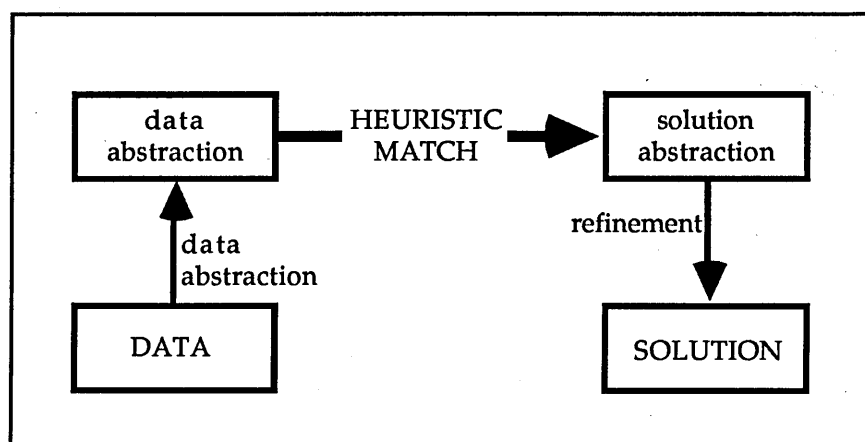


Figure 4-1 Heuristic classification (From: Clancey, 1985:296)

The inference structure for heuristic classification (Figure 4-1) consists of the following components: data, data abstractions, solution abstractions and solution which are related systematically by different kinds of relations and rules of inference. From the KA point of view, Clancey's model identifies the important elements (pieces of knowledge) that make up the problem-solving expertise. Furthermore, by relating the heuristic classification to tasks, the model puts forward, implicitly, a specification for a generic tool for KA.



According to Clancey (1985), the classification method is suitable for solving analysis tasks, because it is possible to pre-enumerate their solutions. The main type of tasks within this category, for which expert systems have been developed, are: debugging, diagnosis, interpretation and selection. This, however, does not imply that all diagnostic tasks, for example, can be solved by heuristic classification. For instance, in medical diagnosis only routine problems, which are characterized by unique mapping between disease and symptoms, are solvable by the classification method (Pople, 1982). For non-routine medical diagnosis problems, in which there is more than one disease explaining the symptoms, the problem solver has to formulate (or construct) a solution. Pople calls these problems "ill-structured". It would, therefore, be more accurate to say that the classification method is suitable for solving structured problems for which solutions can be explicitly enumerated and problem descriptions can be mapped directly to solutions by pre-existing links.

In ASKE, the analysis tasks have been characterized in terms of the important concept categories, on the basis of Clancey's model, to provide task-dependent templates for task modelling. Typically, a task is described as having a data and a solution category. The system's role is to identify: [1] abstraction hierarchies for elements from which a solution is selected; [2] abstraction hierarchies for the data that bear on the selection process; and [3] the heuristics that link elements from one hierarchy to those in another.

The reducing of the KA process to the identification of concept categories is not without justification. Most expert systems have knowledge organized in identifiable concept categories. Consider the domain of medicine, for which "more expert systems have been developed than for any other single problem area" (Waterman, 1986:40). Medical systems typically consists of a category for observation (e.g., symptoms and test

results) and a category for disease. For example,

"The knowledge-base underlying both INTERNIST systems is composed of two basic types of elements: disease entities and manifestations" (Pople, 1985:185).

This is entirely justifiable, because

"... medical diagnosis rests on the premise that there is a unique mapping (a function) from sets of manifestations to disease entities" (Simon, 1985:76).

The rest of this section presents the details of the four analysis tasks.

#### 4.2.1 Selection

Selection is the most basic of the analysis tasks. Selection systems, typically, identify an object from a set of objects on the basis of some criteria (Figure 4-2). For instance, Demaid and Zucker (1988:292) define the materials selection problem as "the need to choose a material from which to manufacture an artifact". What is given is a set of materials with their specifications (e.g., Material X: hydrolysis resistant, time immersed in water @100°C to give a 50% drop in tensile strength). The criteria for selecting is a list of material attributes which are required to enable the product to function successfully (e.g., ability to stand exposure to water at 100°C for short times).

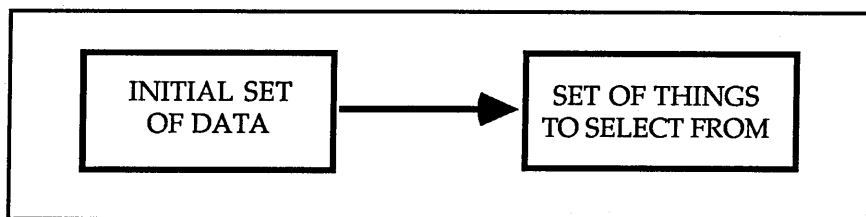


Figure 4-2 Task characteristics for selection

### Examples

LOKE (Sørensen and Nordhus, 1987) selects drilling bits for oil exploration on the basis of geology of the area and the past performance of given bits in similar situations. LOKE thus has knowledge of different types of drilling bits and records of their performance capabilities under different operational conditions.

PERITUS (Swindells and Swindells, 1985) selects engineering materials. The user specifies his/her requirements for each of the general characteristics, such as weldability, corrosion resistance, fluidity, etc. On the basis of required characteristics, the system generates a short list of candidate materials.

#### **4.2.2 Debugging**

Expert systems that perform debugging find remedies for malfunctions. Debugging tasks can be characterized as having "type of malfunction" and "object" as two data categories, and "remedial action" as a category of possible solutions (Figure 4-3). Often, debugging systems incorporate a diagnosis component to uncover the cause(s) of malfunction. For example, in medical expert systems the disorder is diagnosed and then a treatment is prescribed to remedy it. Though the general problem of debugging is quite difficult and requires designing remedies and evaluating them by predicting their effectiveness, many current debugging systems rely on simple tables of associations between types of malfunctions and particular remedies (Waterman, 1986).

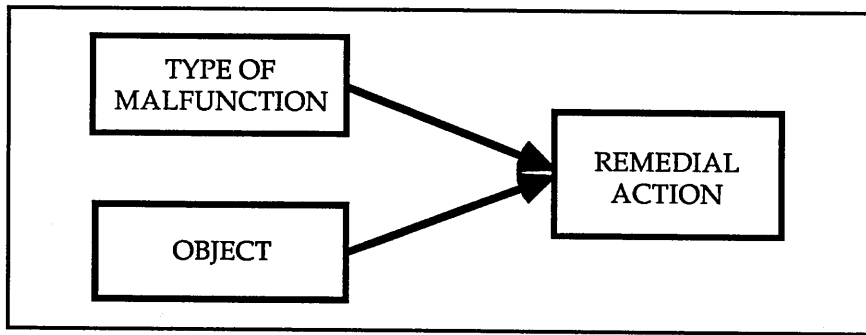


Figure 4-3 Task characteristics for debugging

### Example

ACE (Vesonder et al., 1983) identifies trouble spots in telephone networks and debugs them by recommending appropriate repair and rehabilitative maintenance. ACE locates faulty telephone cables and it decides whether they need preventive maintenance and selects the type of maintenance most likely to be effective.

### 4.2.3 Diagnosis

Diagnosis systems infer system malfunctions from observables. These systems typically relate observed behavioural irregularities with underlying causes by using a table of associations between behaviours and diagnoses (Figure 4-4).

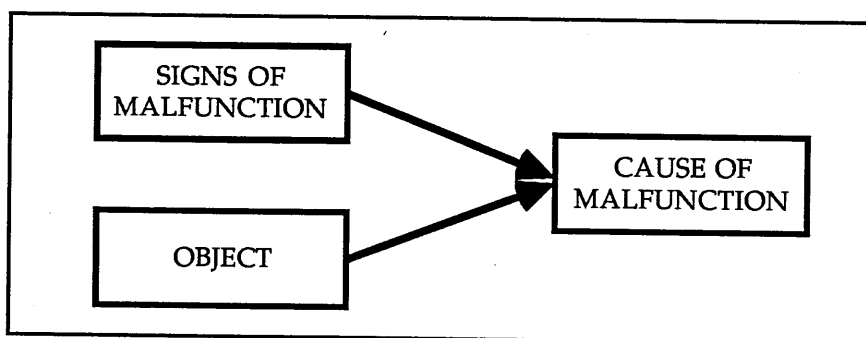


Figure 4-4 Task characteristics for diagnosis

Bennett (1985:52) found, from his survey of diagnostic systems, that these systems "base their inferences on similar types of evidence... *and they advise their users on a 'problem' of some sort (e.g., a bleeding disorder). Furthermore, some of the diagnostic systems are also concerned with recommending a 'repair' that would rectify this problem (e.g., an appropriate drug regimen)". Hence, we often find that diagnosis and debugging are performed successively, as shown in Figure 4-5.*

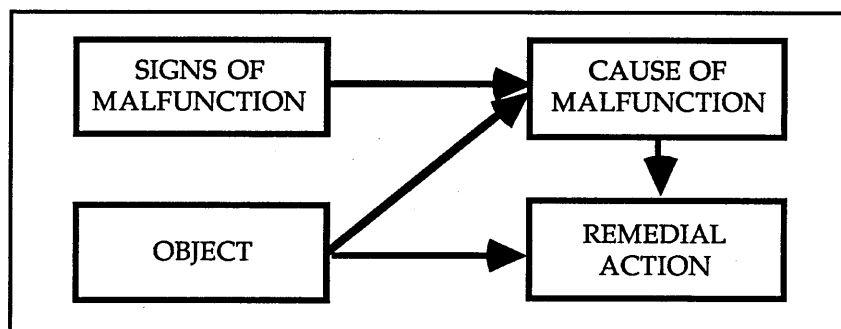


Figure 4-5 Task characteristics for diagnosis and debugging

### Examples

ABEL (Patil et al., 1981) diagnoses acid-base and electrolyte disorders in patients by applying knowledge about the diseases and the symptoms they produce. The system contains data about the patient as well as knowledge about the relations between various disease states. Knowledge is represented within a causal network, a type of semantic net specifying cause-effect relations between diseases and findings.

DART (Bennet and Hollander, 1981) diagnoses faults in computer hardware systems using information about the design of the device being diagnosed. The system works directly from information about the intended structure and expected behaviour of the device to help find design flaws in newly created devices.

#### 4.2.4 Interpretation

Interpretation systems infer situation descriptions from observables. An interpretation system explains observed data by assigning to them symbolic meanings describing the situation or system state accounting for the data (Figure 4-6).

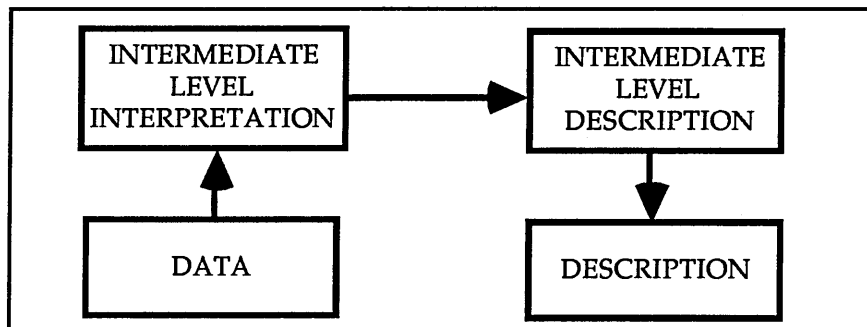


Figure 4-6 Task characteristics of interpretation

Archaeological interpretation, for instance, involves going from initial propositions to terminal propositions or interpretations via a series of intermediary propositions (Gardin, 1980). Gardin's model of archaeological reasoning has been implemented in KIVA (Patel and Stutt, 1989a), an expert system for interpreting settlement sites. From the distribution of objects found at an archaeological site, KIVA generates a description of the activities that took place within each of the habitation areas within a site from the initial data.

#### Examples

CRYALIS (Engelmore and Allan, 1979) infers the three-dimensional structure of protein molecules from x-ray crystallographic data. The knowledge-base consists of several layers of data and hypothesis with mappings between the two. The basic input data is the electron density map. This is abstracted at three levels: peak, skeletal and segment. The

hypothesis part of the knowledge-base consists of three layers: atomic, superatomic and stereotypic.

Dipmeter Advisor (Smith and Baker, 1983) infers subsurface geological structure from measurements of the conductivity of rock in and around a borehole as related to depth below the surface. From the basic data, the dipmeter expert makes inferences regarding the geological history of deposition, the composition and structure of the beds, and the optimum locations for future wells.

### 4.3 Previous Cases as Exemplars

Experience constructing expert systems reveals that the identification of the vocabulary of a problem representation dominates the early KA dialogues between knowledge engineers and experts. Identifying this terminological framework is an important prerequisite to the expression of the rules, making this activity a crucial first step in the design of an expert system prototype. But, Buchanan and Shortliffe (1984:150) caution:

"the most difficult aspect of KA is the initial one of helping the expert conceptualize and structure the domain knowledge for use in problem solving".

Historically, a knowledge engineer has helped the expert overcome these difficulties. Through lengthy discussions, the knowledge engineer helps develop a representation of the expertise, first by suggesting different expert system organizations and then encoding them in an appropriate system-building tool. The knowledge engineer identifies the type of task the system is to perform and suggests the types of domain terms to include in the system, focusing the expert's attention on the essential elements of constructing an interactive consultation program.

The manual method of KA provides a good model for designing

automated systems for KA. In ASKE, the knowledge of task characteristics is utilized as a guide of terms necessary for representing the problem. Previous cases are employed as exemplars; their function is to draw the expert's attention to the plausible elements for the new application. Before giving an example of how ASKE uses previous cases, I would like to look into the nature of expertise so a better case can be made for the usefulness of previous cases as exemplars.

#### 4.3.1 The nature of expertise

The transition from the status of a novice to that of an expert is achieved through experience. Basically, expertise seems to be acquired in three more or less distinct stages (Fitts, 1964). In the first stage, an individual learns which actions to take in a given situation by either observation of performance or instructions. This is the cognitive stage of learning. The second stage is called the associative stage during which the relationships learned in the cognitive stage are practiced until the actions become fluent and accurate. In the third stage the relationships are compiled through repeated practice. This is the autonomous stage where the individual's actions, as a result of overpractice, become automatic and are performed "without thinking" (Johnson, 1983).

At the end of the third stage of learning, the individual acquires the status of an expert which means that s/he is able to "perform the actions appropriately, proficiently, and effortlessly" (Musen, 1988:26). Besides enabling the individual to do a task more efficiently and accurately, experience plays another important role: it enhancing the general problem-solving knowledge so that the skill can be applied to novel problem solving. For example,

"Individual experiences act as exemplars upon which to base later decisions. Analogies to previous cases guide and focus later decision making" (Kolodner and Simpson, 1986:100).



Kolodner refers to the process by which knowledge from a previous case is transferred to a current one as similarity-triggered analogical reasoning. In this kind of reasoning, past cases are indexed and stored in order to facilitate later retrieval and use in solving new and similar problems. A case typically contains a description of the problem, some of the intermediate steps to the solution of the problem, and the solution itself. This form of reasoning aids in problem classification in two ways: [1] predicting additional features to be investigated; and, [2] pointing out alternate or additional classification.

#### 4.3.2 Using previous cases as exemplars

ROGET (Bennet 1985) was the first KA system to use previous cases to acquire knowledge from domain experts. It uses examples of the problem-solving organizations of existing diagnostic expert systems to guide the expert in determining the problem task type of the new application. An extract from a ROGET session (Figure 4-7) shows how ROGET obtains the description of the task that the new diagnostic system, MYCIN75, will be performing. ROGET determines the problem task type by providing the expert with examples of previous expert systems. The examples are simple phrases constructed from answers during previous ROGET sessions. They indicate to the expert the required format for answering the question.

4) Enter a simple phrase that describes the main task  
MYCIN75 will perform during a typical consultation.  
\*\* examples

In CLOT80, the main task of the system was to  
diagnose the hematological disorders of a patient.  
In PUFF78, the main task of the system was to  
diagnose the pulmonary dysfunctions of a patient.

\*\* recommend what drugs to prescribe for the  
infectious diseases of a patient.

Figure 4-7 ROGET dialogue showing use of previous cases

ROGET uses pre-stored examples and the same ones are used everytime. Indeed, there is no need to vary examples as the target system is always a diagnostic one (because of the limited scope of the system). ASKE started with three example knowledge-bases: interpreting burial sites (archeology), diagnosing foot problems (medicine) and selecting marketing strategies (corporate planning). With every new application, a new example is acquired (see Section 6.6 of how this is done). An example that will be used in the target application is selected at run-time, on the basis of closeness of the example to the target (see Section 6.2.4).

ASKE uses previous cases as exemplars for organizing the new knowledge-base. In the development of a task model, the expert is provided with an example of a built system which is similar to the new one. Figure 4-8 represents a scenario of an ASKE session showing the acquisition of a task model for diagnosing pressure sores in the domain of nursing. (The complete session is given in Appendix C.) The example of diagnosing foot problems is selected and used to guide the user (an expert nurse) in specifying the important concept categories for describing the task of diagnosing pressure sores.

<p>For problem-solving in analysis tasks, there are basically two categories of concepts: data and solution. The task of diagnosis involves inferring system malfunction from observables.</p> <p>Data Types:  Signs of Malfunction  Object (which is defective)</p> <p>Solution Types:  Causes of Malfunction</p>
<p>ASKE was used to develop a knowledge-base in the domain of medicine for doing diagnosis of foot problems. The aim of this application was: diagnose foot problems from symptoms.</p> <p>The Data types were:  symptoms (signs and symptoms of foot problems)  patient (person with foot problem)</p> <p>The Solution types were:  problems (the cause of foot problem)</p>
<p>What is the main category of data for the diagnosis of pressure sores?  =&gt; patient.state</p> <p>Comments for patient.state  =&gt; The physical, physiological and psychological condition of the pressure sore patient.</p> <p>What is the object of diagnosis? [Type nil, if none]  =&gt; pressure.sore.patient</p> <p>Comments for patient.state  =&gt; Person prone to pressure sores.</p> <p>What is the main category of solution when doing diagnosis of pressure.sores?  =&gt; care.plans</p> <p>Comments for care plans.  =&gt; The course of treatment for pressure sore patients.</p>

Figure 4-8 ASKE session showing use of previous cases

#### 4.4 Template scheme for representing knowledge

All knowledge that ASKE has, or acquires, is represented in one of four template types: general, acquisition, working and reference. A template is a framework for acquiring and representing problem-solving expertise. It is implemented in a frame language. A frame is a "data structure for representing a stereotyped situation" (Minsky, 1975). A frame language consists of a network of structured nodes connected by relations and organized into a hierarchy. Each node represents a concept that may be described by attributes (or slots) and values (or fillers) associated with the node. Nodes that are low in the hierarchy automatically inherit properties of higher-level nodes.

Structurally, a template is similar to a frame (Figure 4-9). It is described by attributes and values. Each attribute corresponds to a piece of information necessary for developing an expert system. ASKE's interaction with the expert is guided by the attributes: ASKE seeks to fill each of the attribute slots. The filler (or value) is either domain concepts, which are internally represented as frames, or other expertise related knowledge.

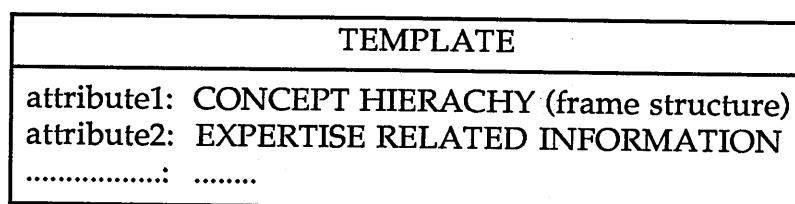


Figure 4-9 Structure of a template

There are two characteristics of a template which distinguishes it from a frame. First, a template is not just a property list. The properties provide expectations of what and when to acquire domain knowledge from the expert. Hence, embedded in the template structure is a methodology for

knowledge acquisition.

The second difference between templates and frames is that while the latter are hierarchically arranged, the former aren't. The four template types are described at the same level, there is no inheritance mechanism between them. They each address a different stage of knowledge-base development. In fact, the four templates, between them, provide the necessary and sufficient structure for specifying a knowledge-base.

#### 4.4.1 General Template

The most basic template is referred to as a general template (GTEMP). It provides a description of the problem at the domain (or discipline) level. It contains personal details of the domain expert and general information about the new knowledge-base. The latter includes domain classification, which is used for selecting reference and acquisition templates for further KA, and information about the project, (e.g., the reason for building the system and who it is designed for).

An example of a GTEMP, for archaeology, is shown in Figure 4-10. The attributes (upper-case) specify the information categories required for a top-level description of the problem. Our experience in developing an expert system for interpreting settlement sites from archaeological remains (Patel and Stutt, 1989a) suggest that these categories are important for understanding the task application. Besides the details about the domain expert which is obviously useful to have, we need to classify the domain for a better understanding of the problem. This involves knowing the name of the domain (or discipline), the task type and the area of the application. Information about project aims and users of the target system is useful for latter stages of producing a usable performance system.

General Template (GTEMP)
EXPERT: Jitu
ADDRESS: HCRL, The Open University, Milton Keynes.
TELEPHONE: 0908 652574
DOMAIN: Archaeology
TASK: Interpretation
SPECIALIST AREA: Settlement.site
TASK DESCRIPTION: Inferring activity areas in a settlement site from archaeological data
PROJECT AIMS: Provide expertise on interpreting settlement sites to archaeologists
USERS: Archaeologist

Figure 4-10 General Template for Archaeology

The user is prompted for values of the given attributes. These attributes are identified to serve an important role in knowledge-base development. For instance, the values of task and specialist areas are used for selecting acquisition and reference templates. The personal details are for reference purposes. The values for project aims and users are acquired because of their potential use in designing user-interfaces.

ASKE contains hand-crafted knowledge-bases for the domains of Archaeology, Medicine and Strategic Planning. These knowledge-bases guide the automatic acquisition of new applications within these domains, and also in other domains. For each of these domains, ASKE has a GTEMP. New ones are created for other domains at run-time. Basically, the different GTEMPs differ in the values of the attributes. The most important attributes are "task" and "specialist area", they hold information on the domain classification which is used for accessing templates for further KA.

The most important function of the GTEMP is to acquire information from the user for selecting acquisition and reference templates (see Section

6.2.4 for descriptions of the selection procedures). These templates are needed to develop the model of the task.

#### 4.4.2 Acquisition Template

In the modelling approach, the first step in KA is to build a task model, that is, a description of the main concept categories and how they are related. ASKE does this with the help of the acquisition template (ATEMP). The ATEMP is the most basic template for KA contains task-specific knowledge. An ATEMP provides expectations of the kind of knowledge that goes into a task model. A task model for analysis problems consists of two main concept categories: data and solution. The data category specify the main input to the system. The solution category describes what will be output by the system. Each of the ATEMPs, hence, contains task-specific information derived from their characteristics (described in the previous section).

An ATEMP is selected on the basis of information, in GTEMP, on the task the new application will perform. The selection procedure maps the task-type to ATEMPs. This is possible as ASKE contains an ATEMP for every specified application task.

TASK TYPE	DATA	SOLUTION
Debugging	Type of Malfunction Object	Remedial Action
Diagnosis	Signs of Malfunction Object	Cause of Malfunction Remedial Action
Interpretation	Data	Solution
Selection	Initial Data	Selection

Figure 4-11 Concept categories for analysis tasks

An ATEMP for every task-type will have the identified concept categories as attributes. Diagnosis has an extra category for 'repair'. This will be utilized only if the diagnostic application will be expected to prescribe a remedy also.

ASKE has ATEMPs for four application tasks: debugging, diagnosis, interpretation and selection. The characteristic of these tasks is that their solutions can be enumerated a priori. Furthermore, they can be solved by the problem-solving method of heuristic classification. Thus, implicit in the ATEMPs is the reasoning strategy for solving the tasks. For example, the system manipulates the input (data) according to the rules of inference (obtained from the relationships between data and solution categories) to output the result (solution). Figure 4-11 summarizes the concept categories for these tasks.

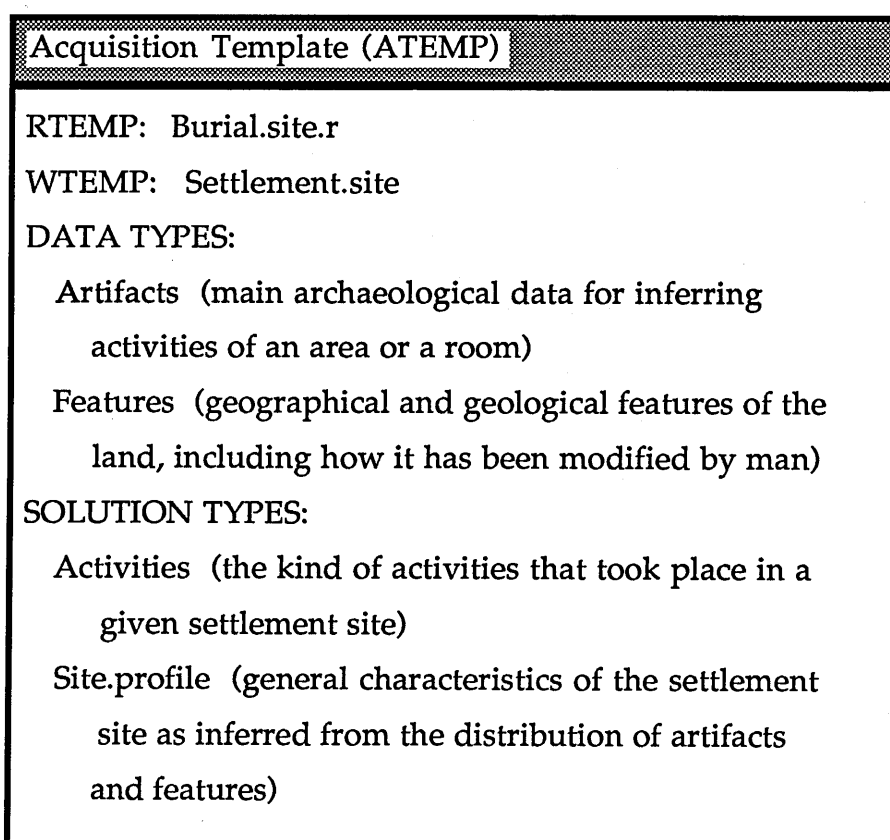


Figure 4-12 ATEMP for archaeological interpretation

The attribute RTEMP is a pointer to the template which was used as an exemplar for developing the task model. The WTEMP attribute holds the name of the template which contains the new knowledge-base. DATA and SOLUTION types have two values each. These values are the main concept categories, which are used in the creation of a WTEMP for the application. In brackets, is a brief description of the concept.



The concept categories are held as attributes of respective ATEMPs. The user is prompted on the basis of the categories. For example, the ATEMP for interpretation has attributes for "data" and "solution". Figure 4-12 shows an ATEMP for archaeological interpretation. The values for data and solution types provide a model for interpreting settlement sites.

The present implementation of ASKE does not allow the user to create new ATEMPs. The user hence can only develop applications for one of the four described problem types. However, in future extensions, flexibility in this direction is envisaged (see section 8.2.2).

#### 4.4.3 Reference Template

In the specialist areas for which knowledge-bases have been developed, ASKE will have a reference template (RTEMP). At the end of the KA session, ASKE abstracts important knowledge from the new knowledge-base and represents it in an RTEMP. RTEMPs, hence, are the previous cases which serve the role of exemplars in KA.

