DESIGN, DEVELOPMENT, AND TESTING OF AN AUTOMATED KNOWLEDGE-ACQUISITION TOOL TO AID PROBLEM SOLVING, DECISION MAKING, AND PLANNING

by

Timothy G. Kotnour

Master's Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Industrial & Systems Engineering

APPROVED:

Kent E. Williams

Dr. K. E. Williams, Co-Chairman

Dr. H. A. Kurstedt, Co-Chairman

Dr. R. C. Williges

September 14, 1992
Blacksburg, Virginia

Design, Development, and Testing of an Automated Knowledge-Acquisition Tool to Aid Problem Solving, Decision Making, and Planning

by
Timothy G. Kotnour
Kent E. Williams, Ph.D., Co-chairman
Harold A. Kurstedt, Ph.D., Co-chairman

(ABSTRACT)

This research studies the process of acquiring knowledge from experts; that is, studies knowledge-acquisition methods to acquire expert knowledge. Forty subjects used a machine-aided knowledge-acquisition tool to model a word processing task. By using the tool, the subjects developed models that were on average 72.8% accurate with a baseline model of the task and 88.5% consistent among themselves.

This research makes four contributions: 1) a complete review of thirty-one knowledge-acquisition methods from manual to machine learning, 2) an evaluation methodology and metrics to evaluate knowledge-acquisition methods, 3) an evaluation of an automated knowledge-acquisition tool called Cognitive Analysis Tool (CAT) developed for this research, and 4) suggested improvements to the current version of the tool.

This research describes, develops a taxonomy of, and evaluates thirty-one knowledge-acquisition methods to determine which method matches a defined set of criteria. A method is chosen, extended, and automated in the form of a machine-aided knowledge-acquisition tool. The method is chosen based on five criteria including a connection between the chosen method and the information processing model of problem solving as defined by Newell and Simon (1972).

This research evaluates the performance of the tool in terms of the accuracy and consistency of the knowledge bases generated by using the tool. A baseline is derived from this study to which other knowledge-acquisition tools' performance can be compared. The evaluation methodology and metrics developed in this research can be used to evaluate other knowledge-acquisition tools.

From this research, four groups of changes to the automated knowledge-acquisition tool are suggested to improve the usability and performance of the tool. The changes are suggested for the user interface and the modes of operation of the tool.

ACKNOWLEDGMENTS

The research contained in this thesis was funded by U.S. Department of Energy Office of New Production Reactors Special Research Grant No. DE-FG02-91NP00119 entitled "Research and Development of Post-Classical Management Tools for Government Program Offices," with Management Systems Laboratories (MSL) of the Industrial and Systems Engineering Department of Virginia Polytechnic Institute and State University. I thank the Department of Energy's Office of New Production Reactors for sponsoring this research.

The development of the automated knowledge-acquisition tool was funded by the Office of Naval Research Project No. 230-11-11OF-106-353492-1 entitled "Automating a Detailed Cognitive Task Analysis for Structuring Curriculum," with Management Systems Laboratories (MSL) of the Industrial and Systems Engineering Department of Virginia Polytechnic Institute and State University. I thank the Office of Naval Research for sponsoring the development of the knowledge-acquisition tool.

I thank the following people for their guidance, feedback, and support throughout my research efforts: Dr. Harold Kurstedt for managing me through the research process; Dr. Bob Williges for helping me understand the process of research and helping keep the pre-proposal and proposal meetings on track; Jim Hughes for his input on how to proceed with research at MSL; Dr. Sherry Casali for her input on methodological issues; and Elliot Hultman for his support. I thank the members of the Cognitive Systems Laboratories for their support: Tom Carolyn, John Deighan, Asif Taiyabi, Jon Marquadt, Cindy Melfi, and Kevin Martin.

Special thanks goes to Dr. Betty Koball for her support and guidance in completing my work. Her input and guidance helped immensely in developing the final methodology. I thank Dr. Kent Williams for his guidance and his transfer of his knowledge to me. He has provided me with the tools and knowledge which will last a lifetime. I am indebted to my wife, Mary Lynn, for her support during these two years, her support made this journey much easier.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE - PURPOSE OF THIS DOCUMENT

This document defines the problem and its setting (Chapter 1), the review of the related literature (Chapter 2), the methodology for evaluation and validation of the developed tool (Chapter 3), discussion of the results from the experiment (Chapter 4), and conclusions with further research topics (Chapter 5).

CHAPTER 1 - THE PROBLEM AND ITS SETTING

1.1 Purpose of this Chapter

This chapter parallels the process used to define the research to be performed.

The following summarizes each section in this chapter:

Section	Description
Problem Statement	Describes what the problem this research is attempting to solve (Leedy, 1989).
Research Question	The question motivating the research.
Operationalized Research Question	Puts the motivating question in practical terms, a question we can do business with.
Conceptual Model and Delimitation	Picture of how key elements of the problem fit together or affect each other. Identifies what the problem is not, limits imposed on the study.
Research Purpose and Objective	Answers the question, "why are we doing this research project?" Describes what the tangible results of this research are.
Sub-Problems	Components of the research which partition the problem into pieces, partition of the conceptual model.
Outputs	Tangible results relating one-to-one to sub-problems. What tangible results come from addressing sub-problems.
Premises	Propositions offered as fact used to define, support, or put boundaries around the problem statement.
This Research and the NPR Grant	Describes how this research is related to the NPR grant.
Type of Research	Describes and justifies the type of research to be done.
Research Hypotheses	A statement about the expected relationships between two or more constructs in a theory or an explanation for a behavior, phenomenon, process, or event.
Research Methodology	Describes the overall process to be followed in doing this research.

Definition of Terms	Definitions of terms relevant to the study.

1.2 Problem Statement

> This research aims to design, develop, and test a knowledge-acquisition tool which performs knowledge acquisition by employing a cognitive task analysis, an extended version of GOMS, to construct problem spaces.

To support problem solving, decision making, and planning we need to first develop a means for diverse knowledge[1] sources to be related and integrated. Managers need to have the right knowledge at the right time to reach the right decision. No longer is all the knowledge contained in a single source. Groups of individuals with their own special knowledge and agendas are working together on complex problems which are comprised of many issues. Many people keep knowledge about each issue in their minds; and each person may know something different about each issue. By having these knowledge sources integrated, all the available knowledge can be brought to bear on the problem. This knowledge can then be stored and retrieved at the most appropriate time for use in making decisions leading to a planned strategy or problem solution. The chain of decisions represents a problem solution. A structured process to extract, integrate, store, and retrieve this knowledge must be implemented (Weber, Liou, Chen, & Nunamaker, 1990).

Problem solving, decision making, and planning are difficult today due to the complex business environment. When making decisions about major programs, such as Apollo in the past, and the space station and nuclear production reactors today, managers must consider many issues, e.g., quality, schedule, cost, critics, legislation, environment, safety, facilities, technical, personnel, and past and

[1] A distinction of data, information, and knowledge is made. A datum is a specific fact plus meaning and a unit of information is data or information compared to a reference. Knowledge is more than one piece of information in a pattern from which inferences and predictions can be made.

present strategies. Figure 1 displays this complex environment in which managers must make decisions.