An RTEMP consists of an abstracted knowledge-base. This is automatically extracted from the current knowledge-base at the end of the session. The contents of an RTEMP include information for identifying the knowledge-base (e.g., domain, task, specialist area) and the main concept categories describing the problem. Figure 4-13 shows an example of an RTEMP, from the domain of Archaeology, for interpreting Burial Sites. The domain, task and specialist attributes indicate where the RTEMP is derived from. The concept categories, with their comments (in brackets), describe a model for interpreting Burial Sites. The RTEMP, hence, can be employed as an exemplar for related problems. Chapter VI describes how this RTEMP was used in the development of the knowledge-base for interpreting Settlement Sites.

Reference Template (RTEMP)
EXPERT: Jitu
DOMAIN: Archaeology
TASK: Interpretation
SPECIALIST AREA: Burial.site
TASK DESCRIPTION: Infer habits of past cultures from from their burial remains
DATA TYPES:
Artifacts (any man-made object)
Ecofacts (any natural object - i.e., floral or faunal)
Features (any archaeological data that is not an artifact or an ecofact)
SOLUTION TYPES:
Hypotheses (the plausible interpretations of burial sites)

Figure 4-13 An RTEMP for Burial.sites

The main data categories for interpreting Burial Sites are: artifacts, ecofacts and features. However, for inferring activities of a Settlement Sites, only artifacts and features are used. An example knowledge-base in a similar application area can thus aid in focussing attention of the important concepts necessary for describing the new application.

The central role of an RTEMP is to guide the domain expert in conceptualizing his/her problem-solving expertise in the format required for knowledge-base development. An RTEMP and an ATEMP are employed in the acquisition of the task model for the problem. The ATEMP provides expectations of what knowledge is required. However, because of the very abstract level of this knowledge, an example of a task model in a similar problem (i.e., RTEMP) is presented. For instance, to an archaeologist, 'data categories' would be more meaningful if supplied with an example: 'data categories, in the domain of archaeology, for interpreting burial sites include artifacts, ecofacts and features'.

#### 4.4.4 Working Template

A working template (WTEMP) is used for holding the new knowledge-base. Relative to other templates, a WTEMP describes the problem at the level of specialist area (i.e., the new application). Unlike GTEMP and ATEMP, a WTEMP does play a direct role in KA.

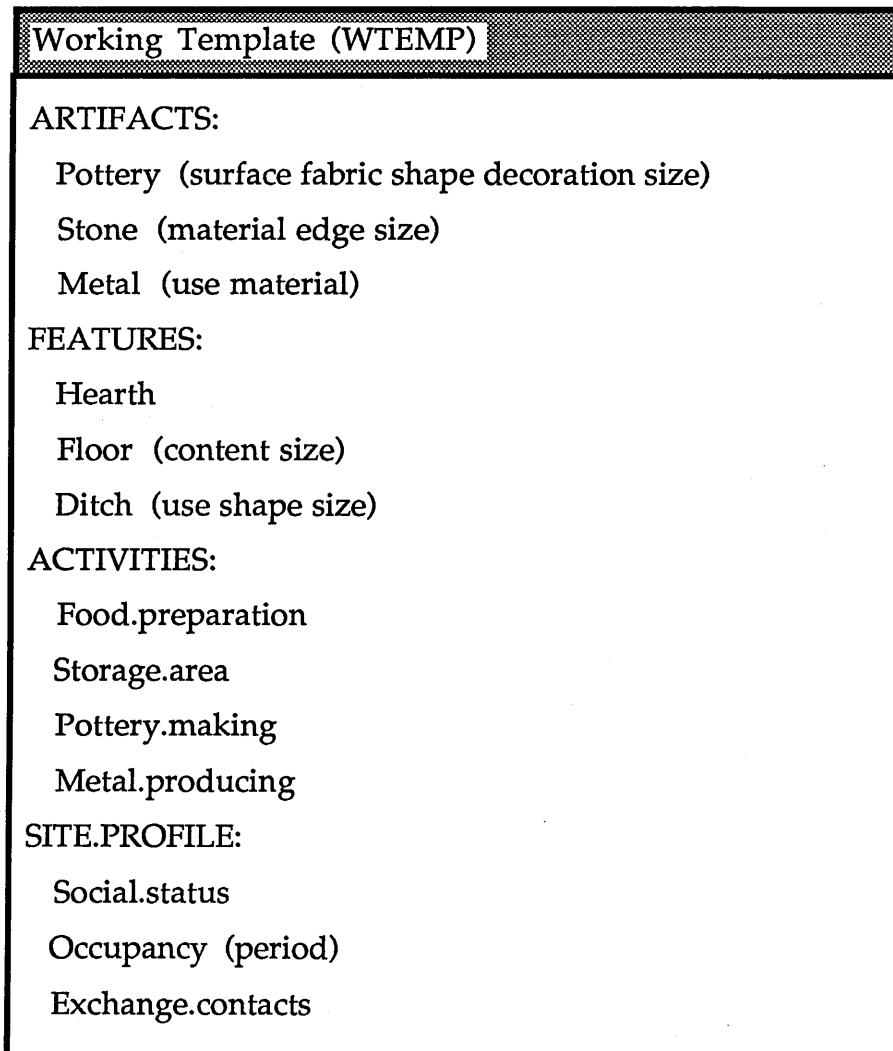


Figure 4-14 A WTEMP for Settlement Sites

Each of the main concept categories, obtained by the ATEMP, are made into slots (or attributes) of the WTEMP. Each slot has a set of concepts as values each of which is represented as a frame. Each concept, furthermore, has attributes, shown in brackets. The attributes of the concepts are displayed in the WTEMP for information purposes; they are however held within the concept frame. For example, *pottery* (value) is an *artifact* (slot). Concept pottery has attributes: surface, fabric, shape, decoration and size.

ASKE creates a new WTEMP for every application. The initial contents of the WTEMP is the task model. The concept categories obtained with the ATEMP provides the attributes for the WTEMP. For example, Figure 4-14 depicts a WTEMP for "settlement sites" which was created from the ATEMP for interpretation (Figure 4-12). The values of these attributes are the concepts for the category designated by the attribute. Each concept is stored in a frame structure.

#### 4.4.5 What's in the template scheme

According to the Knowledge Principle (Lenat and Feigenbaum, 1987), a program must have a great deal of task-specific knowledge in order to exhibit intelligent understanding and action at a high level of competence. This knowledge, however, cannot be obtained from nothing, one must start with some minimum knowledge.

According to Schank and Abelson (1977), our knowledge of activities like going to a restaurant and attending lectures is organized around scripts. A script is a high level knowledge source describing a stereotyped sequence of actions. It provides a conceptual framework, which generates expectancies, and guides plans for actions. For example, a restaurant script can be used in understanding a story on visiting a restaurant. The restaurant script will interpret and organize new information, it will fill in the gaps, and help us understand implications and presuppositions.

A template, like a script, contains prior knowledge structures, which are used to generate expectancies. These structures are acquired by abstracting and generalizing from experiences from others. In terms of memory structure, therefore, templates are closer to MOPs (Schank 1982) than scripts, which consists of specific structures acquired from direct repeated experience. For example,

"In *script-based memory*, what we know in a given situation comes from what we have experienced in more or less identical situations. More general structures (templates and MOPs) allows us to make use of information originally garnered from one situation to help us in a quite different situation. General knowledge structures save space and make information experienced in one situation available for use in another. One disadvantage of the script-based method is its lack of usability in similar but nonidentical situations."  
(Schank, 1982:9)

The template scheme provides a convenient way of representing general and task-specific knowledge. ASKE exploits this scheme for KA. The GTEMP and ATEMP contain general and task-specific knowledge, respectively, abstracted from our experiences at building expert systems. This knowledge provides expectations of what new knowledge to acquire. We use our past experiences in solving (or understanding) new situations. RTEMPs represent past cases which act as exemplars for new applications. Finally, the WTEMP is used for representing the new knowledge-base.

#### 4.5 Summary

KA is difficult because experts rarely have an opportunity to reflect on their thinking processes to the extent and form required for the construction of an expert system. ASKE contains three features which help in facilitating the encoding of expert knowledge.

- 1 In templates, ASKE has a uniform scheme for representing various kinds of knowledge which are used for driving KA.
- 2 Task characteristics which provide expectations of the kind of knowledge that goes into the task model of the problem.
- 3 Use of previous cases to help the expert focus his/her attention on the important concept categories required in the task model.

## Chapter 5

# THE ASKE SYSTEM

*If none of the tools you normally use works, build a new one.*

### 5.1 Introduction

ASKE is a model-based KA system. It is designed for use by domain experts for developing prototype expert systems. It is suitable for building systems which do analysis tasks, i.e., problem types for which it is possible to enumerate all solutions a priori.

The system is built on the premise that the KA process can be quickened by helping the expert organize his/her domain knowledge. This intuition is based on the experience we gained in building an expert system for interpreting archaeological sites (Patel and Stutt, 1989a). The hardest aspect of the system development was the recognition of the important concept categories for the task and how these were inter-related. Once we had the model of the task, the knowledge elicitation became just a routine task.

The system is designed with the aim of providing the expert with an "easy to use" tool. Thus, a major part of the system design consists of an interface for encoding knowledge. The aim of this chapter is to give an overview of the ASKE system. I will first present the methodology implemented in ASKE, and then, describe the architecture of the system giving functional details of its various features.

## 5.2 ASKE Methodology

ASKE acquires problem-solving expertise from the domain expert in two stages. First, a model of the task the new application will be performing, is obtained. Second, this task model of the problem is used for guiding the expert in encoding his/her domain expertise. Figure 5-1 illustrates how ASKE develops a prototype knowledge-base. The example is from the domain of archaeology and involves the development of a knowledge-base for interpreting settlement sites from archaeological data. This knowledge-base was developed using ASKE, and the processing details are described in Chapter VI.

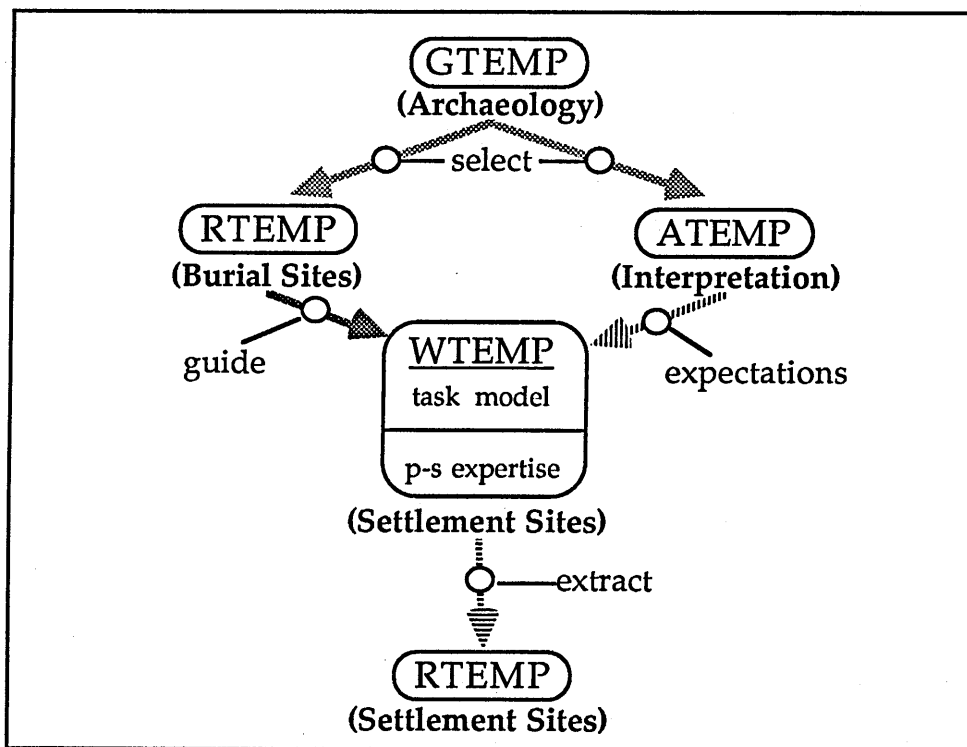


Figure 5-1 Stages in the development of an application with ASKE.

KA starts with GTEMP. From the information acquired, ASKE selects an ATEMP and an RTEMP. The former is employed to acquire the task model of the new application; the latter contains a task model of a similar application and is used to guide the user in building the new task model. The knowledge-base is held in the WTEMP. At the end of the session, ASKE automatically abstracts useful information from the new knowledge-base and stores it in a newly created RTEMP, for later use.

### 5.2.1 Stage One

In the first stage, the interaction between ASKE and the expert is via question-answering. The aim of this stage is to acquire the task model for the new application. This is carried out in the following three steps.

[1] The KA session starts with the problem definition. This involves identifying the domain, the task and the area of application. First, the GTEMP, which has the name of the domain (archaeology), is invoked. The domain expert is prompted for domain classification and information about the project goals.

[2] On the basis of the information in the GTEMP, ASKE selects an ATEMP (interpretation) and an RTEMP (burial sites). The ATEMP contains task-specific knowledge, and it provides expectations of the kind of knowledge required for the task model. The RTEMP holds an abstract knowledge-base in a related application. It is used as an exemplar for building the new task model.

[3] The ATEMP and RTEMP are used to obtain the main concept categories, which define the task model. The domain expert is first told what knowledge is required, and then s/he is presented with an example task model in a related application. Finally, s/he is prompted for the relevant knowledge on the basis of the ATEMP. The task model is a general description of the application task, and it consists of the concept categories for describing the task and the relationships between them.

### 5.2.2 Stage Two

In the second stage, the expert encodes his/her domain expertise into the knowledge-base via a graphical interface. The task model is used in the acquisition of the *facts* and the *rules* part of the knowledge-base. This stage of KA is carried out in the following three steps.



[1] The task model is held in the WTEMP (settlement sites). The first objective is to acquire the concepts and their attributes for every concept category in the task model. Various facilities are provided to facilitate the encoding of the domain concepts. The concepts are stored in their respective hierarchies and form the *facts* part of the knowledge-base.

[2] The next objective is to obtain the rules of inference. The rules describe the heuristic associations between concepts. The domain expert is asked to identify the relationships between concepts in different categories. The actual direction of the relationship is inferred from the underlying heuristic classification problem-solving method.

[3] For every heuristic association between concepts, ASKE automatically creates an if-then rule. The rules, at this stage, are quite simple and non-functional. To make the rules functional, the expert is asked to edit them. The final output is a set of rules which can be mapped directly into KEE or KEATS rule formalism.

At the end of the session, an RTEMP is created by abstracting information from the WTEMP (knowledge-base for interpreting settlement sites). The RTEMP will be stored for future use.

### 5.3 Overview of ASKE

The overall architecture of the system is as depicted in Figure 5-2. There are two interfaces to ASKE: Aske<sup>1</sup> and Rulemaker. The interfaces are coordinated by the control unit, called Cerveau<sup>2</sup>. It has access to the set of templates, which form the basis for interrogating the expert.

---

<sup>1</sup> To distinguish from the system (ASKE), the interface (Aske) is written in lower-case.

<sup>2</sup> The brain (in French).

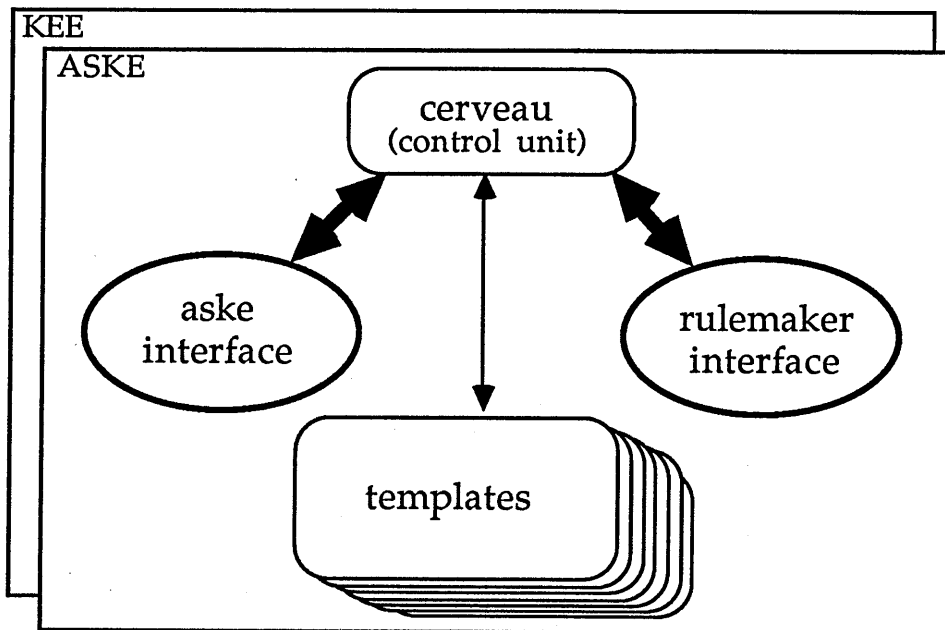


Figure 5-2 Overview of ASKE

ASKE is implemented in KEE. It has two interfaces: Aske and Rulemaker. These interfaces contain various features which allow the encoding of domain knowledge. The movement between the interfaces is controlled by Cerveau, which also has access to the set of templates.

### 5.3.1 Implementation Details

ASKE is written in KEE<sup>1</sup>, and runs on Unisys Explorer<sup>2</sup>.

KEE is a knowledge engineering environment and provides facilities for representing and manipulating knowledge about an application domain. It includes a frame based representation language, a production rule system for reasoning about the knowledge held by the frame based system and a set of highly interactive user interface tools. ASKE uses KEE's frame language as the basic data structure on which the template scheme is built. ASKE's interfaces are written in KEE's Common Windows tool-kit.

### 5.3.2 Interfaces to ASKE

Aske is the main interface of ASKE (Figure 5-3). All interaction, between

<sup>1</sup> Knowledge Engineering Environment (v 3.1) - © Intellicorp (1987)

<sup>2</sup> © Texas Instruments

ASKE and user, until the stage of rule editing (second half of stage two) is carried out in the Aske Interface.

---

---

Figure 5-3

The important elements of the Aske Interface are: 6 icons (top left), Interaction Window (left) and Notebook (top right). The Display Window (DW) (bottom right) is used for displaying the current knowledge-base. At the start of the session, the top-level ASKE knowledge-base, *askedata*, is displayed.

By clicking on the icons appropriate action can be taken. A new session starts when the New KB icon is clicked. Load KB loads a previous unfinished session, Save KB saves the current session into a file, Quit exits the session, Help provides documentation on ASKE; and Rulemaker brings up the Rulemaker interface for editing rules.

---

---

Stage one dialogue takes place in the Interaction Window. This session involves question-answering and the user has little control over the interaction. Task modelling is the most difficult stage of KA and the user has to be talked through. However, in the second stage, interaction via graphics is introduced, providing the user with a greater freedom in encoding his/her knowledge. Further details on the various graphical input/output facilities are given in Section 5.5.

An important component of the Aske Interface is the Notebook. It acts as an intermediate repository of knowledge: it provides the user access to the encoded domain knowledge, as well as a facility for inputting new knowledge directly into the knowledge-base. Notebook is described in Section 5.4.

ASKE is based on the rule-based paradigm and hence an interface for editing rules is provided in Rulemaker (Figure 5-4). The latter part of Stage two is carried out in the Rulemaker Interface. Section 5.6 presents the Rulemaker module of ASKE.

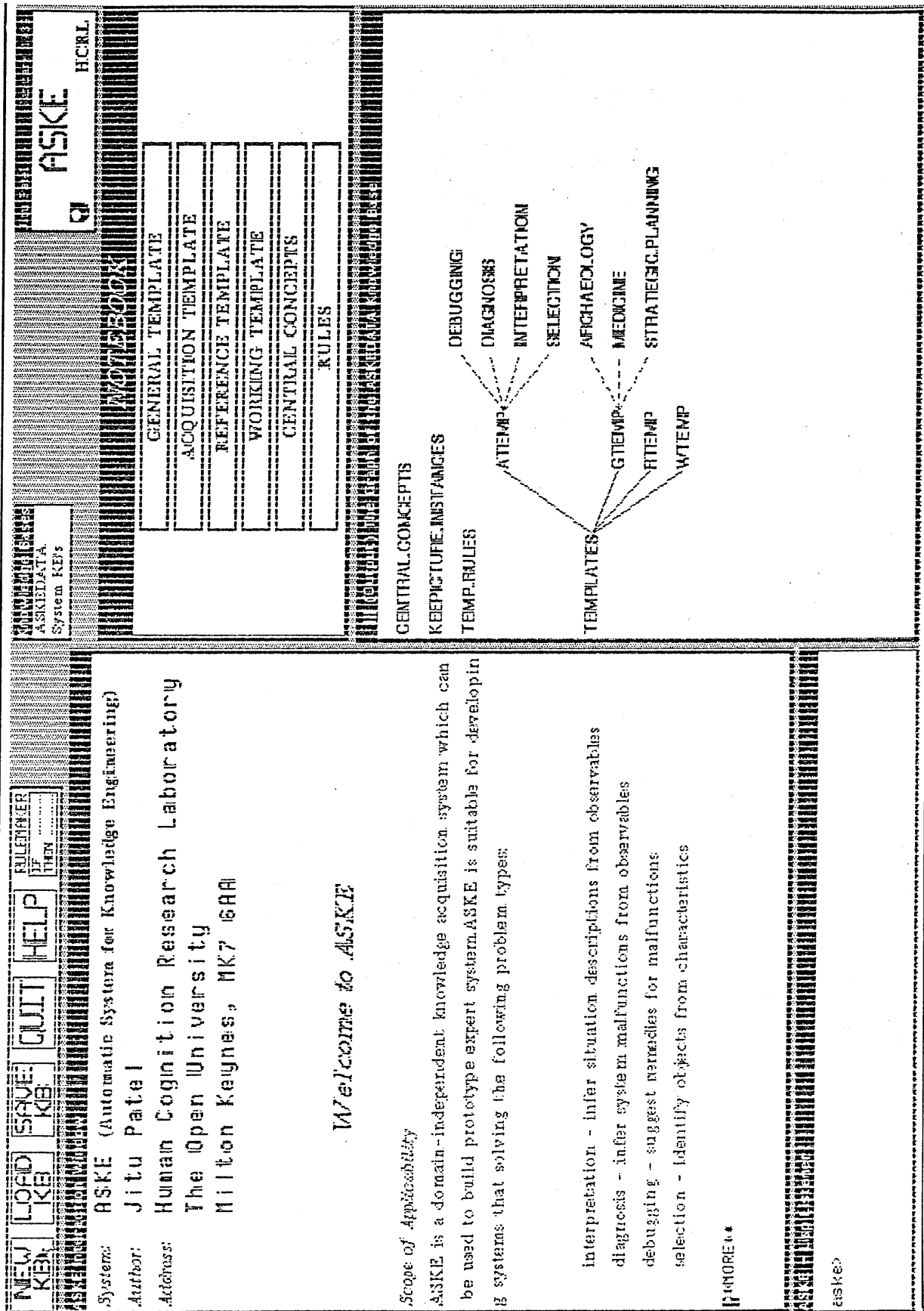


Figure 5-3 The Aske Interface

CONTEXT CLASS RULE QUIT HERE

SETTLEMENT SITE ASKEDATA System KB's

## RULEMAKER

ECRL

TEMP:RULES

- ACTIVITIES.T
  - BUTCHERING.T
  - FOOD.PREPARATION.T
  - MET.AL.PRODUCING.T
  - POTTERY.MAKING.T
  - STORAGE.AREA.T
- ARTIFACTS.T
- FEATURES.T
- SITE.PROFILE.T
  - EXCHANGE.CONTACTS.T
  - OCCUPANCY.T
  - SOCIAL.STATUS.T

TEMP:RULES0

TEMP:RULES 1

TEMP:RULES2

TEMP:RULES3

OCCUPANCY.T

Temp:rule80

IF (butchering)

THEN (occupancy)

(period (permanent seasonal))

((OCCUPANCY) (PERIOD SEASONAL))

CONCLUSION (TYPE END TO EXIT)

Figure 5-4 The Rulemaker Interface

---

---

### Figure 5-4

The important elements of the Rulemaker Interface are: 5 icons (top left), Rule DW (left), Context DW (top right) and Class DW (bottom right). The Rule Editing Window (not in the Figure) opens when the Rule DW is clicked.

Icons. Context, Class and Rule create a new: rule context, rule class and rule, respectively; Aske displays the Aske Interface.

In Context DW, *activities.t* is a rule context, and *butchering.t* a sub-class. The sub-classes are displayed in Class DW. This window provides facilities for rule merging and rule creation. Leaf nodes in Class DW are the rules which, when clicked on, are displayed in Rule DW.

Rule Editing Window is opened when Rule DW is clicked. The Rule Editing Window buffer displays premise or conclusion with left or middle clicks, respectively.

---

---

### 5.3.3 Templates

A template is implemented as a frame. All templates are arranged in a tree structure with the root called *templates* (see DW in Figure 5-3). The templates shown in the Figure are stored in the *askedata* knowledge-base. When a new session starts, a new knowledge-base is created. Then, at different stages of the session, the four template types are copied from *askedata*.

There are two functions which can be used to copy a template from *askedata* to a new knowledge-base: [1] **copy-template** copies the named template (including all the values of the slots) from the *askedata* to the current knowledge-base; and, [2] **create-template** copies just the framework (i.e., slot and valueclass but not the values of the slots) of the template type. See Section 4.4 for details of the four template types.

### 5.3.4 Cerveau

Cerveau is the control unit of ASKE. It coordinates the flow of information between the user and the system and provides a general help facility. The movement of information between the two interfaces of

ASKE is facilitated by Cerveau. Its most important role is, however, in manipulating templates. It has access to the set of templates, and contains the algorithm for creating new ones.

The top-level function for creating a template is **make-template** which takes two arguments: [1] the name of the template, and [2] the type of template. Depending on the type of template, the named template is either created or copied, as the following lisp code shows.

```

; if the template type is
(case type

  ; gtemp, and a gtemp of the given name exists in askedata
  (gtemp (if (member name (get-descendants gtemp))

    ; then copy the gtemp into the new knowledge-base
    (copy-template name)

    ; else, create a new gtemp and call it 'name'
    (create-template name 'type)))

  ; atemp, copy the named atemp from askedata
  (atemp (copy-template name))

  ; rtemp or wtemp, create a new template of the given type
  ((rtemp wtemp) (create-template name 'type)))

```

## 5.4 Notebook

One of the reasons for developing KA systems is to make the encoding of knowledge easier for domain experts. They do not have to understand how knowledge is represented internally. Hence, domain knowledge must also be represented at an intermediate level at which it is meaningful to the user. In ASKE, the Notebook component provide such a facility. It allows the user to look up and to some extent modify the knowledge-base.

The Notebook module contains a simple interpreter which translates the internally represented knowledge-base into a more readable form. The intermediate representation is accessed by a mouse click on one of the six rectangular windows, called *pages*, in the Notebook (see Figure 5-3). There are six pages: one each for the four template types, Central Concepts and Rules. These six pages provide respective pieces of information.

#### 5.4.1 Templates and Rules Pages

The four template pages and the rules page, when clicked on, display their respective contents. For example, the *general template* page would present something like Figure 4-8. The present implementation does not allow the user to modify the displayed knowledge, however.

#### 5.4.2 Central Concepts Page

The *Central Concepts* page allows the user access to the domain concepts. These concepts are displayed in the Central Concepts Window (Figure 5-5), arranged hierarchically in a tree structure, with the *central.concepts* as the root node. Internally, the concepts are represented as frames. Details about implementation are given in Section 5.5.1.

---

---

#### Figure 5-5

When the Central Concepts page of the Notebook is clicked, the Central Concepts Window (CCW) is displayed. CCW is a graphical interface to the domain concepts. The facilities include, adding and deleting of concepts and editing concept attributes. Instructions for input/output are displayed in the Interaction Window.

The concepts are arranged hierarchically in a tree network with *central concepts* as the level 0 node. A new session starts with level 0. At the end of the Stage One of KA, level 1 nodes (or concepts) are identified. These are the main concept categories for the application task. For the interpretation of Settlement Sites application, which is developed in Chapter VI, the level 1 nodes are: *activities*, *artifacts*, *features* and *site.profile*.

---

---



**NEW KEY** **LOAD KEY** **SAVE KEY** **QUIT** **HELP** **RULEMAKER** **THEN**

ASKE DOCUMENTATION WINDOW

SETTLEMENT SITE  
ASKEDATA  
System KEYS

ASKE HCRL

```

CENTRAL CONCEPTS
├── ACTIVITIES
│   ├── BUTCHERING
│   ├── FOOD PREPARATION
│   ├── METAL PRODUCING
│   ├── POTTERY MAKING
│   └── STORAGE AREA
├── ARTIFACTS
│   ├── METAL
│   ├── POTTERY
│   └── STONE
├── FEATURES
│   ├── DITCH
│   ├── FLOOR
│   └── HEARTH
└── SITE PROFILE
    ├── EXCHANGE CONTACTS
    ├── OCCUPANCY
    └── SOCIAL STATUS
            
```

---

**Purpose:** Identify relationships between concepts.

**Comments:** In the Central Concepts Window is given a tree of main concept categories for the task of ARCHAEOLOGY in INTERPRETATION. The data and solution categories are:

**Date:** FEATURES ARTIFACTS

**Solution:** SITE PROFILE ACTIVITIES

The objective is to draw relationship between the concepts in the data categories with those in the solution categories.

----- Instructions: -----

Clicking mouse button on a concept would enable the following operations:

Left 1: Create new object. Left 2: Redisplay object.

Middle 1: Delete/Rename object. Middle 2: Define object relationship.

Right 1: Edit object.

Clicking anywhere else in the window would enable the following operations:

Left 1: Redisplay.

Middle 1: Toggle display.

Right 1: Flush window (quit).

---

[For further help on identifying concept categories in your task look under Input for Duration Concepts in HELP menu]

aske

Figure 5-5 Central Concepts Window

## 5.5 Other Features

There are two main objectives in the first half of stage two: [1] to obtain concept hierarchies, and [2] identify relations between concepts. The user could be prompted for information, however, such a method would be tiresome and boring. Hence, a graphical interface has been introduced. The user is provided with mouse and menu driven facilities: Sketch-Pad and Relations Window. The Sketch-Pad is a facility for inputting concepts and their attributes, the Relations Window is for encoding relationships between concepts.

### 5.5.1 Sketch Pad

One of the main reasons why KA has received so much attention is because the domain expert's problem-solving knowledge is not formally represented, but it is in a compiled form. The expert has to make a special effort to make this knowledge more explicit. And this is not so easy, as Berry (1987:145) points out:

"As far as human experts are concerned they not only have difficulty describing what they do because their knowledge is no longer in declarative form, but because some aspects of their knowledge never have been represented explicitly. They have been learned through experience, rather than being picked up from one or more textbooks."

How to acquire implicit knowledge that has never previously been explicitly represented is an open question. Elicitation of knowledge is however not the only problem. The domain expert needs to provide concept hierarchies as required for expert system development. This for example, how to distinguish and classify concepts and attributes. Consider a hypothetical example where two artifacts, metal and stone, have to be classified in terms of their uses: tool or weapon. There are two ways in which they can be organized: [1] the two artifacts are concepts and the uses are attributes; [2] the two uses are concepts and the artifacts are attributes.

NEW LOAD SAVE QUIT HELP  
 KB KE KB KB

RULE MAKER  
 OF  
 THEM

ASKE  
 HCRI

purpose: identify domain concepts

Comments: Please select a concept, from the CENTRAL CONCEPTS window, for which the selected item is an attribute.

ALL concepts are in the CENTRAL CONCEPTS window. Thus, click mouse or move off window to remove this window. The window will be transferred to the CENTRAL CONCEPTS window.

Instructions

[1] Enter new concepts by typing the concept name followed by <CR>, in the bottom right window. The concept will be displayed in the SKETCH PAD window.

[2] The concept can be moved to the CENTRAL CONCEPTS window by using mouse clicks (defined in the mouse documentation window).

For further help on identifying concept categories in your task look under Input for Domain Concepts in HELP menu

SETTLEMENT-SITE ASKEDATA System KEYS

```

            graph TD
            CC[CENTRAL CONCEPTS] --- ACT[ACTIVITIES]
            CC --- ART[ARTIFACTS]
            CC --- FEAT[FEATURES]
            CC --- SITE[SITE PROFILE]
            ACT --- BUT[BUTCHERING]
            ACT --- FOOD[FOOD PREPARATION]
            ACT --- METAL_PROD[METAL PRODUCING]
            ACT --- POTTERY_MAK[POTTERY MAKING]
            ACT --- STORAGE[STORAGE AREA]
            ART --- METAL[METAL]
            ART --- POTTERY[POTTERY]
            ART --- STONE[STONE]
            FEAT --- CHURCH[CHURCH]
            FEAT --- FLOOR[FLOOR]
            FEAT --- HEARTH[HEARTH]
            SITE --- EXCHANGE[EXCHANGE CONTACTS]
            SITE --- OCCUPANCY[OCCUPANCY]
            SITE --- SOCIAL[SOCIAL STATUS]
            
```

Enter concept: fabric

Figure 5-6 Sketch-Pad

**Figure 5-6**

Sketch-Pad, the centre window, is a facility for inputting data into the knowledge-base at the initial stages of knowledge encoding. Items are entered through the Input Window (bottom left) which are then transferred to the Sketch-Pad. Hence, at this stage, the user does not have to worry about distinguishing between various items - whether they be concepts or attributes. Once again, operating instructions are displayed in the Interaction Window (left).

A left mouse click on an item in the Sketch-Pad indicates it is a concept. This concept is transferred to the CCW; it is member concept of the one indicated by the user. A middle mouse click indicates that the item is an attribute.

Thus, the distinction between a concept and an attribute is not always clear and can be a source of migraine!

The Sketch-Pad, as the name suggests, is a sketching area in which the expert can input concepts without worrying about where they should go in the hierarchy. All concepts are initially entered in the Sketch-Pad (Figure 5-6). These are subsequently transferred to the CCW by means of mouse clicks and menu selections. Note that only the concepts are displayed in the CCW, and attributes of concepts are only displayed when requested.

In the above example, if the user had decided to classify the items in terms of artifacts (i.e., case [2]), they would be represented as follows:

OBJECT	ATTRIBUTE	POSSIBLE VALUES
metal	use	tool, weapon
stone	use	tool, weapon

This is internally represented in a frame structure: every object is a frame and attributes its slots. The set of possible values is a facet of the slot, called *valueclass*. A valueclass specifies what values a slot (or attribute) may have. Because concepts are implemented in the frame language, the

properties of the more general concepts are automatically inherited by the more specific ones. Thus, slots and their valueclass have to be specified only once for a hierarchy of concepts.

When an attribute is moved to the CCW, the user is asked to supply possible values, as shown in Figure 5-7. First, the *value type* has to be identified. A value type defines the format the valueclass of the slot will have. There are three possible types:

- 1 numerical: attribute has a numerical value;
- 2 list of items: attribute takes one of a list of values - e.g., use can have one of (tool weapon);
- 3 anything: the value type is unknown.

The value type 1 suggests that the set of permissible values for the attribute is any number. 2 is the most common value type; it takes a set of values, one or more of which can be the value of the attribute. The third value type suggests that value of the attribute is unknown.

---



---

#### Figure 5-7

Once an item in the Sketch-Pad is identified to be an attribute, it is transferred to the CCW. Next, the user is prompted for the value types for the attribute. For example, *period* item in Sketch-Pad was identified as an attribute of *occupancy*. The user is next asked to specify the value type *period*. There are 3 main options: numerical, list of items or unknown. If the second is selected, the user is asked to supply a list of possible values for the attribute.

---



---

### 5.5.2 Relations Window

The next step in KA, after obtaining the domain concepts, is to identify relationships between them. Two concepts are said to be related if one supports (or inferred-from) the other. For example, if the presence of *butchering* is evidence of *occupancy* of an area then the following relationship exists between the two:

NEW LOAD SAVE QUIT HELP  
 KEYS KEYS KEYS KEYS KEYS

ASKE  
 SETTLEMENT-SITE  
 ASKEDATA  
 System KEYS

HCRL

**Purpose:** Identify domain concepts.

**Comments:**

ALL concepts are first entered in the SKETCH PAD window. These concepts will subsequently be transferred to the CENTRAL CONCEPTS window.

**Instructions** -----

[1] Enter new concepts by typing the concept name followed by <CR>, in the bottom right window. The concept will be displayed in the SKETCH PAD window.

[2] The concept can be moved to the CENTRAL CONCEPTS window by using mouse clicks (defined in the mouse documentation window).

[For further help on identifying concept categories in your task look under Input for Domain Concepts in HELP menu]

CONTENT  
 EDGE  
 MATERIAL

What is the value type for the attribute? (choose one of the following):

1. NUMERICAL-VALUE
2. LIST-OF-VALUES
3. UNKNOWN

=> 2.

Enter possible values:  
 <list-of-words> permanent seasonal

ACTIVITIES: BUTCHERING, FOOD PREPARATION, METAL PRODUCING, POTTERY MAKING, STORAGE AREA

ARTIFACTS: METAL, POTTERY

FEATURES: DITCH, FLOOR, HEARTH

TYPE PROFILE: EXCHANGE CONTACTS, OCCUPANCY, SOCIAL STATUS

Enter concept:

Figure 5-7 Moving Attributes

butchering supports occupancy  
(corollary: occupancy is inferred-from butchering).

The bi-directional nature of a relationship is represented in ASKE with two link types: supports and inferred-from. These links are implemented as slots of frames. Every concept has two slots: supports and inferred.from for recording the respective relationships. They take the name of the concepts to which the given concept is associated.

---

---

#### Figure 5-8

The Relations Window is opened by double-clicking on the concept in the CCW. The associations between this concept to others is displayed in two windows: Supports Relationships and Inferred-from Relationships. Supports Relationships (top right) displays all concepts that the main concept (shown on left) supports. In the example, butchering is the main concept and it support occupancy. Inferred-from Relationships (bottom right) displays concepts from which the main concept is inferred from. Thus, butchering is inferred-from three concepts: floor, metal and pit.

A new relationship for butchering can be added by first clicking the left mouse button when the cursor is on the Relations Window. Then clicking on any concept in the CCW creates a link between the two.

---

---

When a link is made, the names of the related concepts are added to the values of their inferred.from and supports slots. The links are then displayed in the Relations Window. The Relations Window is consists of two halves: the top one displays the supports relationship and the bottom one displays the inferred-from relationship (Figure 5-8). The user, however, does not need to know the link types nor is s/he required to specify (in most cases) the direction of relationship. ASKE knows how concepts are related from its model of heuristic classification. The cases where the model fails to determine direction of a relationship - for example, when two concepts from the same category are explicitly shown to be related, ASKE prompts for it.

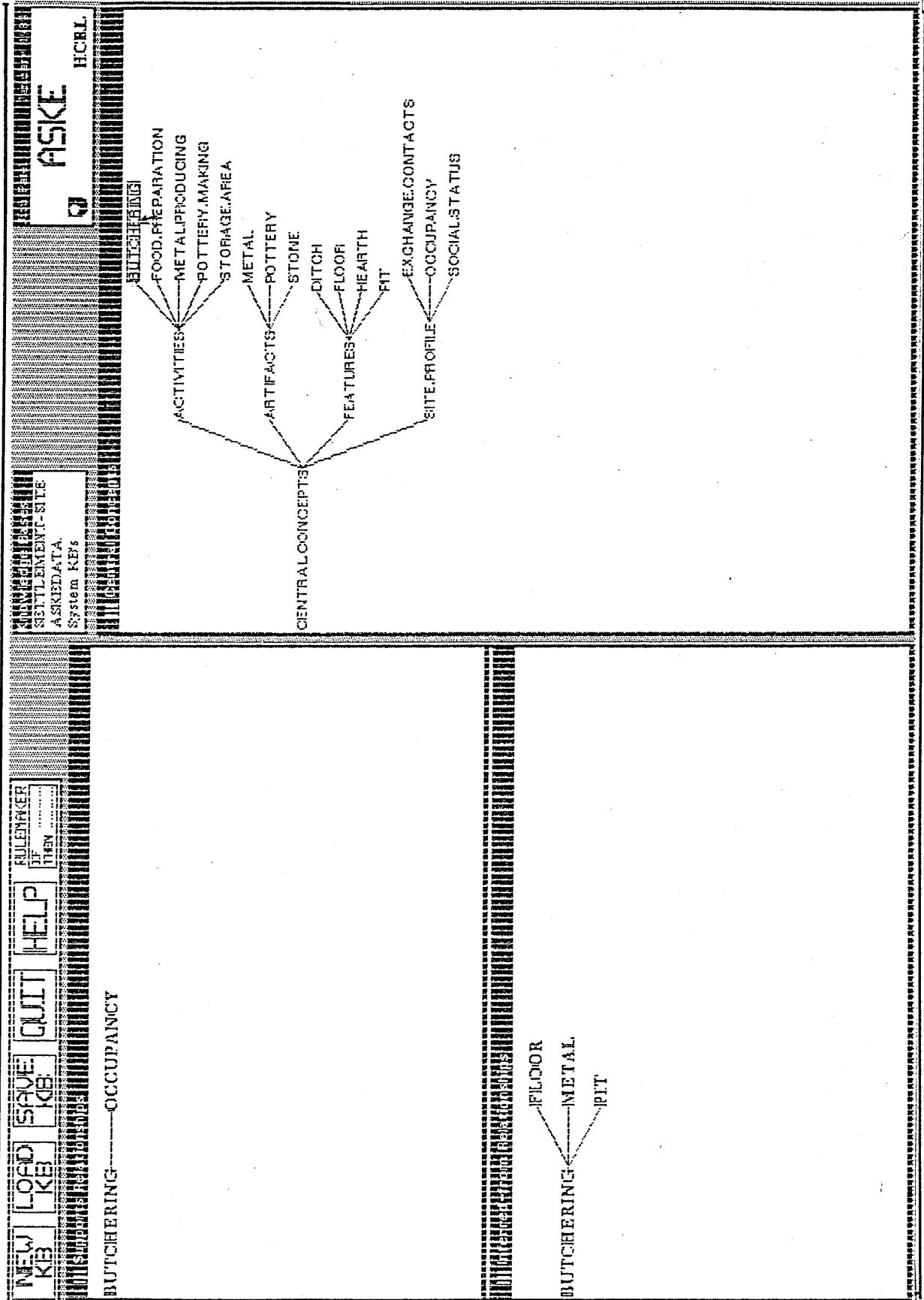


Figure 5-8 Relations Window



This stage of identifying the relationship between concepts is a crucial one: rules for problem-solving are derived from them. ASKE does this automatically. For example, from the above relationship between *butchering* and *occupancy*, the following rule is generated:

```
IF    butchering
THEN  occupancy
```

Hence, the "supported" concept is made into the conclusion of the rule, and the "inferred-from" one into the premise. The attributes of the concepts, if any, would be conjoined to the concept clause.

## 5.6 Rulemaker

Once rules are generated, ASKE switches to the Rulemaker. The Rulemaker interface was designed to make the task of rule editing easier for the user. KEE's rule system provides an extensive facility but is quite complex and requires weeks of training before one can do anything with it. Furthermore, for an initial prototype system, which ASKE helps develop, it is not necessary to have an in depth knowledge of the rule system. In the Rulemaker, an attempt has been made to make rule editing simpler.

### 5.6.1 Contexts and Classes

Rules are arranged hierarchically in contexts and classes. A context contains sets of rules, arranged in a hierarchy of classes and sub-classes, which provide solutions to sub-problems. ASKE creates contexts out of the main concept categories, e.g., *artifact.t* is obtained from *artifact* category type. A set of rules with the same conclusion are arranged in classes, and class is named after the concept - e.g., *butchering.t* (Figure 5-9).

The need for contextual representation of rules comes from our experience with building KIVA (Patel and Stutt, 1989a), an expert system for interpreting archaeological data. The KIVA rulebase is divided into

five contexts corresponding to the five stages involved in solving the interpretation problem (i.e., transforming archaeological finds into description of past civilizations) in archaeology. Thus, contexts also relate to sub-tasks (or solutions to sub-problems) of the task addressed by the expert system.

### 5.6.2 ASKE Rules

For every pair of related concepts ASKE creates a rule, which carries the prefix "temp.rule". These rules are simple in structure and need modifications. The Rulemaker has facilities for rule editing (described in the following section) and merging. Rules merging means two or more rules are squashed into a single rule. All the premises are conjuncted to form the premise of the new rule. The same thing is done to the conclusions of the rules. The operation is carried out in the Class DW. For example, rules *temp.rule31*, *temp.rule32* and *temp.rule33* were merged and edited to produce *temp.rule41* as follows.

<u>Temp.rule31</u> IF metal.producing THEN occupancy period (seasonal permanent)	<u>Temp.rule41</u> IF metal.producing pottery.making ditch use boundary size perimeter THEN occupancy period permanent
<hr/> <u>Temp.rule32</u> IF pottery.making THEN occupancy period (seasonal permanent)	
<hr/> <u>Temp.rule33</u> IF ditch use (boundary sewage) shape (circular linear) size (site.perimeter ) THEN occupancy period (seasonal permanent)	

### 5.6.3 Rules Editing Window

The premise and conclusion of a rule can be edited in the Rule Editing Window (left bottom in Figure 5-9). The editor uses Zmacs and it is opened by clicking left (or middle) mouse-button on the Rule DW. The editor will contain the premise (or conclusion) of the rule displayed in the Rule DW.

---

---

#### Figure 5-9

The premise and conclusion parts of a rule can be edited in the Rule Editing Window (REW), which is displayed with a mouse click when the cursor is on the Rule DW. Depending on whether a left or middle mouse is clicked, the premise or conclusion, resp., is placed in REW buffer.

The example shows the conclusion items in the buffer. The buffer is a Zmacs editor. Basically, what is required of the user is to edit the value of the *period* attribute of the concept *occupancy*. However, new items can be added at this point. When editing is completed, the REW is removed by typing 'end' key. The modified rule is displayed in the Rule DW.

---

---

CONTEXT CLASS RULE QUIT HERE

Temp rule 33

IF

(ditch)

(use (drainage boundary))

(size (site.perimeter area.perimeter))

(shape (circular elongated))

THEN

(occupancy)

(period (permanent seasonal))

CONCLUSION (TYPE END TO EXIT)

TEMP RULES

- ACTIVITIES.T
- ARTIFACTS.T
- FEATURES.T
- SITE.PROFILE.T

BUTCHERING.T

- FOOD.PREPARATION.T
- METAL.PRODUCING.T
- POTTERY.MAKING.T
- STORAGE.AREA.T
- EXCHANGE.CONTACTS.T
- OCCUPANCY.T
- SOCIAL.STATUS.T

OCCUPANCY.T

- TEMP.FULES0
- TEMP.FULES1
- TEMP.FULES2
- TEMP.FULES3

**RULEMAKER**

EICRL

SETTLEMENT-SITE

ASKEDATA

SYSTEM KB'S

Figure 5-9 Editor Window in Rulemaker

# Chapter VI

## PROCESSING IN ASKE

*They say it works, whether you believe in it or not.*

### 6.1 Introduction

In this chapter, I will illustrate how ASKE processes information by going through the main steps in the development of a prototype system in the domain of archaeology. The scenario concerns the development of a knowledge base for interpreting settlement sites.

#### 6.1.1 Archaeology

Archaeology is concerned with explaining prehistoric social organizations. Typically, an archaeologist excavates a site, the finds<sup>1</sup> and features<sup>2</sup> are recorded, classified and interpreted. Of these, only the latter two are conducive to expert system technology (Patel and Stutt, 1989b). The classification task is normally restricted to classifying artifacts. In the interpretation task, the social habits and the organization of past societies are inferred from the distribution of finds and features. For example, Hill (1970:19) writes, on interpreting prehistoric sites in southwest USA,

"Where different kinds of activities are carried out within a community, one would expect to find different kinds of artifacts; and the presence of different artifacts in particular rooms or areas within an archaeological site should be usable as evidence

---

<sup>1</sup> Finds are archaeological objects, which may be either man-made or natural. The former type of finds are referred to as artifacts, and the latter as ecofacts.

<sup>2</sup> Features refers to the various aspects of archaeological sites both man-made and natural (e.g., pits, ditches, walls).

in inferring the activities of these rooms and areas - assuming that one can identify the uses of the artifacts involved. In areas where food processing and cooking was done, for example, one might expect to find such items as mealing bins, metates, manos, firepits, fire-blackened pottery, charred bone remains, and a number of other things as well".

Figure 6-1 depicts the classification of the domain of archaeology. Two of the main areas for which archaeologists do interpretation are settlement and burial sites. The example we will be looking at is the interpretation of settlement sites (shaded area).

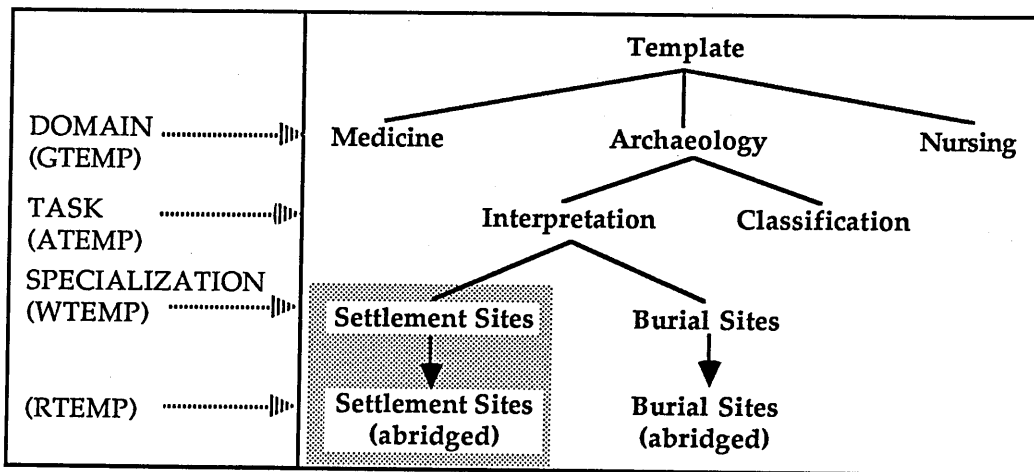


Figure 6-1 Templates for Archaeology, shown in the broader context of several domains to which ASKE has been applied

ASKE has ATEMPs for interpretation and classification (or selection), the two of the most important tasks carried out by Archaeologist. It also has a GTEMP for Archaeology, as it was used for developing a prototype system for interpreting Burial Sites. From this knowledge base, an RTEMP was abstracted (hence it is an abridged Burial Sites). The shaded area is the subject matter of this Chapter.

### 6.1.2 Overview

An ASKE session is divided into four parts (Figure 6-2). In the first part, the objective is to identify task characteristics. In the second part, the task model is built based on the ATEMP and guided by the RTEMP. In the third part, the task model is used as the basis for knowledge elicitation.

The last part involves the user editing the automatically generated rules. At the end of the session, ASKE outputs intermediate-level rules and creates a new RTEMP from the knowledge base.

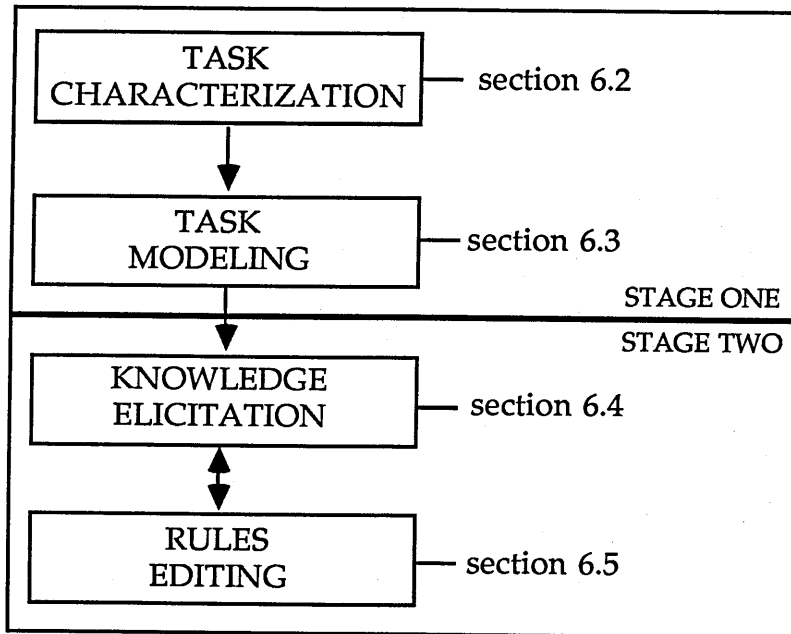


Figure 6-2 Inferencing in ASKE

The actual session is described with the help of screen snapshots of some of the more interesting dialogue between ASKE and the user. In the main text, a general description of the processing and some implementation details are presented.

## 6.2 Task Characterization

The aim of the task characterization stage of KA is to identify the task that the new system will be performing. The session includes an introduction to ASKE, elicitation of domain classification and project goals, and selection of templates for task modelling.

### 6.2.1 Starting a new session

Clicking on the 'New KB' icon starts the session. First, the user is

presented with a short tutorial on ASKE. The session proper begins with a set of questions to elicit user's personal details (Figure 6-3). This information is used as a reference for the knowledge base.

---

---

### Figure 6-3

Stage one is completely system driven. The user is asked questions, which may be either multiple choice or those that require a typed answer. In the latter case, ASKE uses two types of prompts: "<anything>" or "<word>", which take as input a sentence or a word, respectively. Repeated prompts are used for multiple replies. All stage one interactions take place in the Interaction Window.

Personal Details. First, the personal details of the domain expert are obtained. These include the name, the address and the telephone number.

---

---

## 6.2.2 Obtaining the domain classification

Next on the agenda is the elicitation of the domain classification. This includes identifying: [1] the domain of the new application; [2] the task that the system will be performing; and, [3] the area of application.

---

---

### Figure 6-4

Next, the domain classification is elicited. The classification is made up of the name of the domain, the task type of the problem and the specialist area of the new application.

[1] Domain. The Output Window (bottom right) shows the various GTEMPS of the domains in which knowledge bases have been developed. These domains are presented as choices to select from. The user is, however, allowed to build a system in a new domain, (e.g., by choosing 5).

---

---

### Domain

ASKE queries the user for the name of the domain (Figure 6-4). A choice of domains, about which ASKE knows, are presented to select from. From the selection, a GTEMP is identified as follows:



NEW LOAD SAVE QUIT HELP  
KB KB KB

ASKE interface for Windows

ASKE/ASKEDATA  
System KE's

ASKE HCR.L

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

Purpose: Obtain personal details:

Comment:

This is used for reference purposes only.  
[Type NIL, if answer is unknown.]

Enter your first name  
<anything> Jitu

Enter your address  
<anything> HCR.L, The Open University, Milton Keynes.

Enter your telephone number (day time)  
<anything> 0908 652572

\*\*\*MORE\*\*

ask>

GENERAL CONCEPTS

KEEPIC:TYPE:INSTANCES

TEMP:RULES

TEMP:PLATES

DEBUGGING

DIAGNOSIS

INTERPRETATION

SELECTION

ARCHAEOLOGY

MEDICINE

STRATEGIC PLANNING

ATEMP

GTEMP

RTEMP

WTEMP

Figure 6-3 Obtaining personal details

NEW LOAD SAVE QUIT HELP  
 KB KB KB KB

ASKE2  
 ASKEDATA  
 System KEYS

ASKE  
 HCRL

---

Purpose: Identify domain

Comment:

The domain is the general area of the expertise, for example, archaeology, medicine.

e.

[For further help on answering questions, type <CR>.]

---

Enter the name of the domain (choose one of the following):

- 1 STRATEGIC PLANNING
- 2 MEDICINE
- 3 ARCHAEOLOGY
- 4 OTHER

=> 3

IF #MORE\*\*

ASKE>

Figure 6-4 Identifying the domain

IF	domain exists
THEN	use GTEMP of domain
ELSE	create a new GTEMP.

As Archaeology is the domain of the new application, its GTEMP will be used for interrogating the domain expert. If the GTEMP did not exist, (i.e., when the domain is new), a new one is created. This is done by copying an empty GTEMP (i.e., without values for slots) and giving it the name of the new domain.

### Task Type

The next problem is to identify the problem solving task the new system will be performing. As ASKE only knows about the four analysis tasks: debugging, diagnosis, interpretation and selection, these are presented as alternative choices to select from (Figure 6-5).

---

---

### Figure 6-5

[2] Task Type. ASKE provides facilities for developing systems for the following four task types: selection, debugging, diagnosis, and interpretation. There is an ATEMP for each of these.

---

---

### Specialist Area

It is now known that the new system is in the domain of Archaeology and it is for interpretation. To complete the domain classification, the name of the area of application (or specialist area) is needed.

---

---

### Figure 6-6

[3] Specialist Area. The user is presented with the names of previously built knowledge-bases, if any, and prompted for the new application area.

---

---

NEW KB

LOAD KB

SAVE KB

QUIT

HELP

RULEMAKER

IF THEN

ASKE

HCRL

ASKE2

ASKEDATA

System KEYS

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

ARCHAEOLOGY

CENTRAL CONCEPTS

INTERPRETATION

TEMP. RULES

---

**Purpose:** Identify task type

**Comment:**

The task type (or problem type) is the kind of problem the expert usually solves. This can be, for example, diagnostic, repair, design, planning. ASKE is, however, suitable for solving analysis tasks only. The analysis tasks are those which can be solved by the classification problem solving method. Thus, for this category of tasks, the solutions can be pre-enumerated.

Enter the task type (choose one of the following):

- 1 INTERPRETATION
- 2 DIAGNOSIS
- 3 DEBUGGING
- 4 SELECTION

→ 1

IF MORE

ASKE>

Figure 6-5 Identifying task type

NEW LOAD SAVE QUIT HELP  
 KB KB KB

ASKED FOR SETTLEMENT SITE  
 RULETAKER  
 OF  
 THEN

Purpose: Identify specialist area.

Comment:

The specialist area is the actual area of expertise. In the domain of medicine, for example, it will be: foot problems, aids infections, lung diseases.

Enter your specialist area (choose one of the following):

1 BURIAL.SITE  
 2 OTHER  
 => 2

Enter your specialist area  
 <word> settlement.site

IGNORE\*\*

asked

ASKE H.C.R.L.L.

GENERAL TEMPLATE  
 ACQUISITION TEMPLATE  
 REFERENCE TEMPLATE  
 WORKING TEMPLATE  
 CENTRAL CONCEPTS  
 RULES

ARCHAEOLOGY  
 CENTRAL.CONCEPTS  
 INTERPRETATION  
 SETTLEMENT.SITE  
 TEMP.RULES

Figure 6-6 Identifying area of specialization

### 6.2.3 Defining project goals

For knowledge base creation, it is not enough to acquire just the problem-solving expertise. It is vitally important to know why the system is being developed and who will eventually use the system. The former provides the motivation and justification for carrying out the project. The latter plays a role in designing user interface for the systems. Thus, the next set of questions are related to project goal definition (Figure 6-7). This information is stored in the knowledge-base and will be available to the systems developer for the implementation of the user interface.

---

---

#### Figure 6-7

Next, the user is prompted for information on project goals. Three pieces of information are elicited within this category.

- 1 Task description. A statement about what the system will do.
  - 2 The project aim. Why is the system being built? What will be its role?
  - 3 Users. Who will use the system?
- 
- 

### 6.2.4 Selecting templates for the acquisition of task model

The next stage of KA, task modelling, is guided by ATEMP and RTEMP. The ATEMP will provide expectations of what these important concept categories are for the new application. The RTEMP will assist by presenting an example of a task model in a similar application. These two templates are selected on the basis of information in the domain classification.

#### Selecting an ATEMP

The ATEMP is selected on the basis of the task type - for every task type there is a corresponding ATEMP. ASKE has four ATEMPs for the four task types: debugging, diagnosis, interpretation and selection. Hence, in selecting the task type (in the domain classification stage) the user (indirectly) specifies the ATEMP.

<p>NEW LOAD SAVE QUIT HELP          KB KE KB</p> <p>ASKE</p> <p>Purpose: Identify project aims</p> <p>Comment:          Information about project and users will be used for reference purposes.          [Type NIL, if answer is unknown.          For further help on answering questions, type &lt;GF&gt;.]</p> <p>Give a description of the task</p> <p>&lt;anything&gt;</p> <p>e.g., diagnosing foot problems, interpreting archaeological sites</p> <p>&lt;anything&gt; inferring activity areas in a settlement site from archaeological data</p> <p>What are the project aims?</p> <p>&lt;anything&gt; provide expertise on interpreting settlement sites to archaeologists</p> <p>"&gt;</p> <p>Who will be using the system?</p> <p>&lt;anything&gt; archaeologists</p> <p>asker&gt;</p>	<p>ASKE</p> <p>SETTLEMENT.SITE          ASKEDATA          System KEYS</p> <p>HCURL</p> <p>GENERAL TEMPLATE</p> <p>ACQUISITION TEMPLATE</p> <p>REFERENCE TEMPLATE</p> <p>WORKING TEMPLATE</p> <p>CENTRAL CONCEPTS</p> <p>RULES</p> <p>ARCHAEOLOGY</p> <p>CENTRAL CONCEPTS</p> <p>INTERPRETATION</p> <p>SETTLEMENT.SITE</p> <p>TEMP.RULES</p>
---	---

Figure 6-7 Defining the project goals

### Selecting an RTEMP

The selection of an RTEMP, however, takes a slightly more complex procedure. ASKE matches the domain classification (i.e., domain, task-type and specialist area) of the new application to the list of RTEMPs in the knowledge base. The RTEMP which is closest to the required classification is selected. Basically, there are three possible matches:

1	IF	domain, task-type and specialist area match
	THEN	pick specialist area template
2	IF	domain, task-type match
	THEN	pick different specialist area template
3	ELSE	ask user to pick a specialist area from the set of all RTEMPs

Case 1: if a knowledge base, for the same application area, was developed previously, use its RTEMP.

Case 2: a new application is being developed in a domain, in which a knowledge base in a different area, but for the same task has been built. The user is asked to select an RTEMP, which is closest to given application, from those that satisfy the condition, (if more than one exists). Otherwise, s/he is informed about the selected RTEMP (Figure 6-8).

Case 3: if the above two fail, the user is presented with a list of all RTEMP to choose from.

---

---

#### Figure 6-8

The selection of RTEMP is based on finding the closest one, in task and domain, to the new application. As there is only one RTEMP that satisfies the conditions, it is selected. The user is informed about this.

---

---



NEW LOAD SAVE QUIT HELP  
 KB KB KB KB

RULER/CR  
 OF  
 ITEM

ASKE  
 HCRL

---

**Purpose:** Select an RTEMP

**Comments:** The RTEMP will be used to guide the development of the initial task model.

We will be using the BURIAL.SITE, which is a knowledge-base in the domain of ARCHAEOLOGY, as an exemplar. BURIAL.SITE is designed to do the task of INTERPRETATION; its task description is: infer habits of past cultures from their burial remains.

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

---

ARCHAEOLGY  
 CENTRAL.CONCEPTS  
 INTERPRETATION  
 SETTLEMENT.SITE  
 TEMP.RULES

---

ASKE ACTION WINDOW

MORE

aske>

Figure 6-8 Selecting an RTEMP

The following lisp code shows how the above procedure for selecting the RTEMP is implemented in ASKE.

The GTEMP, ATEMP and WTEMP of the new application correspond to the domain, task-type and specialist area. Each of them is stored as a list, the first argument of which is a boolean and the second the name. The templates are accessed by typing the name with the prefix 'current'. For example, \*current-atemp\* has the value: (t interpretation). The boolean 't' means that a knowledge base for interpretation exists. For the current session, gtemp and wtemp have the following values, respectively: (t archaeology) and (nil settlement.site).

```

; this function selects an RTEMP
(defun find-rtemp ()

  ; print the introductory message
  (print-rtemp-information)

  ; selection procedure
  ; if knowledge base exists, i.e., specialist area match
  1 (if (car *current-wtemp*)
        ; then find it, and output result
        (get-rtemp (cadr *current-atemp*))

        ; else, check if task-type match
        (let* ((task-type (cadr *current-atemp*))
               ; if it matches
               2 (s-areas (if (car *current-atemp*)
                              ; list all RTEMPs for the task-type
                              (get-s-areas-of-tasks task-type)

                              ; else, list all RTEMPs in ASKE
                              3 (get-all-s-areas))))

          ; if there is only one RTEMP
          (if ((= (length s-areas) 1)
              ; then, output the result
              (car s-areas)
              ; else, let user select the RTEMP
              (ask-user "Pick an RTEMP" s-areas))))))

```

### 6.3 Obtaining the task model

The building of the task model is the most difficult stage of the entire KA process. At this stage, the expert is expected to lay down the main concept

categories for the domain. Once the categories are identified, half the work towards the development of the knowledge base is done. The problem stems essentially from the fact that the expert is forced to reorganize his/her domain knowledge so that it can be directly encoded into a computer program.

The task model for the problem is obtained in the following three steps.

[1] The user is presented with a description of the task, that is, what the important concept categories are and how they are related. This information is held in the ATEMP. For the present example, the main concept categories for interpretation are described (Figure 6-9).

---

---

Figure 6-9

The central aim of the task modelling stage is to obtain the main concept categories for the application. ASKE uses the ATEMP and the RTEMP, selected earlier, to elicit the task model. First, the user is presented with a brief (generic) description of the task of interpretation.

---

---

[2] A concrete example is presented to help the user relate to the relatively abstract description of the task presented before. Figure 6-10 shows what the main concept categories are in the knowledge base for Burial Sites (which is obtained from its RTEMP). This will be useful in focussing the attention at the right level when providing the model of the new application.

---

---

Figure 6-10

Next, a concrete example of a task model, from a related application area - Burial Sites, is presented. This example is obtained from the RTEMP.

---

---

[3] The user is then prompted for the main data (Figure 6-11) and solution (Figure 6-12) categories for the task. The information required is the name and a short description (or comment) of the concept.

NEW LOAD SAVE QUIT HELP  
 KB KE KB KB KB

RULES KEYS  
 IF THEN

ASKE  
 HCRL

**Purpose:** Identify the main concept categories

**Comments:** For problem solving in analysis tasks, there are basically two categories of concepts: data and solution. At this stage of KA, we are interested in identifying this concept categories.

There may be one or more of each type of concept categories.

\* For DATA categories, type in the name of concept/s which represent the basic data for the problem. The characteristic of data category is that it is either given/observed or obtained by test.

\* For SOLUTION categories, type in the name of concept/s which represent the basic solution for the problem. The characteristic of solution category is that it is inferred from data categories.

---

The task of interpretation involves inferring situation description from sensor data.

**Data Types:**

Data

**Solution Types:**

Descriptions

\*\*MORE\*\*

ARCHAEOLOGY

GENERAL CONCEPTS

INTERPRETATION

SETTLEMENT.SITE

TEMP.RULES

ASKE  
 SETTLEMENT.SITE  
 ASKEDATA  
 System KEYS

ARCHAEOLOGY

GENERAL CONCEPTS

INTERPRETATION

SETTLEMENT.SITE

TEMP.RULES

Figure 6-9 What is required in a task model

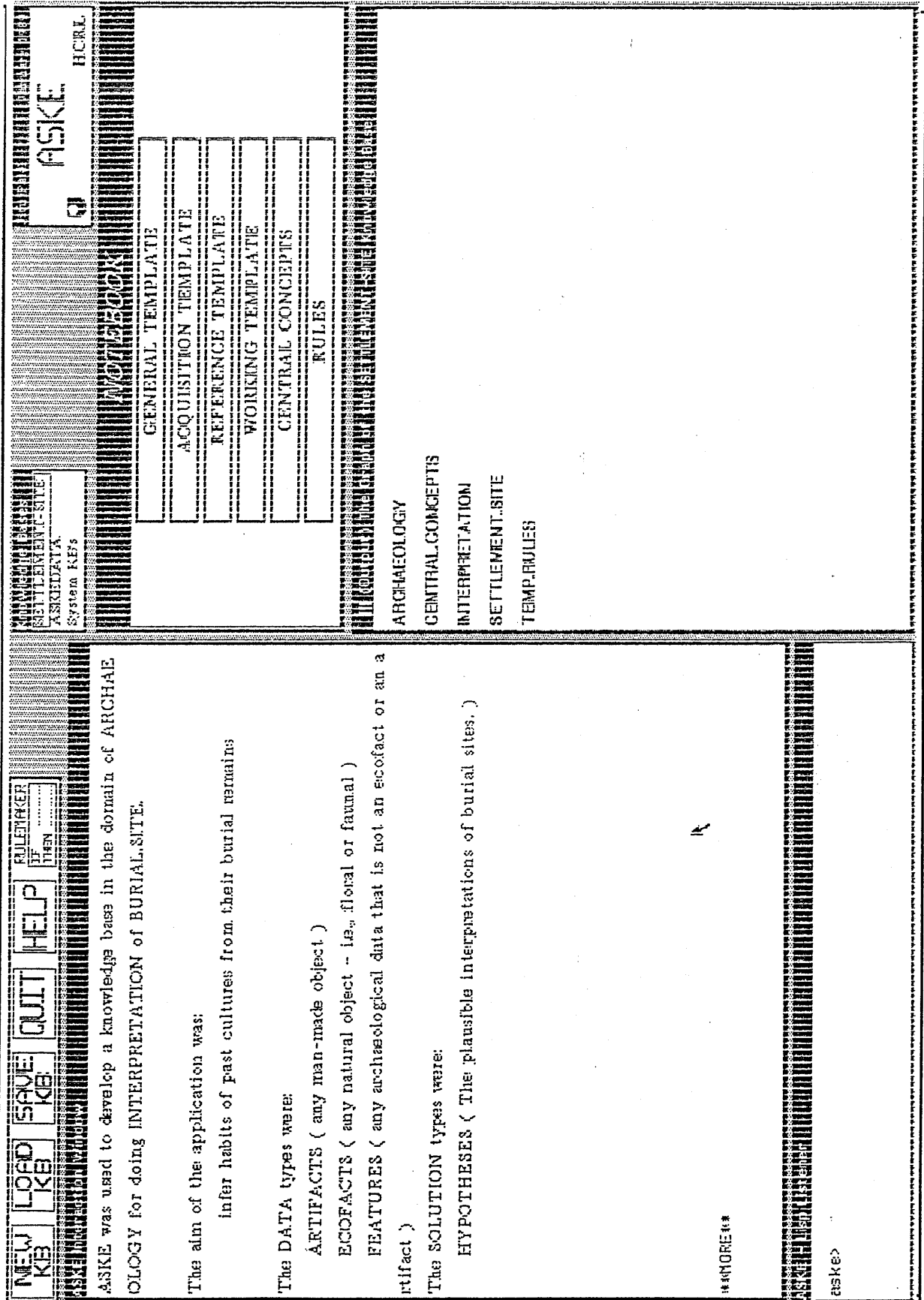


Figure 6-10 An example of a task model

NEW LOAD SAVE QUIT HELP  
 KB KB KB KB

RULEMAKER  
 BY  
 THEN

ASKE  
 HCRL

---

What is the main category of data for the INTERPRETATION of SETTLEMENT.SITE?

<word> artifacts

Comments for ARTIFACTS

<anything> main archaeological data for inferring activities of an area of a room

Enter other data category for INTERPRETATION of SETTLEMENT.SIT

E [Type nil, if none].

<word> features

Comments for FEATURES

<anything> geographical and geological features of the land, including how it has been modified by man

Enter other data category for INTERPRETATION of SETTLEMENT.SIT

E [Type nil, if none].

<word> nil

SETTLEMENT-SITE ASKEDATA

System KEYS

---

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

---

EXPERT: Jitu

DOMAIN: Archaeology

TASK: Interpretation

SPECIALIST AREA: Burial.site

---

TASK DESCRIPTION:

Infer habits of past cultures from their burial remains

---

Data Types:

ARTIFACTS (any man-made object)

ECOFACTS (any natural object - i.e., floral or faunal)

FEATURES (any archaeological data that is not an ecofact or an artifact)

---

Solution Types:

HYPOTHESES (The plausible interpretations of burial sites.)

---

aske>

Figure 6-11 Obtaining the data categories

NEW LOAD SAVE QUIT HELP  
 KB KB KB KB

RULEMAKER  
 IF .....  
 THEN .....

ASKE  
 HCRL

What is the main category of solution when doing INTERPRETATION of SETTLEMENT.SITE?  
 <word> activities

Comments for ACTIVITIES  
 <anything> the kind of activities that took place in a given settlement s  
 lue

Enter other solution category for SETTLEMENT.SITE [Type nil, if non e].  
 <word> site.profile

Comments for SITE.PROFILE  
 <anything> general characteristics of the settlement site as inferred from  
 the distribution of artifacts and features

Enter other solution category for SETTLEMENT.SITE [Type nil, if non e].  
 <word> nil

ASKEDATA  
 System KE's

GENERAL TEMPLATE  
 ACQUISITION TEMPLATE  
 REFERENCE TEMPLATE  
 WORKING TEMPLATE  
 CENTRAL CONCEPTS  
 RULES

EXPERT: Jitu

DOMAIN: Archaeology

TASK: Interpretation

SPECIALIST AREA: Burial.site

TASK DESCRIPTION:  
 Infer habits of past cultures from their burial remains

Data Types:  
 ARTIFACTS (any man-made object)  
 ECOFACTS (any natural object - i.e., floral or faunal)  
 FEATURES (any archaeological data that is not an ecofact or an artifact)

Solution Types:  
 HYPOTHESES (The plausible interpretations of burial sites.)


Figure 6-12 Obtaining the solution categories

NEW LOAD SAVE  
 KB KB KB

QUIT HELP

RULEMARKER  
 IF THEN .....

SETTLEMENT-SITE  
 ASKEDATA  
 SYSTEM KEYS



HCRL

**Purpose:** Identify main concept categories

**Comments:** In the Concept Categories Window is given a tree of main concept categories for the task of ARCHAEOLOGY in INTERPRETATION. The objective objective is to check and, if necessary, amend the categories. The data and solution categories given are:

Data: FEATURES ARTIFACTS

Solution: SITE.PROFILE ACTIVITIES

----- Instructions -----

Clicking mouse button on a concept would enable the following operations:

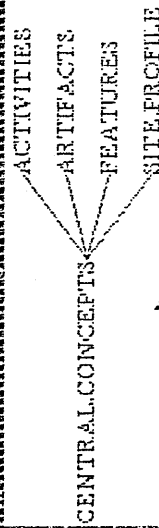
- Left Mouse 1: Create new object.
- Middle Mouse 1: Delete/Rename object.
- Right Mouse 1: Flush window.

Clicking anywhere else in the window would enable the following operations:

- Left Mouse 1: Create new object.
- Right Mouse 1: Flush window (quit).

-----

[For further help on identifying concept categories in your task look under report for Concept Categories in [HELP ITEMS]]



CENTRAL.CONCEPTS

- ACTIVITIES
- ARTIFACTS
- FEATURES
- SITE.PROFILE

aske>

Figure 6-13 Main Concept Categories



---

---

**Figure 6-11**

Then, the user is prompted for the main data categories for the interpretation of Settlement Sites.

The data categories are: artifacts and features

The RTEMP for Burial Sites can be viewed by clicking on the *reference template* page of the Notebook.

---

---

---

---

**Figure 6-12**

Finally, the solution categories are elicited.

The solution categories are: activities and site.profile.

(Note: If more than one descriptive words are used, they have to be joined together with a dot, e.g. site.profile. This is because KEE recognizes spaces as separators.)

---

---

---

---

**Figure 6-13**

At the end of the task modelling stage, the user is presented with the specified task model, which is displayed in the Main Concepts Categories Window (left). This window allows further amendments to the the task model.

---

---

Note that ASKE knows that the basic relation between categories is: data support solution (or solution is inferred-from data). It, however, does not know if there are any causal relationships within concept categories. If there are, and it will be known later, the user will be prompted for the direction of relationship.

## 6.4 Knowledge Elicitation

The knowledge elicitation stage has two main objectives. First, to acquire concepts in the various data and solution categories. Second, to identify the heuristic associations between concepts.

### 6.4.1 Obtaining Domain Concepts

The elicitation of domain concepts is guided by the task model, which

helps the expert focus on the relevant knowledge. The main concept categories are displayed in the Central Concepts Window (CCW). Initially, items (concepts and attributes) are entered into the Sketch-Pad via the Input Window (see Figure 6-14). The reason for including the Sketch-Pad facility is the implicit assumption that domain experts do not have formalized knowledge bases in their heads. In fact, a lot of experts build expert systems to formalize their experiential knowledge (e.g., Baker, 1986), and it is as good a reason as any for using the technology.

When the items are entered into the Sketch-Pad, they are held in a temporary buffer. They are only added to the knowledge base when they are moved to the CCW. Moving of concepts and attributes is done via mouse clicks and menus. For example, when the concept *occupancy* is moved from Sketch-Pad to CCW, the user specifies where the concept will go in the hierarchy. *Occupancy* is a *site.profile*. Now, two things happen.

- [1] A frame for *occupancy* is created in the knowledge-base. This frame (which takes the name of the concept) is linked to the parent of the concept (i.e., *site.profile*). Thus *occupancy* inherits all the attributes of *site.profile*. *Occupancy* also inherits the following four slots from the top-level concept *central.concepts*.

SLOTS	POSSIBLE VALUES	COMMENT
type	data, solution	the type of concept
inferred. from	other concepts	from which concepts is this one inferred
supports	other concepts	which concepts does this one support
t.rules	temp.rules	rules which contain this concept

- [2] The new concept name is added to the WTEMP as a value of the attribute, the main concept category which is the ancestor of the

concept. Hence, *occupancy* is added to the list of values for *site.profile* attribute of *Settlement.Sites*

The following lisp code shows how the adding of a new concept to the knowledge base is implemented.

```

; This function adds the concept to the central concepts window
; it takes two arguments: 1) the object to be transferred; and, 2)
the
; the position in the concept hierarchy where it goes.
(defun add-concept-to-main-window (object parent)

  ; find the main concept category,
  ; which is the ancestor of the object
  (let ((ancestor (find-main-concept-category object)))

    ; create a new frame for the object
    ; and link it to the parent node
    (create-a-concept object *current-kb* parent)

    ; add the concept name to WTEMP
    (add-concept object ancestor (cadr *current-wtemp*)))

  ; delete the object from the Sketch-Pad
  (delete-sp-object object)

  ; redisplay the windows
  (display-sketch-pad)
  (display-main-concepts)))

```

---


#### Figure 6-14

In Stage two, a graphical interface is employed. Information on mouse functionality is also displayed in the Mouse Documentation Window.

#### Obtaining Domain Concepts

New concepts and attributes are entered through the Input Window (bottom right) to the Sketch-Pad (middle). The CCW (top right) displays the concept hierarchy. The root node of the concepts tree is *central.concepts*. The first level nodes are the main concept categories. Left mouse click on an item in the Sketch-Pad specifies it as a concept, a middle one identifies it as an attribute. The concept or attribute is then moved from the Sketch-Pad to CCW.

---



ASKE

HCURL

SETTLEMENT-SITE  
ASKEDATA  
System KEYS

ACTIVITIES  
ARTIFACTS  
FEATURES  
SITE/PROP FILE

CENTRAL.CONCEPTS

SKETCH PAD

ENTER NEW CONCEPTS

REMOVE CONCEPTS

ENTER CONCEPT PERIOD

**NEW** **LOAD** **SAVE** **QUIT** **HELP**  
**KB** **KB** **KB** **KB** **KB**

**ENTER NEW CONCEPTS**

**Purpose:** Identify domain concepts.

**Comments:**

ALL concepts are first entered in the SKETCH PAD window. These concepts will subsequently be transferred to the CENTRAL CONCEPTS window.

**Instructions:**

[1] Enter new concepts by typing the concept name followed by <CR>, in the bottom right window. The concept will be displayed in the SKETCH PAD window.

[2] The concept can be moved to the CENTRAL CONCEPTS window by using mouse clicks (defined in the mouse documentation window).

[3] For further help on identifying concept categories in your task look under Input for Domain Concepts in HELP menu]

ASKED

DITCH  
FLOOR  
HEARTH  
MATERIAL  
SIZE  
STONE

Figure 6-14 Entering concepts and attributes

The Sketch-Pad is removed when the user feels there are no more concepts to be added. However, subsequently, new knowledge can be added directly into the CCW, which can be opened, (if closed), by clicking on the *central concepts* page of the Notebook.

---

---

#### Figure 6-15

The Sketch-Pad is removed after 'enough' concepts and attributes have been entered. The CCW, however, stays. This is the main window for knowledge elicitation. It can be opened by clicking on the *central concepts* page of the Notebook.

Besides entering new concepts, the window allows editing of concept attributes and identifying relations between concepts. When a concept is clicked on (with a right mouse click), a menu for editing the concept's attributes is presented. The menu allows three options: adding a new attribute, delete an attribute and edit a given attribute.

Editing the attribute values. The user is first informed about what the possible values of the attribute are. S/he is prompted for the new value type and the possible values for the attribute.

---

---

---

---

#### Figure 6-16

Adding a new attribute. The user is informed about the reserved attributes, those that are system defined and cannot be used again. Then, s/he is told about the other attributes that the concept has.

First, the name of the new attribute is acquired. Followed by its valueclass, what possible values it can have.

---

---

---

---

#### Figure 6-17

Double clicking on a concept in the CCW opens the dual windows for the display of 'supports' (top left) and 'inferred-from' (bottom left) relations.

To identify the associations between *butchering* and other concepts, first *butchering* is selected (by double mouse click on the concept in CCW). Supports Relationships shows which concepts *butchering* supports, Inferred-from Relationships shows the concepts from which *butchering* is inferred. A left mouse click on either of these brings up a window with how to go about identifying a relationship.

---

---

NEW KE | LOAD KE | SAVE KE | QUIT KE | HELP KE

**ASKE**  
 HCRL

---

**Purpose:** Identify relationships between concepts.

**Comments:** In the Central Concepts Window is given a tree of main concept categories for the task of ARCHAEOLOGY in INTERPRETATION. The data and solution categories are:

**Date:** FEATURES ARTIFACTS

**Solution:** SITE.PROFILE .ACTIVITIES

The objective is to draw relationships between categories with those in the solution categories

----- Instructions -----

Clicking mouse button on a concept would enable:

Left 1: Create a new object. Left 2: Middle

Middle 1: Delete/Rename object. Middle

Right 1: Edit object. Middle

Clicking anywhere else in the window would enable:

Left 1: Redisplay. Left 2: Middle

Middle 1: Toggle display. Middle

Right 1: Flush window (quit). Right

**Concept:** METAL

**Attribute:** USE

**Possible values:** ((ONE OF TOOL ORNAMENT))

Enter new possible value type: (choose one of the following):

- 1 NUMERICAL-VALUE
- 2 LIST-OF-VALUES
- 3 UNKNOWN

=> 2

Enter possible values:

<list-of-words> weapon tool ornament

----- Instructions -----

Clicking mouse button on a concept would enable:

Left 1: Create a new object. Left 2: Middle

Middle 1: Delete/Rename object. Middle

Right 1: Edit object. Middle

Clicking anywhere else in the window would enable:

Left 1: Redisplay. Left 2: Middle

Middle 1: Toggle display. Middle

Right 1: Flush window (quit). Right

[For further help on identifying concept categories for Duration (Concepts in HELP menu)]

aske>

Figure 6-15 Editing concept attributes

NEW LOAD SAVE QUIT HELP  
 KB KE KB

RULEMAKER  
 IF THEN

ASKE  
 SETTLEMENT-SITE  
 ASKEDATA  
 System KEs

ASKE  
 HCURL

**Purpose:** identify relationships between concepts.

**Comments:** In the Central Concepts Window is given a tree of main concept categories for the task of ARCHAEOLOGY in INTERPRETATION. The data and solution categories are:

Date: FEATURES ARTIFACTS

Solution: SITE.PROFILE ACTIVITIES

The objective is to draw relationships between categories with those in the solution category

----- Instructions: -----

Clicking mouse button on a concept would enable:

Left 1: Create new object. Left 2: Middle

Middle 1: Delete/Rename object. Middle

Right 1: Edit object. Right 2: Middle

Clicking anywhere else in the window would enable:

Left 1: Redisplay. Left 2: Middle

Middle 1: Toggle display. Middle

Right 1: Flush window (quit). Right 2: Middle

-----

[For further help on identifying concept categories for Domain (Concepts in HELP menu)]

(Reserved attributes:  
 TYPE EVIDENCE INFERRED FROM SUPPORTS T.RULES )

(Other attributes:  
 CONTENT )

Enter an attribute of FLOOR:  
 <word> size

Enter possible value type for SIZE (choose one of the following):

1. NUMERICAL-VALUE
2. LIST-OF-VALUES
3. UNKNOWN

=> 2.

Enter list of values for SIZE  
 <list-of-words> small medium large

ACTIVITIES: BUTCHERING  
 FOOD PREPARATION  
 METAL PRODUCING  
 POTTERY MAKING  
 STORAGE AREA

ARTIFACTS: METAL  
 POTTERY  
 STONE

FEATURES: DITCH  
 FLOOR  
 HEARTH  
 PIT

SITE PROFILE: EXCHANGE CONTACTS  
 OCCUPANCY  
 SOCIAL STATUS

Figure 6-16 Adding a new attribute

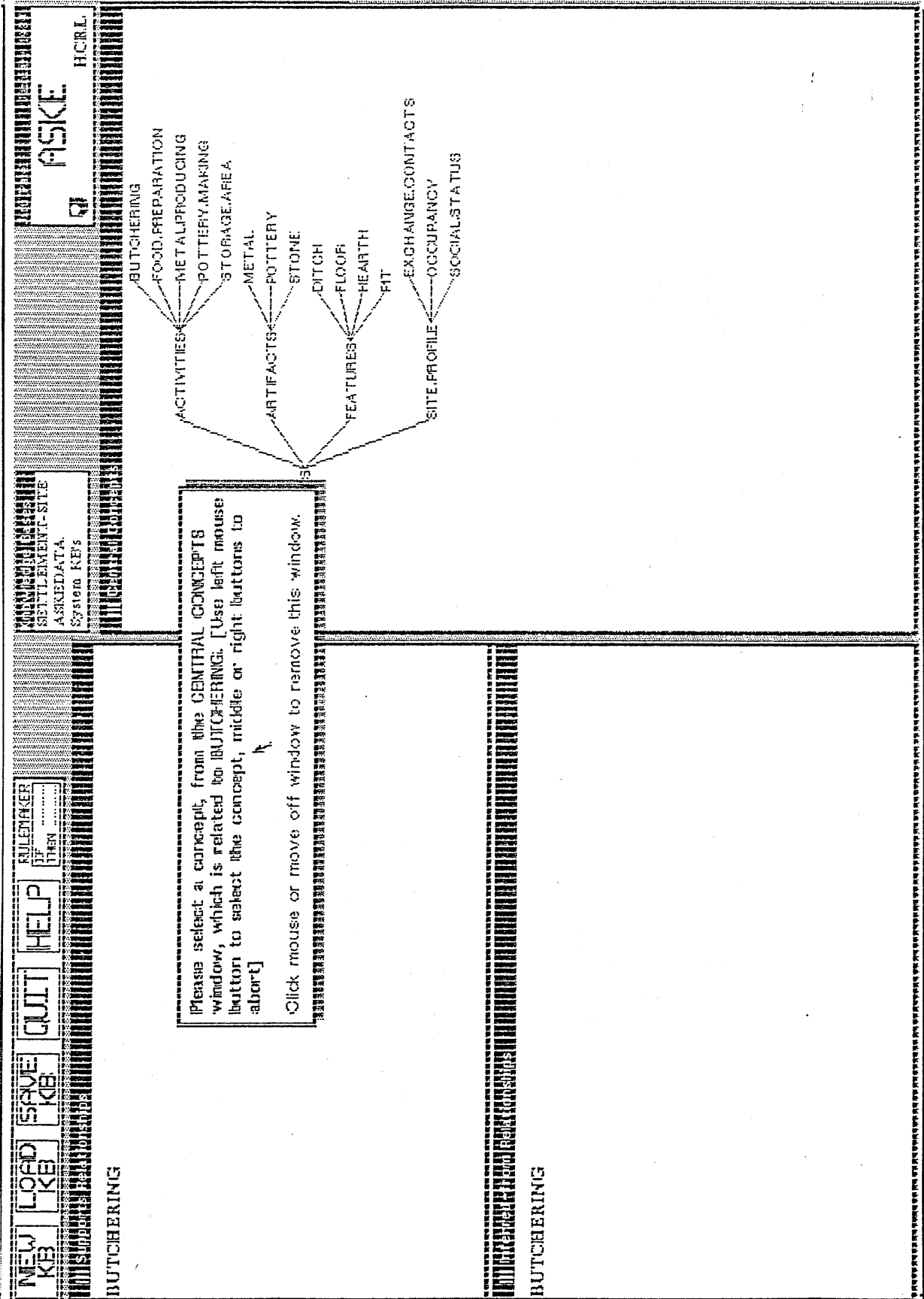


Figure 6-17 Types of relationships



### 6.4.2 Identifying Relationships

After the concept categories have been obtained, the next task is to identify the associations between concepts. Double-clicking the middle mouse button on a concept in CCW opens up the Relations Window, which is split into two halves, top and bottom displaying the "supports" and "inferred-from" relationships for the concept, respectively (Figure 6-17).

ASKE knows the relationship between data and solution categories the former supports the latter or the latter is inferred-from the former. But, if two concepts from the same main category are identified to be associated, the user is prompted for the exact relationship between the two, as shown in Figure 6-18. The following shows how links are created in ASKE.

```
In every frame there four important attributes:
  type - indicates whether the concept is data or solution category;
  inferred.from - from which concept is this one inferred;
  supports - which concept does this one support;
  t.rules - the rules which have this concept.

; This function identifies the relationship between concepts.
; It takes two arguments: 1) the main concept, which is in
; Relations Window, and 2) the concept in the CCW.
(defun make-a-relationship (concept1 concept2)

  ; find the 'type' of the two concepts
  (let ((forward-link (find-type-of-concept concept1))
        (backward-link (find-type-of-concept concept2)))

    ; if the concepts are of the same type
    (if (equal forward-link backward-link)

        ; find the direction of the link, and accordingly assign
        ; the type value to concept1
        (setq forward-link (prompt-for-link-direction))

        ; if the type of concept1 is solution
        (if (equal forward-link 'solution)

            ; then assert that concept1 supports concept2
            (add-link-type concept1 concept2)

            ; else assert that concept2 supports concept1
            (add-link-type concept2 concept1))))))
```

While adding a link, ASKE also creates a temporary rule. The rules are generated at link creation time, and not after all links are made, because the two tasks of link and rule making are associated. What happens at the time of concept linking is, given two concepts  $\partial$  and  $\Omega$  with the relationship:  $\partial$  supports  $\Omega$ , the following values are added.

	supports	inferred-from	t.rule	temp.rule12	
$\partial$	$\Omega$		temp.rule12	IF	$\partial$
$\Omega$		$\partial$	temp.rule12	THEN	$\Omega$

A new rule 'temp.rule12' is created with  $\partial$  as premise and  $\Omega$  as conclusion. The rule, called association rule, is added to the 't.rules' slot of each concept. All this is carried out by the 'add-link-type' function.

ASKE generates temporary rules, which are held in a tree structure with the root node temp.rules. The rule has two slots: premise and conclusion. These slots take concepts as values. Rules with the same conclusion are clustered together in a class, which takes the name of the conclusion with a suffix '.t'.

*; This function creates a link between concepts and builds a temp rule. It takes two arguments: 1) the supporting concept; and 2) the supported concept.*

```
(defun add-link-type (concept1 concept2)
```

```
  ; add concept2 to the value of supports in concept1
  (add-value concept1 'supports concept2)
```

```
  ; add concept1 to the value of inferred-from in concept2
  (add-value concept2 'inferred-from concept1)
```

```
  ; find the rule class; if none exists, create one
```

```
  (let* ((class (cond ((class-exists concept2))
                     (t (create-class concept2))))
```

```
    ; create a new rule in the rule class
```

```
    (rule (create-rule (gentemp "temp.rule") class)))
```

```
  ; add concept1 and concept2 to the premise and conclusion
  ; slots of the new rule
```

```
  (add-value rule 'premise concept1)
```

```
  (add-value rule 'conclusion concept2)
```

```
  ; add new rule to the value of t.rules in the two concepts
```

```
  (add-value (list concept1 concept2) 't.rules rule)))
```

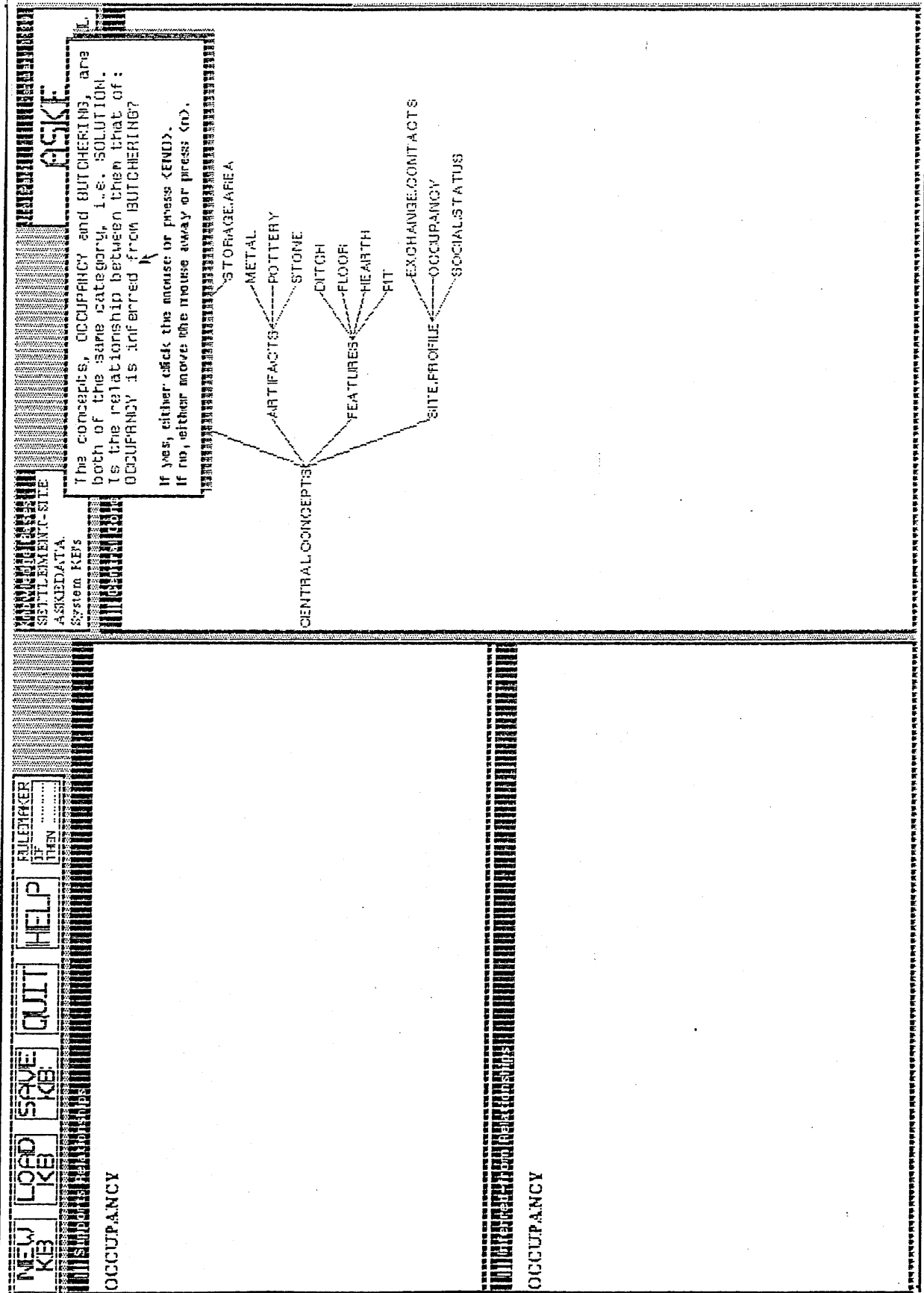


Figure 6-18 Same category concepts

---

---

**Figure 6-18**

ASKE knows that data categories concepts support concepts from the solution categories. (This information is given in the ATEMP and is included in the task model.)

When two concepts from the same category are identified to be associated, ASKE asks for the nature of relationship. For example, *butchering* (which is an activity) and *occupancy* (which is a site.profile) are related. They are from the same category, solution of which both activities and site.profile are members. The user is prompted for the direction of relationship.

Butchering supports occupancy.

---

---

### 6.4.3 Querying about Unaccounted Concepts

The user can quit from the relationship identification stage at any time. ASKE outputs a status message informing the user of the concepts whose associational links haven't been identified (Figure 6-19). The user can either go back to the previous stage or proceed to the rule-editing stage.

---

---

**Figure 6-19**

After the user quits from the relationship identifying stage (which is done by closing the CCW), ASKE searches the list of domain concepts for any unrelated concepts: those for which no associations have been identified. The user is informed about the unidentified concepts, if any.

Ditch, from the data category is still unattached.

---

---

NEW KB | LOAD KB | SAVE KB | QUIT | HELP

FULLERKER  
BY  
THEN

ASKE  
HCRL

---

SETTLEMENT-SITE  
ASKEDATA  
System KE'S

ARCHAEOLOGY

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

---

1 DITCH

The relationships of the following data concepts have not yet been identified.

.....

You can go back to the concepts creating window by clicking on the Central Concepts Window of the Notebook.

ARCHAEOLOGY

ACTIVITIES

ARTIFACTS

FEATURES

SITE PROFILE

CENTRAL CONCEPTS

INTERPRETATION

SETTLEMENT-SITE

BUTCHERING

FOOD PREPARATION

METAL PRODUCING

POTTERY MAKING

STORAGE AREA

METAL

POTTERY

STONE

DITCH

FLOOR

HEARTH

PIT

EXCHANGE CONTACTS

OCCUPANCY

SOCIAL STATUS

TEMP. FRU

ASKEDATA

ASKED

ASKE

Figure 6-19 Informing about unaccounted concepts

The processing for this stage is carried out by the following lisp functions.

```

; This function looks for any unattached concept.
(defun find-unfilled-concepts ()

  ; get all concepts in data and solution category
  (let* ((data (get-all-concepts-in-data-category))
         (soln (get-all-concepts-in-solution-category))

         ; find unrelated concepts
         (u-data (find-unrelated-concepts data))
         (u-soln (find-unrelated-concepts soln)))

    ; if there are any concepts unaccounted for
    (when (or u-data u-soln)

      ; inform the user about them
      (if u-data (inform-user u-data))
      (if u-soln (inform-user u-soln))))))

; This function recursively checks unrelated concepts. It takes
; one argument. a list of concepts.
(defun find-unrelated-concepts (c-list)

  ; continue checking the list until it is empty
  (when c-list

    ; check if any rules are created for the concept
    (if (get-value (car c-list) 't.rules)

      ; if not, go to the next concept
      (find-unrelated-concepts (cdr c-list))

      ; if yes, store the concept; and continue checking
      (cons (car c-list) (find-unrelated-concepts (cdr c-list))))))

```

## 6.5 Rules Editing

The rules generated at the previous stage (i.e., associational rules) are displayed in the Rulemaker for editing (See Appendix A for the complete list of ASKE generated rules for this session). Rules are arranged in tree structure with the 'temp.rules' as the root rule. The rest of the rule hierarchy is organized in contexts, classes and associational rules.

An association rule was created with a single concept each in the premise and the conclusion. At the editing stage, the various slots of the concepts are conjuncted with the concept. The user is required to edit the values as well as merge various rules.

### Rules Merging

Figure 6-20 shows how the various rules, whose conclusion is the activity of *storage.area* are merged. Merger of rules is accomplished by the following function.

```
; This function merges a given set of rules into one. It takes two
; argument: 1) rules to be merged; and 2) the replacement rule.
(defun merge-rules (setof-rules new-rule)

  ; go through the list of rules till it is empty
  (when setof-rules

    ; get the first rule in the list
    (let* ((rule (car setof-rules))
           (other-rules (cdr setof-rules)) ; rest of the rules
           (prem (get-value rule 'premise)) ; get premise
           (conc (get-value rule 'conclusion))) ; and conclusion

      ; add the premise and conclusion of the rule to the resp.
      ; slots of the new rule; add only if value doesn't exist
      (add-new-value new-rule 'premise prem)
      (add-new-value new-rule 'conclusion conc)

      ; delete the rule and the corresponding links from
      ; its premise and conclusion concepts
      (remove-rule rule)

      ; recurse with the rest of the rules
      (merge-rules other-rules new-rule))))
```

### Editing the Premise and the Conclusion

The merged rule, *temp.rule48*, is displayed in the Rule Display Window. The premise and conclusion can be edited, individually, in the Rule Editing Window. Basically, what is displayed in the Rule Editor is a list of concepts. A concept list consists of the concept as the first item and its

attributes making up rest of the list. For example, the premise of temp.rule48, displayed in the buffer of Rule Editing Window (Figure 6-21) consists of a list of 3 lists:

```
( ( (floor)
    (content features)
    (size (small medium large)) ) ----- 1
  ( (pottery) )
    (surface (blackened not.blackened))
    (size (small medium large))
    (shape (circular cylindrical oblong))
    (fabric (fine coarse)) ) ----- 2
  ( (pit)
    (use (storage refuse posthole)) ) ) ----- 3
```

The task at hand is to delete attributes and values which do not play a role in reasoning. This is accomplished with the 'rubout' key and the mouse cursor. At the end of editing, the Rule Editing Window is removed and the contents of the buffer displayed in the Rule Display Window. The rule is finally stored in an intermediate form:

```
Temp.rule48
IF
  The content of FLOOR is (POTTERY PIT)
  The size of FLOOR is SMALL
  The size of POTTERY is LARGE
  The fabric of POTTERY is COARSE
  The decoration of POTTERY is PLAIN
  The use of PIT is STORAGE
THEN
  activities is STORAGE.AREA
```

### Intermediate-level Rules

From the present session, 12 intermediate rules were generated. These include 7 for inferring the activity of an area and 5 for inferring site.profile. These are given in Appendix B.



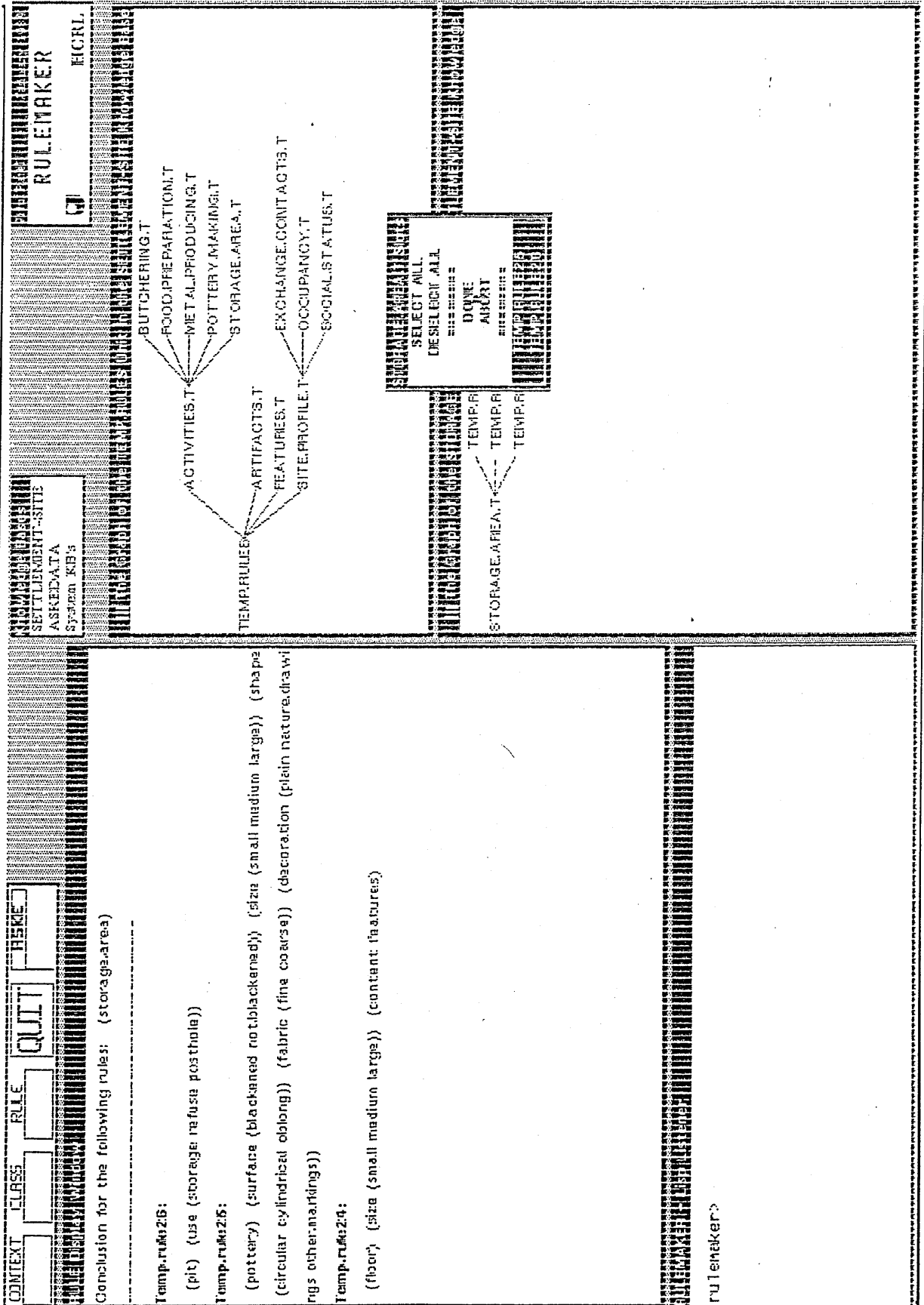


Figure 6-20 Merging rules

CONTEXT CLASS RULE QUIT HERE

RULEMAKER

FCRL

SETTLEMENT-SITE  
ASKEDATA  
System KIB's

TEMPRULES48

```

      ACTIVITIES.T --- BUTCHERING.T
      METAL.PRODUCING.T
      POTTERY.MAKING.T
      STORAGE.AREA.T
      EXCHANGE.CONTACTS.T
      OCCUPANCY.T
      SOCIAL.STATUS.T
      ARTIFACTS.T
      FEATURES.T
      SITE.PROFILE.T
    
```

STORAGE.AREA.T --- TEMP.FULE4B

Temprule48

```

      IF
      (floor)
      (content features)
      (size (small medium large))
      (pottery)
      (surface (blackened not.blackened))
      (size (small medium large))
      (shape (circular cylindrical oblong))
      (fabric (fine coarse))
      (decoration (plain mature.drawings other.markings))
      (pit)
      (use (storage refuse posthole))
      THEN
      ((FLOOR)
      (CONTENT (pottery pit))
      (SIZE SMALL))
      ((POTTERY)
      (SIZE LARGE)
      (FABRIC COARSE)
      (DECORATION PLAIN))
      ((PIT)
      (USE STORAGE))
      )
    
```

PREMISE (TYPE END TO EXIT)

Figure 6-21 Editing the premise

---

---

**Figure 6-20**

From the associations between concepts, ASKE automatically generates rules. These rules are organized in hierarchies of contexts, classes and rules. The root node of the rules tree is *temp.rules*. The contexts are obtained from the main concept categories. Hence, there are four contexts: *activities.t*, *artifacts.t*, *features.t* and *site.profile.t*.

The rules that ASKE creates are very simple: from 'metal supports metal.producing' is created 'if metal then metal.producing'. All the rules with the same conclusion are clustered together into classes. There are eight main classes and they correspond to the concepts of the solution type.

Rules within classes can be merged to produce a complex rule. For example, *temp.rule24*, *temp.rule25* and *temp.rule26* are merged into *temp.rule48*.

---

---

---

---

**Figure 6-21**

*Temp.rule48* is displayed in the Rule Display Window. Its premise and conclusion can now be edited. By clicking the left mouse button when the cursor is on the window, the Rule Editing Window is opened with the premise placed in its buffer; (middle button puts the conclusion in the buffer).

---

---

## 6.6 Creating an RTEMP

The last thing that ASKE does, is automatically create an RTEMP from the knowledge base. The RTEMP takes the name of the WTEMP with a suffix '.R'. Figure 6-22 shows the new RTEMP created from the Settlement Sites knowledge-base.

---

---

**Figure 6-22**

At the end of the ASKE session, a new RTEMP is created by abstracting information from the current knowledge-base. This is displayed in the Aske Interface in the Interaction Window.

---

---

NEW  
KEY

LOAD  
KEY

SAVE  
KEY

QUIT  
KEY

HELP

RULEMAKER  
MENU

ASKE

HCR.L

SETTLEMENT-SITE  
ASKEDATA  
System KEYS

WATERSHED

RTEMP: SETTLEMENT.SITE.R

EXPERT: Jltu

DOMAIN: Archaeology

TASK: Interpretation

SPECIALIST AREA: Settlement-site

TASK DESCRIPTION:  
 Inferring activity areas in a settlement site from archaeological data

---

Data Types:  
 FEATURES (geographical and geological features of the land, including how it has been modified by man)  
 ARTIFACTS (main archaeological data for inferring activities of an area or a room)

Solution Types:  
 SITE.PROFILE (general characteristics of the settlement site as inferred from the distribution of artifacts and features)  
 ACTIVITIES (the kind of activities that took place in a given settlement site)

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

ARCHAEOLOGY

ACTIVITIES

ARTIFACTS

CENTRAL CONCEPTS

INTERPRETATION

SETTLEMENT.SITE

SETTLEMENT.SITE.R

BUTCHERING

FOOD.PREPARATION

METAL.PRODUCING

POTTERY.MAKING

STORAGE.AREA

METAL

POTTERY

STONE

CATCH

FLOOR

HEARTH

PIT

EXCHANGE.CONTACTS

OCCUPANCY

SOCIAL.STATUS

BUTCHERING.IT

TEMP.FINAL

asked>

Figure 6-22 The new RTEMP

# Chapter VII

## DISCUSSION

*By examining the processes of your own expertise you risk becoming like the centipede who got tangled up in her own legs and stopped dead when she tried to figure out how she moved a hundred legs in harmony.*

### 7.1 Introduction

In this Chapter, I would like to bring out in the open some of the assumptions guiding the design of ASKE. I will start the discussion with a look at the template approach to KA, as it is at the heart of the methodology presented in ASKE. What is the template approach? Is it any good? What advantage has ASKE got over other KA systems? I will then go on to discuss the notion of KA as a modelling activity. Is it justified? What are the methodological implications of this?

Many of the KA systems, including ASKE, use the model of heuristic classification as the paradigm for developing expert systems. The main question here is relating the scope of the model. How generic is a tool if it can work for only a circumscribed area? I will try to answer some of these questions. Finally, I will argue that the methodology for automatic KA is more than just an intelligent interface. Why is ASKE not just an interface to KEE? What role are intelligent interfaces to play in the design of KA systems? A description of the limitations of the ASKE approach to KA and research that requires to be done is presented in the following Chapter.

## 7.2 Template driven KA

The first question, and probably the most important one, is: what is a template? With respect to the structure, one can say that a template is a frame. But, templates do not have any mechanism for inheritance, which is one of the characteristic features of frames. This is essentially because such a feature is not required for the kind of task that templates are utilised for. Templates play a number of roles in ASKE. They direct interviewing (e.g., GTEMP and ATEMP); they help the domain expert in focussing attention at the right level for concept identification (e.g., RTEMP); they provide a structure for representing domain knowledge (e.g., WTEMP).

How good is the template approach? How does it help in KA? To answer these questions we will have to look at what has been achieved by ASKE.

### 7.2.1 What ASKE started with

Initially, ASKE contained three hand-crafted knowledge bases: selection of Software Marketing strategies (Corporate Planning), interpretation of Burial Sites (Archaeology), and diagnosis of Foot Problems (Medicine). Hence, ASKE started with the following templates:

GTEMP	WTEMP	RTEMP
Corporate.Planning	Software.Marketing	Software.Marketing.R
Archaeology	Burial.Sites	Burial.Sites.R
Medicine	Foot.Problems	Foot.Problems.R

Furthermore, ASKE has four ATEMPs: debugging, diagnosis, interpretation and selection. These ATEMPs, along with the GTEMPs and WTEMPs, were hand-crafted. The RTEMPs were generated automatically from the knowledge bases.

### 7.2.2 What has been achieved

The manually acquired knowledge bases have been successfully applied in the automatic acquisition of four new knowledge bases, two of which are in new domains. The application areas (WTEMP) are as follows:

GTEMP	WTEMP	ATEMP	RTEMP USED
Archaeology	Settlement.Sites	Interpretation	Burial.Sites.R
Medicine	Aids.Infections	Diagnosis	Foot.Problems.R
Motor.Mechanics	Engines	Diagnosis	Aids.Infections.R
Nursing	Pressure.Sore	Diagnosis	Aids.Infections.R

By comparing what ASKE initially started with and what it has achieved, we can say that the approach works. The main advantage of the template approach is that the system is able to build knowledge-bases in completely new domains. ASKE does this by utilizing the existing templates as case examples for acquiring new templates from the domain experts.

It must be pointed out that in the acquisition of all these knowledge bases I was, either directly or indirectly (i.e., I sat next to the domain expert), the user of ASKE. In the latter case, it was necessary because the interface was not 100% crash-proof (see Section 7.5 for more on the Interface). Furthermore, in all instances, the knowledge was obtained, using the methodology, from an "academic" expert, whose domain knowledge tends to be logically structured (Shadbolt and Burton, 1989). According to Shadbolt and Burton, there are three types of experts: academic, practitioner and samurai, who may perform differently in the KA situation. It is hence hard to predict how successful ASKE will be in acquiring knowledge from the other two types of experts, who tend to be much more interested in performance than in the application of theory. An academic expert as a knowledge source is, however, not such a big limitation since many of the well known expert systems (e.g., DENDRAL, MYCIN and ONCOCIN) were built using academic experts.

### 7.2.3 Comparison with other systems

What advantage has ASKE got over other systems? So far, no knowledge-based KA system has been able to achieve the scope of applicability of the technique-based KA systems. For instance, MOLE can only be used in diagnostic applications; ETS can be applied to analysis tasks. Both systems, like ASKE, work within the heuristic classification paradigm (see Section 7.4 for more on this). With the template approach, ASKE has achieved the breadth of scope of the technique-based KA systems. It can be used to develop systems for any analysis task, theoretically. In practice, ASKE has ATEMPs for the four task types, but future extensions (see Section 8.2.2) to ASKE will allow more flexibility in creating new ATEMPs.

The designers of both KNACK and PROTEGE, the other model-based KA systems, argue that their systems can be applied to different tasks. Neither has demonstrated this, however. One possible reason for this is that they are limited by their problem-solving methods of acquire-and-present (KNACK) and skeletal-plan-refinement (PROTEGE). Both of these methods are very specific. The advantage of having such strong problem-solving methods is that they facilitate the recognition of concepts for correct problem definition. And this is important, as Kitto (1988:14-7) points out:

"A common source of confusion for most domain experts is creating the conceptual model. Experts are unsure of which terms in their domains constitute *concepts* which are *characteristics*, and which are specific examples (instantiations) of characteristics."

The limitation of powerful problem-solving methods is that they have a rather narrow scope of applicability. However, by using a more general problem-solving method, like heuristic classification, one loses the advantage of the specialist methods. There is no provision for identifying important concept categories required for the problem definition. ASKE



solves this problem by using previous cases (i.e., RTEMPs), which with the ATEMPs, are used to provide the domain expert with the kind of help that would be available in the more specific problem-solving methods.

### 7.3 KA is a modelling activity

ASKE is founded on the tenet that KA is a modelling activity. The idea that KA is a modelling enterprise is in vogue with researchers working in the field (e.g., Motta et al., 1989b; Musen, 1988; Wielinga and Breuker, 1986). It is also consistent with the general view that "knowledge bases contain models of systems in the world" (Clancey, 1989:10). There are two main questions that need to be answered: What is the justification for doing so? What are the methodological implications of adopting this approach? I will start with the second question first.

The basic idea underlying the view that KA is a modelling activity is that the model of the task developed in the initial stages of KA can be used to elicit problem-solving knowledge. Implicit in this perspective is the notion that KA can be carried out in a top-down fashion (Motta, et. al., 1989b). The main implication for the designers of KA systems is that the system should start with some knowledge or strategies for formulating an initial model of the application. Thus, unlike the bottom-up approach where KA is carried out with just the knowledge of different elicitation techniques, the emphasis is now on the use of task-specific knowledge. Indeed, this is the thrust behind the task-specific and knowledge analysis methodologies of knowledge engineering. The aim of the former is to use role-limiting methods (McDermott, 1988) or generic tasks (Bylander and Chandrasekaran, 1987), while the latter employs interpretation models (Wielinga and Breuker, 1986) for KA.

According to Chandrasekaran (1987), "knowledge systems should be built out of building blocks, each of which is appropriate for a basic type of

problem solving" (1183). The building block (or generic task) uses forms of knowledge and control strategies that are characteristic of it. In ASKE, the task-specific knowledge is hard-wired into the ATEMPs.

### 7.3.1 Task and Interpretation models

The notion of task model in ASKE is akin to that of interpretation model in KADS (Breuker and Wielinga, 1985); it describes the meta-level structure of a generic task. Also, both task and interpretation models support top-down KA. At the start of a KA session, the model that best fits the problem area is chosen from a library of models (task or interpretation) and applied to it. One of the main differences between the two is the real purpose they serve. The task model helps the expert focus his/her attention on the concepts that are important for describing the task. This approach is identical to that prescribed by the task-specific methodology. The interpretation model plays a role in the knowledge analysis stage. It provides guide-lines for interpreting data:

"an interpretation model is a kind of catalogue of types of ingredients the knowledge engineer can look for in the data, and thus functions as an organizer that provides coherence to these data" (Wielinga and Breuker, 1986:22).

A more significant difference between the task and interpretation model is that while the former is automatically selected, the selection of the latter is in the hands of the knowledge engineer.

### 7.3.2 Applicability of models

The view that knowledge bases are models endorses the fact that they are based on assumptions and prone to failure. As a result the designer of a system employing models to drive KA has an onus to determine its range of applicability. This is indeed a non-trivial task, as experience suggests.

"Clever shortcuts and elegant formalities are worthless unless the experts can fit their own knowledge into the framework provided by the designer. Only when a program's vocabulary is 'natural' to experts can they help refine and augment the knowledge base to bring the system's performance up to their own level of expertise" (Buchanan, 1982).

When models are used as an acquisition tool, there is a great potential for a mismatch between the model and the problem it is supposed to represent. This does not, however, render the modelling approach fruitless because "the use of pre-existing models, even inadequate ones, can lead to dramatic improvements in the time required for performing KA and building a prototypical system" (Motta et al., 1989a:317). Therefore, the main justification for the modelling approach is that it facilitates KA.

#### 7.4 Heuristic Classification Paradigm

Clancey (1985) analyzed a number of expert systems and found that these programs exhibited a similar pattern of reasoning:

"These programs proceed through easily identifiable phases of data abstraction, heuristic mapping only a hierarchy of pre-enumerated solutions, and refinement within this hierarchy. In short, these programs relate concepts in different classification hierarchies by non-hierarchical, uncertain inferences. We call this combination of reasoning heuristic classification" (290).

The heuristic classification model has been used by a number of KA systems (including ASKE) as the basic method of problem-solving. The model has been argued to be applicable to solving analysis tasks, which are characterized by the fact that all their possible sets of solutions can be specified a priori. As mentioned earlier (Section 4.2), the applicability of the model is restricted to "structured" problems only. A further restriction is that the task should not be time-critical, i.e., the solution of the problem should not vary with time. When a solution does vary with time, a third

factor is introduced to the pre-existing link between problem description and solution. However, the heuristic classification model does not cater for the time element. Therefore, tasks such as monitoring which are usually time-critical will not be solvable by the classification method. Given these limitations of the scope of the model, the main question is: how generic is a tool based on the heuristic classification model?

#### 7.4.1 Heuristic Classification and KA Systems

Boose (1988) has shown that there is a possible association between tasks and problem-solving methods. Figure 7-1 shows the mapping between analysis tasks and heuristic classification. There is a similar mapping between heuristic construction and synthesis tasks.

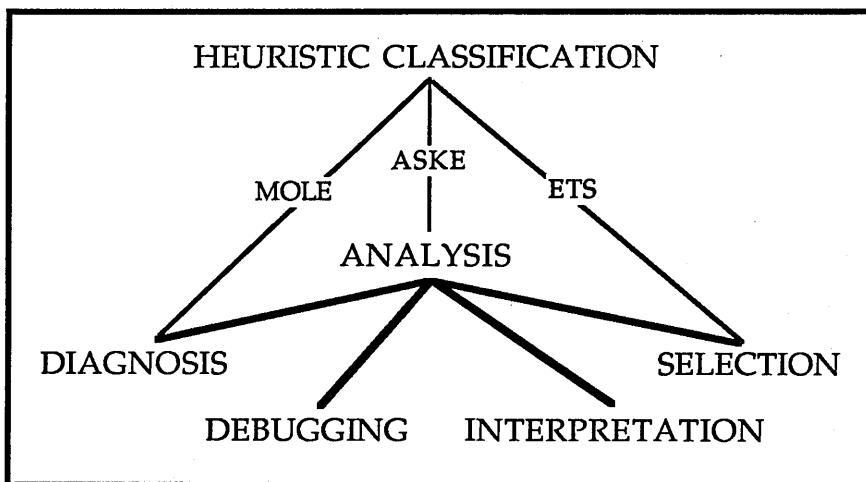


Figure 7-1 KA systems linking analysis tasks and heuristic classification (from Boose, 1988)

The Figure shows that there is a possible mapping between problem-solving methods and application tasks, and the association is manifested in the KA tools. The three systems: ETS, MOLE and ASKE, use the heuristic classification problem-solving method. In all three systems, the method is hard-wired into the system. Figure 7-2 summarises the essential

differences between these systems. In ETS the classification method plays a rather secondary role to the repertory grid methodology. The knowledge elicitation stage is carried out using the repertory grid technique. It is only at the latter stages of generating rules that knowledge is organized according to the model specifications.

	ETS	MOLE	ASKE
Technique for knowledge elicitation	Repertory grid method	Cover-and-differentiate, which is based on the heuristic classification model	Based on the heuristic classification model
Type of knowledge to elicit	Implicit in the grid method	Implicit in the above technique	Explicit in the ATEMPs
Breadth	Selection and simple classification tasks	Diagnostic tasks	Analysis tasks

Figure 7-2 How ASKE differs from ETS and MOLE.

In MOLE, the classification model is used as the general problem-solving method on which more specific method, called role-limiting method, is based. A role-limiting method "defines the roles that the task-specific knowledge it requires must play and the forms in which that knowledge can be represented" (McDermott, 1988:228). Essentially, a role-limiting method consists of a repetitive cycle of procedures for identifying and processing required knowledge. Thus, MOLE utilizes the power of the classification model to a greater extent and more effectively than ETS. The elicitation stage is driven by the role-limiting method, which is based on the classification model. However, the range of applicability of MOLE is quite narrow: it can be used for diagnostic tasks only.

In ASKE the classification model is implicit in the ATEMP. To make the system more general, the use of a more specific problem-solving methods was avoided. Instead, the classification model is used directly to carry out knowledge elicitation. The kind of knowledge to be elicited is separated from the elicitation technique and represented explicitly in the ATEMPs. This way, the full scope of the classification model has been achieved in ASKE. Relative to other KA systems, ASKE has a greater scope of applicability (or breadth). But still, with the inference structure implicit in the design, the system is limited. With further extensions, (about which more is said in the next Chapter), it will be possible to make the classification problem-solving method explicit.

#### 7.4.2 Problem-Solving Methods and Generic Tasks

Chandrasekaran (1987) suggests that KA should not be considered in isolation, but rather should be conceived of as an adjunct to some specified problem-solving task. The task is referred to as *generic task*, which is "an elementary generic combination of a problem, representation, and inference strategy about concepts" (Bylander and Chandrasekaran, 1987:235). In generic tasks, they have attempted to identify the modules of problem-solving needed to address a given set of task demands. In so, generic tasks appear more like problem-solving methods: they are more descriptive of the problem-solver than the application task (Woodward, 1989). Compared to heuristic classification, generic tasks are elementary problem-solving methods closely associated to application tasks. Heuristic classification, on the other hand, is linked to application tasks by KA tools.

### 7.5 Intelligent interfaces

One of the important factors in manual KA is the effectiveness of the knowledge engineer in communicating with the domain expert. On automating the task, the communication factor is not removed, but it is displaced. The issue of man-machine interface has acquired a new meaning. In designing ASKE, it was evident that interface issues are tied

in with the KA methodology. For example, the second stage of KA relies on the use of mouse and menus for encoding knowledge. The main reason for this is because "the question-and-answer style of entry is extremely tedious, cumbersome and time-consuming" (Kitto, 1988:14-3).

In this final section, I will try to answer two critical questions: Why is ASKE not just an intelligent interface? Where do intelligent interfaces figure in in the design of KA systems?

### **7.5.1 ASKE is more than an intelligent interface to KEE**

At first glance, ASKE looks like a front-end to KEE. And, this is not entirely an incorrect perception. ASKE does provide a facility for encoding knowledge directly into KEE. However, ASKE is designed as a stand-alone system. It is a KA system and it produces intermediate rules which can be translated into KEE or KEATS rules. Because the templates are implemented in frame-language, ASKE can use any environment with this facility.

Another characteristic which distinguishes ASKE from an intelligent interface is that it is goal driven. Embedded in the ASKE system is a methodology for KA and the interface is just a facility for interacting with the program.

### **7.5.2 Intelligent interfaces and KA systems**

Intelligent interfaces are necessary if the system is going to be used for real applications. Indeed, this is true for computer systems in general.

"For a high-performance computer program to capture the sustained, widespread attention of working scientists, it must contain a large number of features that make it easy and pleasant to use. These features are commonly termed 'human engineering aspects' of a program. In very rare instances, a program will be so useful that it will be widely adopted even without proper attention to human engineering. But the general principle seems to be that

programs that are only understandable to programmers are used only by programmers, if at all" (Buchanan, 1979).

The implication for a KA system, however good its methodology may be, is that without a good interface there is very little chance of it succeeding outside of the place of its development. For instance, both KNACK and AQUINAS were found difficult to use when field tested . Both tools required "much progress in the development of intelligent interfaces to improve usability by a domain expert" (Kitto, 1988).

In ASKE, a lot of effort has gone into designing the interface. For example, the Sketch-Pad, the Central-Concepts Window, and the Relations Window were designed to facilitate the encoding of domain concepts and their relations. The Notebook facility provides a quick access to the knowledge-base. The Rulemaker Interface was designed for creating and editing rules. In spite of these facilities, ASKE is not easy to use. More work is needed, especially in the way of documentation and help facilities.



# Chapter VIII

## CONCLUSION

*Some people will believe they were better off in the good old days.*

### 8.1 Recap

In this concluding Chapter I would like to summarize the work presented in this thesis. I will contain myself to answering the following questions: What was the main goal of this thesis? What has been achieved? What has not been accomplished? Finally, I present suggestions for future research.

#### 8.1.1 Project Aims: Revisited

The main purpose behind the research in automatic KA is to provide tools that either supplement or replace the knowledge engineer in developing expert systems. The present work is a contribution to the latter. It is rather an ambitious objective and to make it more tractable, the main goal of this research was limited to: the design of a tool that could be used by domain experts to develop prototype systems for analysis tasks.

Analysis of the performance of existing KA systems have shown that there is a tradeoff between the scope of applicability (or breadth) and quality of knowledge bases developed (or depth). Knowledge-based KA systems score on depth but lose out on breadth; technique-based KA systems do better on breadth and perform poorly on depth. The aim of the thesis was to provide a methodology and an implemented tool which will cut through

the breadth-depth problem.

### 8.1.2 What has been accomplished.

A two-stage model of KA was implemented in a computer program, called ASKE. In the first stage, ASKE performs like a technique-based KA system to produce a model of the application task. In the second stage, this task model provides the driving force for KA; thus it acts in the fashion of a knowledge-based KA system.

ASKE derives its power for KA from the knowledge representation scheme of templates. Four types of templates are utilized for the acquisition and representation of problem-solving expertise:

- 1] GTEMP, the general template, is used for the acquisition of general problem knowledge such as the domain classification and project goals.
- 2] ATEMP, the acquisition template, contains task-specific information which is utilized in the acquisition of the task model from the domain expert.
- 3] WTEMP, the working template, is used for representing the domain knowledge.
- 4] RTEMP, the reference template, is an abstracted knowledge base, derived from the WTEMP and used as a guide in the acquisition of the task model.

These templates facilitates the encoding and retrieving of knowledge. They hold generic knowledge about tasks which allows ASKE to target the interview on important aspects of the problem.

The template approach has been used, to date, in the development of initial prototype systems in the domains of Archaeology, Corporate Planning, Medicine, Motor Mechanics and Nursing. From the progress so far, ASKE seems to have achieved the goal of wide scope of applicability: it is suitable for developing prototype systems which solve analysis tasks.

On the breadth factor, the performance of other KA systems (e.g., MOLE, ETS, KNACK and PROTEGE) is not comparable to that of ASKE. For example, MOLE can only be used for diagnostic tasks. The main test for ASKE would be to compare it with a technique-based KA system (i.e., ETS). For this, ETS was rationally reconstructed and the program was called FETS (Figure out ETS). When FETS was applied to interpretation task it performed poorly. FETS could not handle different levels of knowledge (i.e., intermediate steps) necessary for describing the task of interpretation. ASKE, however, does work for the interpretation task.

On the depth factor, ASKE's intermediate-level rules were more expressive than the FETS generated rules. This was mainly because the constructs in FETS are bipolar. For example, the construct "age" can have only two values: old or not-old. It would be a problem to have more than two values for "age" (e.g., child, young adult, middle age, old age) or even numerical values.

ASKE's performance on the depth factor is however not comparable to that of MOLE. The main reason for this is that MOLE consists of a performance component which allows it to produce functional knowledge-bases. ASKE only produces intermediate-level rules therefore a proper evaluation of the system is not possible. The issue of depth has not been solved; but I have suggestions about how to solve it, in Section 8.2.1.

### 8.1.3 The problem areas

There are a number of problem areas, in ASKE, which can do with further work:

- ASKE uses a very simple model of task, based on two or three characteristics. This is because the task characteristics are constrained by the heuristic classification model. The ATEMPs, which use these task

characteristics as a guide are therefore limited in scope. One possible solution is to make the task characteristics independent of the heuristic classification model, which is made explicit.

- All the ATEMPs that ASKE has were initially defined and ASKE does not allow the creation of new ones. In the following section I will describe how this problem may be solved.
- The representation of the problem types is only geared for the acquisition of shallow domain knowledge. It is not clear how the template approach would succeed in acquiring knowledge about reasoning from first principles.
- ASKE does not handle multiple tasks, except for diagnosis and debugging. Thus, if a problem requires both interpretation and diagnosis, the expert is forced to separate the task into two. But, even then, there is no facility for merging two knowledge bases.

## 8.2 Future Research

I would like to describe some of the proposed extensions to ASKE and suggest directions for further research which would go towards taking KA systems outside of research establishments.

### 8.2.1 Performance Component

ASKE has, as yet, not been used for building large applications. But, from its performance at building small prototypes, ASKE is predicted to work as well in bigger applications provided the proposed application is for solving analysis tasks. In its present state, ASKE is best not used for developing large knowledge-bases, for the system does not carry out consistency checking of the rule base.

The final output of ASKE is intermediate-level rules. The system

however needs to produce functional knowledge-bases. For this purpose, further extension in the form of a performance system is proposed. This would enable ASKE to check consistency of the knowledge-base and provide feed-back to the user.

### 8.2.2 Meta-Knowledge

ASKE does not contain any knowledge about itself. It knows about the various templates but the inferencing is a simplistic one. By using meta-knowledge, it should be possible to increase the functionality of the system. Essentially, the meta-knowledge will consist of strategy for acquiring the kind of knowledge that is hard-wired into ASKE, for example, the inference structure and task characteristics. Particularly, ASKE would be able to do the following.

- [1] Creation of new ATEMPs. In the present implementation, the ATEMPs are hand-crafted. It is possible to encode the principles of creating ATEMPs into computer language which can be stored as meta-knowledge and used in generating new ATEMPs. With this new ability, the system will however have to acquire the ATEMPs separate from the knowledge-bases as the expertise encoded in the two comes from different experts. For the ATEMPs, the expert is an experienced knowledge engineer. The ATEMPs automatically acquired from the knowledge engineer can then be used in automatic acquisition of knowledge from domain experts.
- [2] Extending the scope of the system to cover synthesis tasks. ASKE can only handle analysis tasks because it used the problem-solving method of heuristic classification. This is implicit in the ATEMPs. However, by making heuristic classification explicit, it is possible to modify the problem-solving method and make the inference structure independent of the system architecture. By doing so it would be possible to actually acquire the inference structure, that will be used for

KA, acquired interactively from "expert" knowledge engineers.

From the above discussion it is clear that the expertise of the knowledge engineer is needed in the development of KA systems. The relation between KA systems and knowledge engineers will however will be akin to that between a domain expert and an expert system. The expert system is not intended to replace the domain expert but to act as an intelligent assistant.

### **8.2.3 Problem-Solving Methods and Application Tasks**

The mapping between heuristic classification and analysis task is not a robust one. One of the reasons is that the task types are described at a very general level. For instance, there is an epistemological difference between medical diagnosis and electronic diagnosis (Clancey, 1985). Furthermore, Chandrasekaran (1987) has argued that heuristic classification is a generic category made up of many elementary problem-solving methods. More research is needed here to identify the exact relationship between problem-solving methods and application tasks.

## REFERENCES

- Anderson, J. (1987) Skill acquisition: compilation of weak-method problem solutions. Psychological Review, 94, 192-210.
- Anjewierden, A. (1987) Knowledge acquisition tools. A I Communications, 0, 29-38.
- Baker, K. (1986) Archaeology and expert systems: some problems encountered during practical work. Proceedings of the Third International Expert Systems Conference, London, 211-219.
- Barr, A. and Feigenbaum, E. (1981) The Handbook of Artificial Intelligence, Volume 1, Addison-Wesley, Reading, Massachusetts.
- Bennet, J. (1985) ROGET: a knowledge based system for acquiring the conceptual structure of a diagnostic expert system. Journal of Automated Reasoning, 1, 49-74.
- Bennet, J. and Hollander, C. (1981) DART: an expert system for computer fault diagnosis. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, 843-845.
- Berry, D. (1987) The problem of implicit knowledge. Expert Systems, 4, 144-151.
- Bobrow, D., Mittal, S., and Stefik, M. (1986) Expert systems: perils and promise. Communications of the ACM, 29, 880-894.
- Boose, J. H. (1985) A knowledge acquisition program for expert systems based on personal construct psychology. International Journal of Man-Machine Studies, 23, 495-525.
- Boose, J. H. (1988) A survey of knowledge acquisition techniques and tools. Proceedings of the Third Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.

- Boose, J. and Bradshaw, J. (1987) Expertise transfer and complex problems: using AQUINAS as a knowledge acquisition workbench for expert systems. International Journal of Man-Machine Studies, 26, 21-25.
- Brachman, R. (1978) A structural paradigm for representing knowledge. BBN Technical Report, Bolt, Beranek and Newman, Cambridge, Mass.
- Breuker, J. and Wielinga, B. (1985) KADS: structured knowledge acquisition for expert systems. Proceedings of Fifth International Workshop on Expert Systems and their Applications, Avignon, France.
- Breuker, J. and Wielinga, B. (1987) Use of models in the interpretation of verbal data. In Kidd, A. (ed) Knowledge Acquisition for Expert Systems: A practical handbook, Plenum Press, New York, 17-44.
- Buchanan, B., Sutherland, G. and Feigenbaum, E. (1969) Rediscovering some problems of artificial intelligence in the context of organic chemistry. In Meltzer, B. and Michie, D. (eds) Machine Intelligence 5, Edinburgh University Press, Edinburgh, 253-280.
- Buchanan, B. (1979) Issues of representation in conveying the scope and limitations of intelligent assistant programs. In Hayes, J., Michie, D. and Mikulich, L. (eds) Machine Intelligence 9, Ellis Horwood, Chichester, 409-425.
- Buchanan, B. (1982) New research on expert systems. In Hayes, J., Michie, D. and Pao, Y-H. (eds) Machine Intelligence 10, Willey and Sons, New York, 269-299.
- Buchanan, B. and Shortliffe, E. (eds) (1984) Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading, MA.
- Burton, A. and Shadbolt, N. (1987) Knowledge engineering. TR 87-2-1, Department of Psychology, University of Nottingham, Nottingham, UK.



- Burton, A., Shadbolt, N., Hedgecock, A. and Rugg, G. (1987) A formal evaluation of knowledge elicitation techniques for expert systems: domain 1. In Moralee, S. (eds) Research and development in expert systems IV, Cambridge University Press, Cambridge, UK.
- Burton, A., Shadbolt, N., Rugg, G. and Hedgecock, A. (1988) Knowledge elicitation techniques in classification domains. Proceedings of the Eighth European Conference on Artificial Intelligence, 85-90.
- Bylander, T. and Chandrasekaran, B. (1987) Generic tasks for knowledge-based reasoning: the "right" level of abstraction of Knowledge acquisition. International Journal of Man-Machine Studies, 26, 231-243.
- Chandrasekaran, B. (1987) Towards a functional architecture for intelligence based on generic information processing tasks. Proceedings of the Tenth International Joint Conference of Artificial Intelligence, 1183-1192.
- Chase, W. and Simon, H. (1973) The mind's eye in chess. In Chase, W. (ed.) Visual Information Processing, Academic Press, New York, 215-282.
- Clancey, W. (1985) Heuristic classification. Artificial Intelligence, 27, 289-350.
- Clancey, W. (1989) Viewing knowledge bases as qualitative models. IEEE Expert, 9-23.
- Collins, H., Green, R., and Draper, R. (1985) Where's the expertise? Expert Systems as a medium of knowledge transfer. In Merry, M. (ed.) Expert Systems 85, Cambridge University Press, Cambridge.
- Davis, R. (1979) Interactive transfer of expertise: acquisition of new inference rules. Artificial Intelligence, 12, 121-157.
- Demaid, A. and Zucker, J. (1988) A conceptual model for materials selection. Materials Selection, 4, 291-297.

- Duda, R. and Gaschnig, J. (1981) Knowledge-based expert systems coming of age. BYTE, 6, 238-281.
- Engelmore, R. and Allan, T. (1979) Structure and function of the crysalis system. Proceedings of the Sixth International Joint Conference of Artificial Intelligence,
- Ericsson, K. and Simon, H. (1980) Verbal reports as data, Psychological Reviews, 87, 215-251.
- Eshelman, L. and McDermott, J. (1986) MOLE: a knowledge acquisition tool that uses its head. Proceedings of the Fifth National Conference on Artificial Intelligence, 950-955.
- Feigenbaum, E. (1977) The art of artificial intelligence: I. themes and case studies of knowledge engineering. Proceedings of the Fifth International Joint Conference of Artificial Intelligence, 1014-1029.
- Feigenbaum, E. and McCorduck, P. (1983) The fifth generation, Pan Books, London.
- Fitts, P. (1964) Perceptual-motor skill learning. In Melton, A. (ed.) Categories of Human Learning, Academic Press, New York.
- Forsythe, D. and Buchanan, B. (1988) Knowledge elicitation in theory and practice: an integrated program of research and training. Proceedings of the Third Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.
- Gaines, B. (1987) An overview of knowledge-acquisition and transfer. International Journal of Man-Machine Studies, 26, 453-472.
- Gammack, J. and Young, R. (1985) Psychological techniques for eliciting expert knowledge. In Bramer, M. (ed.) Research and development in expert systems, Cambridge University Press, Cambridge, UK.
- Gardin, J-C. (1980) Archaeological constructs. Cambridge University Press, Cambridge, UK.

- Godden, D. and Baddeley, A. (1975) Context dependency in two natural environments: on land and underwater. British Journal of Psychology, 66, 325-331.
- Grover, M. (1983) A pragmatic knowledge acquisition methodology. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 436-438.
- Gruber, T. and Cohen, P. (1987) Design for acquisition: principles of knowledge-based design to facilitate knowledge acquisition. International Journal of Man-Machine Studies, 26, 143-159.
- Hart, A. (1986) Knowledge acquisition for expert systems, Kogan Page, London.
- Hayes-Roth, F., Waterman, D. and Lenat, D. (eds). (1983) Building expert systems, Addison-Wesley, Reading, Massachusetts.
- Hill, J. (1970) Prehistoric social organization in the american southwest: theory and method. In Longacre, W. (ed.) Reconstructing prehistoric pueblo societies, University of New Mexico Press, Albuquerque.
- Johnson, P. (1983) What kind of expert should a system be? The Journal of Medicine and Philosophy, 8, 77-97.
- Kitto, C. (1988) Progress in automated knowledge acquisition tools: how close are we to replacing the knowledge engineer? Proceedings of the Third Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.
- Klinker, G., Bentolila, J., Genetet, S., Grimes, M., and McDermott, J. (1987) KNACK - report-driven knowledge acquisition. International Journal of Man-Machine Studies, 26, 65-79.
- Knuth, D. and Moore, R. (1975) An analysis of alpha-beta pruning. Artificial Intelligence, 6, .
- Kolodner, J. and Simpson, R. (1986) Problem solving and dynamic

- memory. In J. Kolodner and C. Riesbeck (eds) Experience, memory, and reasoning, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 99-126.
- LaFrance, M. (1987) The knowledge acquisition grid: a method for training knowledge engineers. International Journal of Man-Machine Studies, 26, 245-255.
- Lenat, D. and Feigenbaum, E. (1987) On the thresholds of knowledge. Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 1173-1182.
- Marcus, S., McDermott, J., and Wang, T. (1985) Knowledge acquisition for constructive systems. Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 637-639.
- McDermott, J. (1982) R1: a rule-based configurer of computer systems. Artificial Intelligence, 19, 39-88.
- McDermott, J. (1988) - Preliminary steps toward a taxonomy of problem-solving methods. In Marcus, S. (ed.) Automatic knowledge acquisition for expert systems, Kluwer Academic Publishers, Boston.
- Michalski, R. (1983) A theory and methodology of inductive learning. Artificial Intelligence, 20, 111-161.
- Michalski, R. and Chilausky, R. (1980) Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology. International Journal of Man-Machine Studies, 12, 63-87.
- Michie, D. (1986) The superarticulacy phenomenon in the context of software manufacture. In Michie, D. and Bratko, I. Expert systems: automating knowledge acquisition, Addison-Wesley, Wokingham, 21-44.
- Minsky, M. (1975) A framework for representing knowledge. In Winston, P. (ed.) The psychology of computer vision, McGraw-Hill, New York.

- Motta, E., Rajan, T., and Eisenstadt, M. (1989a) A methodology and tool for knowledge acquisition in KEATS-2. In Guida, G. and Tasso, C (eds.) Topics in expert system design: methodologies and tools, North-Holland, Amsterdam, 297-322.
- Motta, E., Rajan, T., and Eisenstadt, M. (1989b) Knowledge acquisition as a process of model refinement. International Journal of Man-Machine Studies. (Also available as: HCRL Technical Report no. 40, The Open University, Milton Keynes, UK.)
- Musen, M. (1988) An editor for the conceptual models of interactive knowledge acquisition tools. Proceedings of the Third Knowledge Acquisition for Knowledge Based Systems Workshop, Banff, Canada.
- Neale, I. (1988) First generation expert systems: a review of knowledge acquisition methodologies. Knowledge Engineering Review, 3, 105-145.
- Newell, A. (1982) The knowledge level. Artificial Intelligence, 18, 87-127.
- Newell, A. and Simon, H. (1972) Human problem solving, Prentice-Hall, Engelwood Cliffs, NJ.
- Nisbett, R. and Wilson, T. (1977) Telling more than we can know: verbal reports on mental processes. Psychological Review, 84, 231-259.
- Patel, J. (1988) ASKE: towards an automated knowledge acquisition system. HCRL Technical Report No. 36, The Open University, Milton Keynes.
- Patel, J. (1989a) ASKE: towards automated knowledge acquisition. Proceedings of the Third European Knowledge Acquisition for Knowledge Based Systems Workshop, Paris.
- Patel, J. (1989b) On the road to automatic knowledge engineering. Proceedings of the Eleventh Joint Conference on Artificial Intelligence, Detroit.
- Patel, J. and Stutt, A. (1989a) Reducing uncertainty in archaeological

interpretation using expectation-based reasoning. In Kelly, B. and Rector, A. (eds.) Research and development in expert systems V, Cambridge University Press, Cambridge.

Patel, J. and Stutt, A. (1989b) Beyond classification: the use of artificial intelligence techniques for the interpretation of archaeological data. Proceedings of Computer Applications in Archaeology, University of York.

Patil, R., Szolovits, P. and Schwartz, W. (1981) Causal understanding of patient illness in medical diagnosis. Proceedings of the Seventh International Joint Conference of Artificial Intelligence, 893-845.

Pople, H. (1982) Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnostics. In Szolovits, P. (ed.) Artificial Intelligence in Medicine, Westview Press, Boulder, USA, 119-190.

Pople, H. (1985) Coming to grips with the multiple-diagnosis problem. In Schaffner, K. (ed.) Logic of discovery and diagnosis in medicine, University of California press, Berkeley, USA, 181-198.

Quinlan, J. (1982) Semi-autonomous acquisition of pattern-based knowledge. In Michie, D. (ed.) Introductory readings in expert systems, Gordon and Breach, New York, 192-207.

Reichgelt, H. and van Harmelen, F. (1986) Criteria for choosing representation languages and control regimes for expert systems. Knowledge Engineering Review, 1, 2-17.

Schank, R. (1982) Dynamic memory. Cambridge University Press, Cambridge.

Schank, R. and Abelson, R. (1977) Scripts, plans, goals, and understanding: an inquiry into human knowledge structures. Lawrence Erlbaum Associates, Hillsdale, N.J.

Shadbolt, N. and Burton, M. (1989) Knowledge elicitation. In Wilson, J.

and Corlett, N. (eds) Evaluation of human work: practical ergonomics methodology, Taylor and France.

Shaw, M. and Gaines, B. (1987) Advances in interactive knowledge engineering. In Bramer, M. (ed.) Research and development in expert systems III, Cambridge University Press, Cambridge, UK, 111-122.

Simon, H. (1985) Artificial-intelligence approaches to problem solving and clinical diagnosis. In Schaffner, K. (ed.) Logic of discovery and diagnosis in medicine, University of California press, Berkeley, USA, 72-93.

Smith, R. and Baker, J. (1983) The dipmeter advisor system. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 122-29.

Sørensen, T. and Nordhus, O. (1987) LOKE: a drill bit selection system. In Bramer, M. (ed.) Research and development in expert systems IV, Cambridge University Press, Cambridge, UK, 32-41.

Swindells, N. and Swindells, R. (1985) System for engineering materials selection. Materials Selection, 1, 301-304.

Vesonder, G., Stolfo, S., Zielinski, J., Miller, F. and Copp, D. (1983) ACE: an expert system for telephone cable maintenance. Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 116-121.

Waterman, D. (1986) A guide to expert systems, Addison-Wesley, Reading, MA.

Welbank, M. (1983) A review of knowledge acquisition techniques for expert systems. Martlesham Consultancy Services, Ipswich.

Weiss, s. and Kulikowski, C. (1984) A practical guide to designing expert systems, Rowman and Allanheld, New Jersey.

Wielinga, B. and Breuker, J. (1986) Models of expertise. Proceedings of the

Seventh European Conference on Artificial Intelligence, Brighton, UK.

Winograd, T. (1975) Frame representations and the declarative/procedural controversy. In Bobrow, D. and Collins, A. (eds) Representation and Understanding: Studies in Cognitive Science, Academic Press, New York, 185-210.

Woodward, B. (1989) Integrating task demands and problem-solving methods to develop useful taxonomies for knowledge acquisition. Proceedings of the Third European Knowledge Acquisition for Knowledge Based Systems Workshop, Paris.



## APPENDIX A

The following rules were generated automatically by ASKE.

Rules for Knowledge Base : SETTLEMENT-SITE

Author : Jitu

Created : 13-10-89

---

### Temp. rule 6

IF

The use of PIT is (STORAGE REFUSE POSTHOLE)

THEN

activities is BUTCHERING

### Temp. rule 7

IF

The use of METAL is (WEAPON TOOL ORNAMENT)

The material of METAL is (IRON BRONZE GOLD)

THEN

activities is BUTCHERING

### Temp. rule 8

IF

The size of FLOOR is (SMALL MEDIUM LARGE)

The content of FLOOR is FEATURES

THEN

activities is BUTCHERING

### Temp. rule 9

IF

The size of FLOOR is (SMALL MEDIUM LARGE)

The content of FLOOR is FEATURES

THEN

activities if FOOD.PREPARATION

### Temp. rule 10

IF

features is HEARTH

THEN

activities is FOOD.PREPARATION

Temp. rule 11

IF

The surface of POTTERY is (BLACKENED NOT.BLACKENED)

The size of POTTERY is (SMALL MEDIUM LARGE)

The shape of POTTERY is (CIRCULAR CYLINDRICAL OBLONG)

The fabric of POTTERY is (FINE COARSE)

The decoration of POTTERY is

(PLAIN NATURE.DRAWING OTHER.MARKINGS)

THEN

activities is FOOD.PREPARATION

Temp. rule 12

IF

The use of METAL is (WEAPON TOOL ORNAMENT)

The material of METAL is (IRON BRONZE GOLD)

THEN

activities is FOOD.PREPARATION

Temp. rule 13

IF

The edge of STONE is (SHARP SERRATED BLUNT)

The material of STONE is (CHIPPED GROUND)

THEN

activities is FOOD.PREPARATION

Temp. rule 14

IF

The use of PIT is (STORAGE REFUSE POSTHOLE)

THEN

activities is FOOD.PREPARATION

Temp. rule 15

IF

The size of FLOOR is (SMALL MEDIUM LARGE)

The content of FLOOR is FEATURES

THEN

activities is METAL.PRODUCING

Temp. rule 16

IF

features is HEARTH

THEN

activities is METAL.PRODUCING

Temp. rule 17

IF

The use of METAL is (WEAPON TOOL ORNAMENT)

The material of METAL is (IRON BRONZE GOLD)

THEN

activities is METAL.PRODUCING

Temp. rule 18

IF

The edge of STONE is (SHARP SERRATED BLUNT)

The material is STONE is (CHIPPED GROUND)

THEN

activities is METAL.PRODUCING

Temp. rule 19

IF

The size of FLOOR is (SMALL MEDIUM LARGE)

The content of FLOOR is FEATURES

THEN

activities is POTTERY.MAKING

Temp. rule 20

IF

The edge of STONE is (SHARP SERRATED BLUNT)

The material of STONE is (CHIPPED GROUND)

THEN

activities is POTTERY.MAKING

Temp. rule 21

IF

The surface of POTTERY is (BLACKENED NOT BLACKENED)

The size of POTTERY is (SMALL MEDIUM LARGE)

The shape of POTTERY is (CIRCULAR CYLINDRICAL OBLONG)

The fabric of POTTERY is (FINE COARSE)

The decoration of POTTERY is

(PLAIN NATURE.DRAWING OTHER.MARKINGS)

THEN

activities is POTTERY.MAKING

Temp. rule 22

IF

features is HEARTH

THEN

activities is POTTERY.MAKING

Temp. rule 23

IF

The use of PIT is (STORAGE REFUSE POSTHOLE)

THEN

activities is POTTERY.MAKING

Temp. rule 24

IF

The size of FLOOR is (SMALL MEDIUM LARGE)

The content of FLOOR is FEATURES

THEN

activities is STORAGE.AREA

Temp. rule 25

IF

The surface of POTTERY is (BLACKENED NOT.BLACKENED)

The size of POTTERY is (SMALL MEDIUM LARGE)

The shape of POTTERY is (CIRCULAR CYLINDRICAL OBLONG)

The fabric of POTTERY is (FINE COARSE)

The decoration of POTTERY is

(PLAIN NATURE.DRAWINGS OTHER. MARKINGS)

THEN

activities is STORAGE.AREA

Temp. rule 26

IF

The use of PIT is (STORAGE REFUSE POSTHOLE)

THEN

activities is STORAGE.AREA

Temp. rule 27

IF

The use of METAL is (WEAPON TOOL ORNAMENT)

The material of METAL is (IRON BRONZE GOLD)

THEN

site.profile is EXCHANGE.CONTACTS

Temp. rule 28

IF

The surface of POTTERY is (BLACKENED NOT.BLACKENED)

The size of POTTERY is (SMALL MEDIUM LARGE)

The shape of POTTERY is (CIRCULAR CYLINDRICAL OBLONG)

The fabric of POTTERY is (FINE COARSE)

The decoration of POTTERY is

(PLAIN NATURE.DRAWINGS OTHER.MARKINGS)

THEN

site.profile is EXCHANGE.CONTACTS.

Temp. rule 29

IF

The use of METAL is (WEAPON TOOL ORNAMENT)

The material of METAL is (IRON BRONZE GOLD)

THEN

site.profile is SOCIAL.STATUS

Temp. rule 30

IF

activities is BUTCHERING

THEN

The period of OCCUPANCY is (PERMANENT SEASONAL)

Temp. rule 31

IF

activities is METAL.PRODUCING

THEN

The period of OCCUPANCY is (PERMANENT SEASONAL)

Temp. rule 32

IF

activities is POTTERY.MAKING

THEN

The period of OCCUPANCY is (PERMANENT SEASONAL)

Temp. rule 33

IF

The use of DITCH is (DRAINAGE BOUNDARY)

The size of DITCH is (SITE.PERIMETER AREA.PERIMETER)

The shape of DITCH is (CIRCULAR ELONGATED)

THEN

The period of OCCUPANCY is (PERMANENT SEASONAL)

## APPENDIX B

The final output of the session.

Rules for Knowledge Base : SETTLEMENT-SITE

Author : Jitu

Created : 13-10-89

---

### Temp. rule 27

IF

The use of METAL is ORNAMENT

The material of METAL is GOLD

THEN

site.profile is EXCHANGE.CONTACTS

### Temp. rule 28

IF

The surface of POTTERY is NOT.BLACKENED

The fabric of POTTERY is FINE

The decoration of POTTERY is NATURE.DRAWINGS

THEN

site.profile is EXCHANGE.CONTACTS

### Temp. rule 29

IF

The use of METAL is (WEAPON ORNAMENT)

The material of METAL is (BRONZE GOLD)

THEN

site.profile is SOCIAL.STATUS

### Temp. rule 30

IF

activities is BUTCHERING

THEN

The period of OCCUPANCY is SEASONAL

Temp. rule 34

IF

The content of FLOOR is (METAL PIT)  
The size of FLOOR is MEDIUM  
The use of METAL is TOOL  
The material of METAL is IRON  
The use of PIT is REFUSE

THEN

activities is BUTCHERING

Temp. rule 36

IF

The content of FLOOR is (POTTERY METAL HEARTH)  
The size of FLOOR is LARGE  
The surface of POTTERY is BLACKENED  
The shape of POTTERY is COARSE  
The decoration of POTTERY is PLAIN  
The use of METAL is IRON  
features is HEARTH

THEN

activities is FOOD.PREPARATION

Temp. rule 37

IF

The content of FLOOR is (PIT STONE)  
The size of FLOOR is MEDIUM  
The use of PIT is STORAGE  
The edge of STONE is BLUNT  
The material of STONE is GROUND

THEN

activities is FOOD.PREPARATION.

Temp. rule 38

IF

The content of FLOOR is (HEARTH METAL STONE)  
The size of FLOOR is LARGE  
features is HEARTH  
The material of METAL is (IRON BRONZE)  
The edge of STONE is BLUNT  
The material of STONE is GROUND

THEN

activities is METAL.PRODUCING

Temp. rule 41

IF

activities is METAL.PRODUCING  
activities is POTTERY.MAKING  
The use of DITCH is BOUNDARY  
The size of DITCH is SITE.PERIMETER

THEN

The period of OCCUPANCY is PERMANENT

Temp. rule 43

IF

The content of FLOOR is (STONE POTTERY)  
The size of FLOOR is MEDIUM  
The edge of STONE is GROUND  
The surface of POTTERY is BLACKENED

THEN

activities is POTTERY.MAKING

Temp. rule 47

IF

The content of FLOOR is (HEARTH PIT POTTERY)  
The size of FLOOR is LARGE  
features is HEARTH  
The use of PIT is REFUSE  
The surface of POTTERY is BLACKENED

THEN

activities is POTTERY.MAKING

Temp. rule 48

IF

The content of FLOOR is (POTTERY PIT)  
The size of FLOOR is SMALL  
The size of POTTERY is LARGE  
The fabric of POTTERY is COARSE  
The decoration of POTTERY is PLAIN  
The use of PIT is STORAGE

THEN

activities is STORAGE.AREA



## APPENDIX C

A daily chore for the nursing profession is diagnosing and treating patient problems. One of such problems is pressure sores. Identifying at-risk pressure sore patients and drawing-up effective care plans to prevent the occurrence of the problem requires expertise. The following session develops an initial knowledge-base for diagnosing pressure sores.

### Stage One: Task Characterization

- **Domain: Nursing**  
Nursing is a new domain hence ASKE creates a GTEMP for it.
- **Task-type: Diagnosis**  
The ATEMP for diagnosis is selected.
- **Specialist area: Pressure Sores**  
A WTEMP for Pressure Sores is created.
- **Project Goals**  
The goal of the new application is to diagnose and draw-up care plans for pressure sore patients. The system will be used by nurses.
- **RTEMP: Foot Problems (Medical diagnosis)**  
The user selects the RTEMP to use from the set of diagnostic systems that ASKE knows about.

NEW LOGO SAVE QUIT HELP  
KB KB KB KB

RULES/CR  
BY  
THEN

ASKE HCRL

---

MANAGED CASE  
ASKED  
ASKED DATA  
SYSTEM KEYS

MANUAL BOOK

---

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

---

MANAGED CASE BY THEN

---

CENTRAL CONCEPTS  
NURSING A  
TEMP.RULES

---

Purpose: Identify domain

Comment:

The domain is the general area of the expertise, for example, archaeology, medicine.

e.

[For further help on answering questions, type (CRD).]

---

Enter the name of the domain (choose one of the following):

- 1 STRATEGIC PLANNING
- 2 MEDICINE
- 3 ARCHAEOLOGY
- 4 OTHER

(-) 4

Enter the domain name

<word> nursing.

IF #MORE#

aske>

The domain of the new application is Nursing. As a GTEMP for nursing does not exist, a new one is created.

NEW  
KB

LOAD  
KB

SAVE  
KB

QUIT  
KB

HELP  
KB

RULES  
KB

ASKE  
KB

HCRL

**Purpose:** Identify task type.

**Comment:**

The task type (or problem type) is the kind of problem the expert usually solves. This can be, for example, diagnostic, repair, design, planning. ASKE is, however, suitable for solving analysis tasks only. The analysis tasks are those which can be solved by the classification problem solving method. Thus, for this category of tasks, the solutions can be pre-enumerated.

---

Enter the task type (choose one of the following):

- 1 INTERPRETATION
- 2 DIAGNOSIS
- 3 DEBUGGING
- 4 SELECTION

⇒ 2

MEMORIES

---

task:

MANUALS  
ASKED  
SYSTEM KEYS

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

CENTRAL CONCEPTS

DIAGNOSIS

NURSING

TEMP.RULES

The new application will be performing the task of diagnosis. The ATEMP for diagnosis is selected.

<p>NEW LOAD SAVE QUIT HELP          KB KB KB KB</p>	<p>PULLENKER          OF          1749</p>	<p>ASKED          ASKED DATA          System KEYS</p>	<p>HCRL  <b>ASKED</b></p>	<p>NOTEBOOK</p>	<p>GENERAL TEMPLATES          ACQUISITION TEMPLATE          REFERENCE TEMPLATE          WORKING TEMPLATE          CENTRAL CONCEPTS          RULES</p>
<p>Purpose: identify specialist area.</p> <p>Comment:          The specialist area is the actual area of expertise. In the domain of medicine, for example, it will be: foot problems, aids infections, lung diseases.</p> <p>Enter your specialist area          (word) pressure.sores</p>					
<p>*****          #NNORIE**          *****</p>					
<p>asked&gt;</p>					
<p>CENTRAL CONCEPTS          DIAGNOSIS          NURSING          PRESSURE.SORES          TEMP.RULES</p>					

The area of specialization is Pressure Sores. A new WTEMP, called pressure.sores, is created.

NEW KEY	LOAD KB	SAVE KB	QUIT HELP	RULENIFIER IF THEN	ASKE HCRL
PRESSURE-SORES ASKEDATA System KEYS					
GENERAL TEMPLATE ACQUISITION TEMPLATE REFERENCE TEMPLATE WORKING TEMPLATE CENTRAL CONCEPTS RULES					
CENTRAL CONCEPTS DIAGNOSIS NURSING PRESSURE-SORES TEMP-RULES					
** Who will be using the system? <anything> Nurses **					
** What are the project aims? <anything> Make pressure sore diagnostic knowledge available to nurses. **					
** Give a description of the task. <anything> Diagnose and draw-up care plans for pressure sore patients. **					
** Information about project and users will be used for reference purposes. [Type NIL, if answer is unknown. For further help on answering questions, type <GR>.] **					
** Purpose: Identify project aims Comment: **					
** MEMORIES* **					

The main project goals are identified.

<p>NEW LOAD SAVE QUIT HELP          KB KE KB</p> <p>Purpose: Select an RTEMP</p> <p>Comments: The RTEMP will be used to guide the development of the initial task model.</p>	<p>ASKER HCRL</p> <p>GENERAL TEMPLATE</p> <p>ACQUISITION TEMPLATE</p> <p>REFERENCE TEMPLATE</p> <p>WORKING TEMPLATE</p> <p>CENTRAL CONCEPTS</p> <p>RULES</p>
<p>Pick an RTEMP (choose one of the following):</p> <ul style="list-style-type: none"> <li>1 (AIDS.INFECTION MEDICINE)</li> <li>2 (FOOT.PROBLEMS MEDICINE)</li> </ul> <p>=&gt; 2</p> <p>We will be using the FOOT.PROBLEMS, which is a knowledge-base in the domain of MEDICINE, as an exemplar. FOOT.PROBLEMS is designed to do the task of DIAGNOSIS; its task description is: diagnose foot problems from symptoms.</p>	<p>CENTRAL CONCEPTS</p> <p>DIAGNOSIS</p> <p>NURSING</p> <p>PRESSURE-SORIES</p> <p>TEMP.RULES</p>
<p>***RORIE**</p> <p>asker&gt;</p>	

The diagnosing of Foot problems from the domain of medicine is selected as the RTEMP for this application.

### Stage Two: Task Modeling

- What knowledge is required for a diagnostic system.  
The user is presented with information about the classification model for diagnostic systems.
- An example is shown.  
The selected RTEMP is presented as an exemplar.
- Obtaining the main data categories.  
The user is prompted for the main concept categories of data for the diagnosis of pressure sores.
- Obtaining the main solution categories.  
The user is prompted for the main concept categories of solution for the diagnosis of pressure sores.
- The specified concept categories for the new application are presented for any last minute amendments.

NEW KEB LOAD SAVE QUIT HELP  
KEB KEB KEB KEB KEB KEB

RULEMAKER  
IF THEN

ASKED  
ASKED  
ASKED

HCRL

**Purpose:** Identify the main concept categories

**Comments:** For problem solving in analysis tasks, there are basically two categories of concepts: data and solution. At this stage of KA, we are interested in identifying this concept categories.

There may be one or more of each type of concept categories.

\* For DATA categories, type in the name of concept/s which represent the basic data for the problem. The characteristic of data category is that it is either given/observed or obtained by test.

\* For SOLUTION categories, type in the name of concept/s which represent the basic solution for the problem. The characteristic of solution category is that it is inferred from data categories.

---

The task of diagnosis involves inferring system malfunctions from observables.

**Data Types:**

- Signs of Malfunction
- Object

**Solution Types:**

- Causes of Malfunction

##MORE##

**WORKBOOK**

GENERAL TEMPLATE
ACQUISITION TEMPLATE
REFERENCE TEMPLATE
WORKING TEMPLATE
CENTRAL CONCEPTS
RULES

**CENTRAL CONCEPTS**

DIAGNOSIS

NUFSING

PRESSURE..SORES

TEMP..RULES

The user is presented with information on the kind of knowledge required of him/her, at this stage of KA.



<p>NEW LOAD SAVE QUIT HELP          KEF KBF KEB KEB</p> <p>RULEMAKER          OF          THEM</p>	<p>MANAGED BY          PRESSURE-SORES          ASKEDATA          System KEYS</p> <p>ASKE HCRAL</p>
<p>ASKEDATA was used to develop a knowledge base in the domain of MEDICIN          E for doing DIAGNOSIS of FOOT-PROBLEMS.</p> <p>The aim of the application was:          diagnose foot problems from symptoms</p> <p>The DATA types were:          PATIENT ( Person suffering from foot problems. )          SYMPTOMS ( The signs and symptoms of foot problems. )</p> <p>The SOLUTION types were:          PROBLEMS ( The cause of the foot problem. )</p>	<p>GENERAL TEMPLATE</p> <p>ACQUISITION TEMPLATE</p> <p>REFERENCE TEMPLATE</p> <p>WORKING TEMPLATE</p> <p>CENTRAL CONCEPTS</p> <p>RULES</p> <p>CENTRAL CONCEPTS</p> <p>DIAGNOSIS</p> <p>NURSING</p> <p>PRESSURE-SORES</p> <p>TEMP-RULES</p>
<p>HELP</p>	

Next, the RTEMP for diagnosing Foot Problems is presented.

NEW KB    LOAD KB    SAVE KB    QUIT    HELP

FULLMEMBER  
IF  
THEN

ASKED  
System KPs

ASKE  
HCRL

---

What is the main category of data for the DIAGNOSIS of PRESSURE-SORES?

<word> patient.state

Comments for PATIENT.STATE

<anything> The physical, physiological and psychological condition of the pressure sore patient.

What is the OBJECT of DIAGNOSIS? [Type nil, if none.]

<word> pressura.sore.patient

Comments for PRESSURE.SORE.PATIENT

<anything> Person prone to pressure sores.

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

---

EXPERT: Jitu Patel

DOMAIN: Medicine

TASK: Diagnosis

SPECIALIST AREA: Foot.problems

TASK DESCRIPTION:

Diagnose foot problems from symptoms

Data Types:

PATIENT (the person have foot problem)

SYMPTOMS (sigas and symptoms of foot problems)

Solution Types:

PROBLEMS (The cause of the foot problems.)

---

NEW KB    LOAD KB    SAVE KB    QUIT    HELP

FULLMEMBER  
IF  
THEN

ASKED  
System KPs

ASKE  
HCRL

---

What is the main category of data for the DIAGNOSIS of PRESSURE-SORES?

<word> patient.state

Comments for PATIENT.STATE

<anything> The physical, physiological and psychological condition of the pressure sore patient.

What is the OBJECT of DIAGNOSIS? [Type nil, if none.]

<word> pressura.sore.patient

Comments for PRESSURE.SORE.PATIENT

<anything> Person prone to pressure sores.

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

---

EXPERT: Jitu Patel

DOMAIN: Medicine

TASK: Diagnosis

SPECIALIST AREA: Foot.problems

TASK DESCRIPTION:

Diagnose foot problems from symptoms

Data Types:

PATIENT (the person have foot problem)

SYMPTOMS (sigas and symptoms of foot problems)

Solution Types:

PROBLEMS (The cause of the foot problems.)

First, the main data categories are elicited. The concept category for the data is called Patient.State.

NEW  
KEY

LOAD  
KEY

SAVE  
KEY

QUIT

HELP

RULEMAKER  
IF  
THEN

ASKE  
HCRL

What is the main category of solution when doing DIAGNOSIS of PRES  
SURE-SORES?

<word> care.plans

Comments for CARE.PLANs

<anything> The course of treatment for pressure sore patients.

Enter a category for treating CARE.PLANs [Type nil, if none]:

<word> nil

GENERAL TEMPLATE

ACQUISITION TEMPLATE

REFERENCE TEMPLATE

WORKING TEMPLATE

CENTRAL CONCEPTS

RULES

EXPERT: Jitu panel

DOMAIN: Medicine

TASK: Diagnosis

SPECIALIST AREA: Foot.problems

TASK DESCRIPTION:

Diagnose foot problems from symptoms

Data Types:

PATIENT (the person have foot problem)

SYMPTOMS (signs and symptoms of foot problems)

Solution Types:

PROBLEMS. (The cause of the foot problem.)

Then, the solution categories are obtained. The aim of the new application is to draw up Care.Plans.

NEW LOAD SAVE QUIT HELP  
 KE KE KB

RULER/KEY  
 BY  
 THEN

ASKED HCRL

---

**Purpose:** Identify main concept categories

**Comments:** In the Concept Categories Window is given a tree of main concept categories for the task of NURSING in DIAGNOSIS. The objective objective is to check and, if necessary, amend the categories. The data and solution categories given are:

**Date:** PRESSURE.SCORE.PATIENT PATIENT.STATE

**Solution:** CARE.PLANS

**Instructions:**

Clicking mouse button on a concept would enable the following operations:

- Left Mouse 1: Create new object.
- Middle Mouse 1: Delete/Rename object.
- Right Mouse 1: Flush window.

Clicking anywhere else in the window would enable the following operations:

- Left Mouse 1: Create new object.
- Right Mouse 1: Flush window (quit).

[For further help on identifying concept categories in your task look under Input for Concept Categories in HELP menu]

ASKED  
 PRESSURE.SCORE.PATIENT  
 SYSTEM KEYS

CARE.PLANS  
 PATIENT.STATE  
 PRESSURE.SCORE.PATIENT

A

---

ASKED

The main concept categories obtained previously are presented for final amendments.

### Stage Three: Knowledge Elicitation

- Obtain domain concepts.

The Sketch-Pad facility is provided for entering new concepts and their attributes. All domain concepts are displayed in the Central Concepts Window, which also has facilities for entering and editing concepts and attributes.

- Identifying relationships between concepts.

The Relations Window displays the 'supports' (top) and 'inferred-from' (bottom) relationships of any given concept with other concepts. The window also provides facility for defining new associations. If this stage is quit before all concept relationships have been identified, the user is informed about the unaccounted concepts.

NEW LOAD SAVE QUIT HELP  
 KB KB KB

ASKER  
 THEN

HCRL

Purpose: Identify domain concepts.

Comments:

ALL concepts are first entered in the SKETCH PAD window. These concepts will subsequently be transferred to the CENTRAL CONCEPTS window.

----- Instructions -----

[1] Enter new concepts by typing the concept name followed by <CR> in the bottom right window. This concept will be displayed in the SKETCH PAD window.

[2] The concept can be moved to the CENTRAL CONCEPTS window by using mouse clicks (defined in the mouse documentation window).

-----

[For further help on identifying concept categories in your task book under Input for Domain Concepts in HELP menu]

ASKER

Enter concept: Incontinence

```

            graph TD
                A[CENTRAL...CONCEPTS] --- B[CARE PLANS]
                A --- C[PATIENT STATE]
                A --- D[PRESSURE SCORE PATIENT]
                B --- E[MEDICATION]
                B --- F[NUTRITION]
                C --- G[IMMOBILITY]
                C --- H[LOW ACTIVITY]
                C --- I[MALNUTRITION]
                C --- J[SKIN CONDITION]
            
```

ASKEDATA  
 System KEYS

The user is asked to enter concepts (and their attributes) that are in the identified concept categories.

NEW LOAD SAVE QUIT HELP  
 KB KB KB KB

ASKE  
 HCRL

Purpose: Identify domain concepts.

Comments:

ALL concepts are first entered in the SKETCH PAD window. These concepts will subsequently be transferred to the CENTRAL CONCEPTS window.

----- Instructions -----

[1] Enter new concepts by typing the concept name followed by <CR>, in the bottom right window. The concept will be displayed in the SKETCH PAD window.

[2] The concept can be moved to the CENTRAL CONCEPTS window by using mouse clicks (defined in the mouse documentation window).

-----

[For further help on identifying concept categories in your task look under Input for Domain Concepts in HELP menu]

AGE  
 MEDICAL CONDITION  
 MEDICAL HISTORY

Enter new possible value type: (choose one of the following):

- 1 NUMERICAL-VALUE
- 2 LIST-OF-VALUES
- 3 UNKNOWN

-> 2

Enter possible values:  
 <list-of-words> pain hypoxia no illness

MANAGEMENT PLAN  
 PRESSURE SCORES  
 ASKEDATA  
 System KEs

Enter concept:

ASKE  
 HCRL

Diagram labels:

- MEDICATION
- MOBILIZATION
- NUTRITION
- PHYSIOTHERAPY
- IMPACTILITY
- INCONTINENCE
- LOW ACTIVITY
- MALNUTRITION
- SKIN CONDITION
- PATIENT STATE
- RESURVE SCORE PATIENT

When an attribute is entered, the user is asked to specify the possible values it will take.

**NEW** **LOAD** **SAVE** **QUIT** **HELP**

**KB** **KB** **KB** **KB** **KB**

**HELP** **INDEX** **OF** **THE** **CONCEPTS**

**HELP** **INDEX** **OF** **THE** **CONCEPTS**

**HELP** **INDEX** **OF** **THE** **CONCEPTS**

ASKE

HCRL

---

**Purpose:** Identify relationships between concepts.

**Comments:** In the Central Concepts Window is given a tree of main concept categories for the task of NURSING in DIAGNOSIS. The data and solution categories are:

**Data:** PATIENT.STATE PRESSURE SCORE.PATIENT

**Solution:** CARE.PLANS

The objective is to draw relationship between the concepts in the data categories with those in the solution categories.

**Instructions:**

Clicking mouse button on a concept would enable the following operations:

Left 1: Create new object. Left 2: Redisplay object.

Middle 1: Delete/Remove object. Middle 2: Define object relationship.

Right 1: Edit object. Right 2: Toggle font.

Clicking anywhere else in the window would enable the following operations:

Left 1: Redisplay.

Middle 1: Toggle display.

Right 1: Flush window (quit).

---

[For further help on identifying concept categories in your task look under Input for Diagnostics Concepts in HELP menu]

---

aske >

**ASKE**

**HCRL**

Concepts can be entered directly into the Central Concepts Window. Attribute editing facilities are also provided.



NEW LOAD SAVE QUIT HELP  
KB KB KB

ASKE HCRL

**Purpose:** Identify relationships between concepts.

**Comments:** In the Central Concepts Window is given a tree of main concept categories for the task of NURSING in DIAGNOSIS. The data and solution categories are:

Data: PATIENT.STATE PRESSURE.SORE.PATIENT

Solution: CARE.PLANS

The objective is to draw relationships between data categories with those in the solution categories.

----- Instructions -----

Clicking mouse button on a concept would enable:

Left 1: Create new object. Left 2: Middle 1: Delete/Rename object. Middle Right 1: Edit object.

Clicking anywhere else in the window would enable:

Left 1: Redisplay. Left 2: Middle 1: Toggle display. Right 1: Flush window (quit).

[For further help on identifying concept categories for Diagnosis (Concepts in HELP menu)]

Concept: PRESSURE.SORE.PATIENT

Attributes: MEDICAL.CONDITION

Possible values: ((ONE OF PAIN HYPOXIA NO.ILLNESS))

Enter new possible value type: (choose one of the following):

1. NUMERICAL-VALUE
2. LIST-OF-VALUES
3. UNKNOWN

=> 2

Enter possible values:

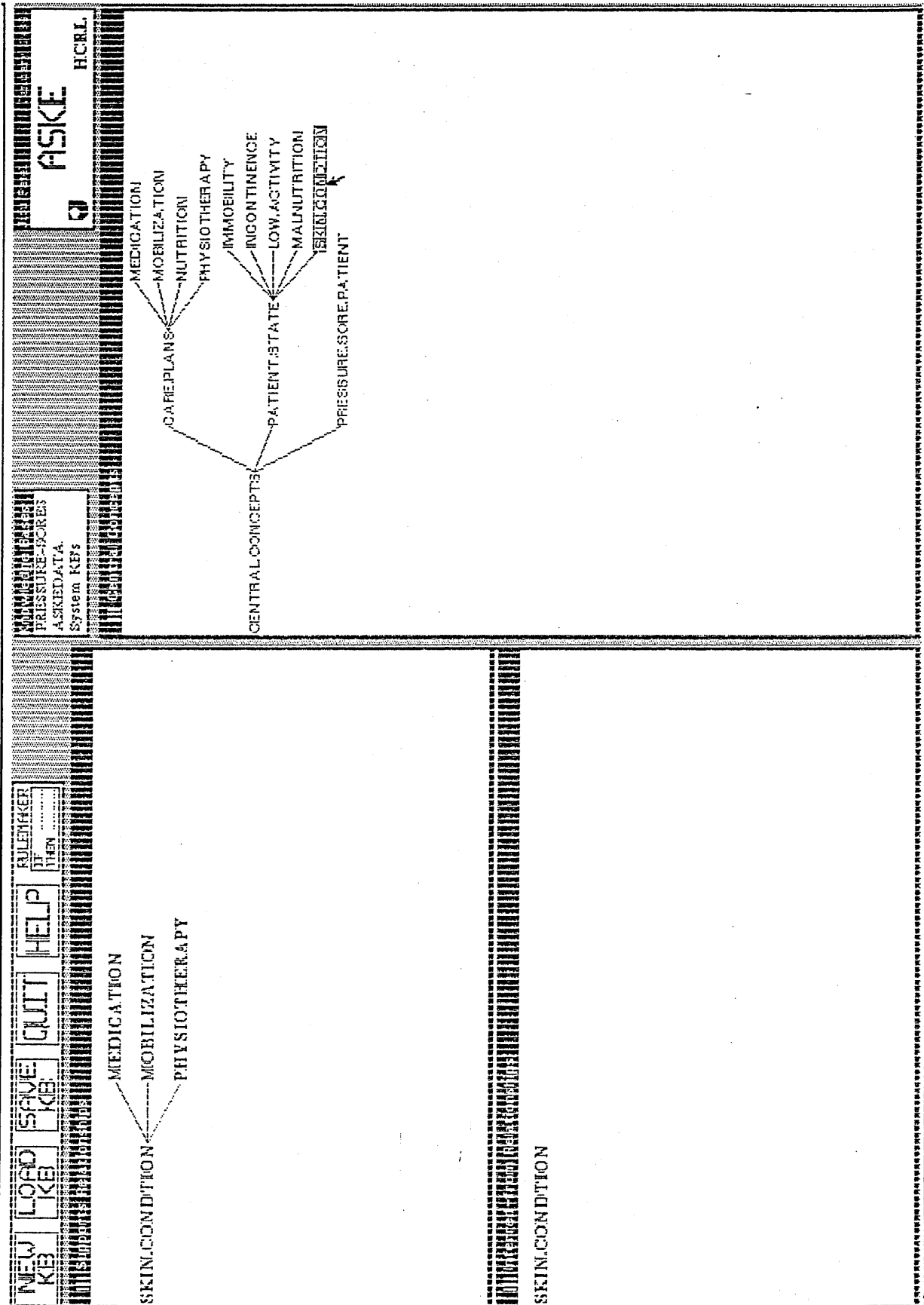
<list-of-words> pain hypoxia toxemia

Diagram illustrating relationships between concepts:

```

    graph TD
      CARE_PLANS[CARE.PLANS] --- MEDICATION[MEDICATION]
      CARE_PLANS --- MOBILIZATION[MOBILIZATION]
      CARE_PLANS --- NUTRITION[NUTRITION]
      CARE_PLANS --- PHYSIO_THERAPY[PHYSIO.THERAPY]
      CARE_PLANS --- IMMOBILITY[IMMOBILITY]
      CARE_PLANS --- INCONTINENCE[INCONTINENCE]
      CARE_PLANS --- LOW_ACTIVITY[LOW.ACTIVITY]
      CARE_PLANS --- MALNUTRITION[MALNUTRITION]
      CARE_PLANS --- SKIN_CONDITION[SKIN.CONDITION]
      CARE_PLANS --- PATIENT_STATE[PATIENT.STATE]
      CARE_PLANS --- PRESSURE_SORE_PATIENT[PRESSURE.SORE.PATIENT]
    
```

The list of possible values for the medical.condition attribute are amended.



The next objective is to identify the relationships between concepts. This is done in the Relations Window.

On the basis of the identified relationships between concepts, ASKE automatically generates rules (also called association rules). The following rules were generated from this session.

Rules for Knowledge Base : PRESSURE-SORES

Author : Jitu

Created : 13-10-89

---

Temp. rule 52

IF

The sex of PRESSURE.SORE.PATIENT is (MALE FEMALE)

The age of PRESSURE.SORE.PATIENT is (<50 50-60 60-70 70-80 >80)

The medical.history of PRESSURE.SORE.PATIENT is  
(SERIOUS ILLNESS NO.ILLNESS)

The medical.condition of PRESSURE.SORE.PATIENT is  
(PAIN HYPOXIA TOXAEMIA)

The medical treatment of PRESSURE.SORE.PATIENT is (DRUGS WOUNDS)

THEN

care.plans is MEDICATION

Temp. rule 53

IF

patient.state is LOW.ACTIVITY

THEN

care.plans is MEDICATION

Temp. rule 54

IF

The reddened.area is SKIN.CONDITION is  
(NORMAL MODERATE CHRONIC)

The breaks.in.skin of SKIN.CONDITION is  
(NORMAL MODERATE CHRONIC)

THEN

care.plans is MEDICATION

Temp. rule 55

IF

The kind of INCONTINENCE is (URINE DOUBLE)

THEN

care.plans is MEDICATION

Temp. rule 56

IF

The reddened.area is SKIN.CONDITION is  
(NORMAL MODERATE CHRONIC)

The breaks.in.skin of SKIN.CONDITION is  
(NORMAL MODERATE CHRONIC)

THEN

care.plans is MOBILIZATION

Temp. rule 57

IF

patient.state is LOW.ACTIVITY

THEN

care.plans is MOBILIZATION

Temp. rule 58

IF

The sex of PRESSURE.SORE.PATIENT is (MALE FEMALE)

The age of PRESSURE.SORE.PATIENT is (<50 50-60 60-70 70-80 >80)

The medical.history of PRESSURE.SORE.PATIENT is  
(SERIOUS ILLNESS NO.ILLNESS)

The medical.condition of PRESSURE.SORE.PATIENT is  
(PAIN HYPOXIA TOXAEMIA)

The medical treatment of PRESSURE.SORE.PATIENT is (DRUGS WOUNDS)

THEN

care.plans is MOBILIZATION

Temp. rule 59

IF

The kind of MALNUTRITION is  
(MALNUTRITION DEHYDRATION WEIGHT.ABNORMALITY)

THEN

care.plans is NUTRITION

Temp. rule 60

IF

The kind of INCONTINENCE is (URINE DOUBLE)

THEN

care.plans is NUTRITION

Temp. rule 61

IF

The sex of PRESSURE.SORE.PATIENT is (MALE FEMALE)

The age of PRESSURE.SORE.PATIENT is (&lt;50 50-60 60-70 70-80 &gt;80)

The medical.history of PRESSURE.SORE.PATIENT is  
(SERIOUS ILLNESS NO.ILLNESS)The medical.condition of PRESSURE.SORE.PATIENT is  
(PAIN HYPOXIA TOXAEMIA)

The medical treatment of PRESSURE.SORE.PATIENT is (DRUGS WOUNDS)

THEN

care.plans is NUTRITION

Temp. rule 62

IF

The reddened.area is SKIN.CONDITION is  
(NORMAL MODERATE CHRONIC)The breaks.in.skin of SKIN.CONDITION is  
(NORMAL MODERATE CHRONIC)

THEN

care.plans is PHYSIOTHERAPY

Temp. rule 63

IF

The sex of PRESSURE.SORE.PATIENT is (MALE FEMALE)

The age of PRESSURE.SORE.PATIENT is (&lt;50 50-60 60-70 70-80 &gt;80)

The medical.history of PRESSURE.SORE.PATIENT is  
(SERIOUS ILLNESS NO.ILLNESS)The medical.condition of PRESSURE.SORE.PATIENT is  
(PAIN HYPOXIA TOXAEMIA)

The medical treatment of PRESSURE.SORE.PATIENT is (DRUGS WOUNDS)

THEN

care.plans is PHYSIOTHERAPY

Temp. rule 64

IF

The source of IMMOBILITY is (PARALYSIS COMA EQUIPMENT/SPLINTAGE)

THEN

care.plans is PHYSIOTHERAPY

Temp. rule 65

IF

The kind of MALNUTRITION is  
(MALNUTRITION DEHYDRATION WEIGHT.ABNORMALITY)

THEN

care.plans is PHYSIOTHERAPY

### Stage Four: Rules Editing

ASKE generated association rules are displayed in the Rulemaker Interface and the user is invited to edit them. Editing of rules involves:

- Merging rules.  
Two or more association rules are merged into a single complex rule.
- Editing premise and conclusion of the rule.  
The premise and conclusion of a rule can be edited in the Rule Editing Window, which offers Zmacs editing facility.

CONTEXT

CLASS

RULE

QUIT

HSKIE

PRESSURE-SORES  
ASKED-DATA  
System XIB's

RULEMAKER  
FCRL

Temp rule 69 -----

IF

- (pressure.sore.patient)
- (sex (male female))
- (age (<50 50-60 60-70 70-80 >80))
- (medical.history (serious.illness no.illness))
- (medical.condition (pain hypoxia toxemia))
- (medical.treatment (drugs wounds))
- (skin.condition)
- (reddened.area (normal moderate chronic))
- (breaks.in.skin (normal moderate chronic))

THEN

- (physiotherapy)

END (PRESSURE-SORE-PATIENT)

(MEDICAL-HISTORY NO-ILLNESS)

((MEDICAL-TREATMENT WOUNDS))

((SKIN-CONDITION))

(REDDENED-AREA MODERATE)

(BREAKS.IN.SKIN (NORMAL MODERATE))

PRETISE (TYPE END TO EXIT)

The Rulemaker Interface provides facilities for editing the rules.

At the end of the present session the following edited rules are output.

Rules for Knowledge Base : PRESSURE-SORES

Author : Jitu

Created : 13-10-89

---

Temp. rule 69

IF

The medical.history of PRESSURE.SORE.PATIENT is NO.ILLNESS

The medical.history of PRESSURE.SORE.PATIENT is WOUNDS

The reddened.area of SKIN.CONDITION is MODERATE

The breaks.in.skin of SKIN.CONDITION is (NORMAL MODERATE)

THEN

care.plans is PHYSIOTHERAPY

Temp. rule 70

IF

The age of PRESSURE.SORE.PATIENT is <50

The medical.history of PRESSURE.SORE.PATIENT is NO.ILLNESS

The medical.condition of PRESSURE.SORE.PATIENT is PAIN

The source of IMMOBILITY is EQUIPMENT/SPLINTAGE

THEN

care.plans is PHYSIOTHERAPY

Temp. rule 71

IF

The age of PRESSURE.SORE.PATIENT is <50

The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS

The kind of MALNUTRITION is WEIGHT.ABNORMALITY

THEN

care.plans is PHYSIOTHERAPY

Temp. rule 73

IF

The medical.history of PRESSURE.SORE.PATIENT is NO.ILLNESS

The medical.condition of PRESSURE.SORE.PATIENT is  
(HYPOXIA TOXAEMIA)

The kind of MALNUTRITION is MALNUTRITION

THEN

care.plans is NUTRITION



Temp. rule 74

IF

The medical.history of PRESSURE.SORE.PATIENT is SERIOUS.ILLNESS

The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS

The kind of INCONTINENCE is (URINE DOUBLE)

THEN

care.plans is NUTRITION

Temp. rule 76

IF

The age of PRESSURE.SORE.PATIENT is (&lt;50 50-60)

The medical.history of PRESSURE.SORE.PATIENT is SERIOUS ILLNESS

The medical.condition of PRESSURE.SORE.PATIENT is  
(HYPOXIA TOXAEMIA)

The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS

The reddened.area of SKIN.CONDITION is NORMAL

The breaks.in.skin of SKIN.CONDITION is NORMAL

THEN

care.plans is MOBILIZATION

Temp. rule 77

IF

The age of PRESSURE.SORE.PATIENT is &lt;50

The medical.history of PRESSURE.SORE.PATIENT is SERIOUS.ILLNESS

The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS

patient.state is LOW.ACTIVITY

THEN

care.plans is MOBILIZATION

Temp. rule 80

IF

The age of PRESSURE.SORE.PATIENT is &lt;50

The medical.history of PRESSURE.SORE.PATIENT is SERIOUS.ILLNESS

The medical.condition of PRESSURE.SORE.PATIENT is PAIN

The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS

patient.state is LOW.ACTIVITY

THEN

care.plans is MEDICATION

Temp. rule 81

IF

The medical.condition of PRESSURE.SORE.PATIENT is PAIN  
The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS  
The reddened.area of SKIN.CONDITION is CHRONIC  
The breaks.in.skin of SKIN.CONDITION is (NORMAL MODERATE)

THEN

care.plans is MEDICATION

Temp. rule 82

IF

The medical.history of PRESSURE.SORE.PATIENT is SERIOUS.ILLNESS  
The medical.treatment of PRESSURE.SORE.PATIENT is DRUGS  
The kind of INCONTINENCE is (URINE DOUBLE)

THEN

care.plans is MEDICATION

---

Finally, ASKE creates a new RTEMP for the application. The content of the RTEMP are abstracted from the current knowledge-base.

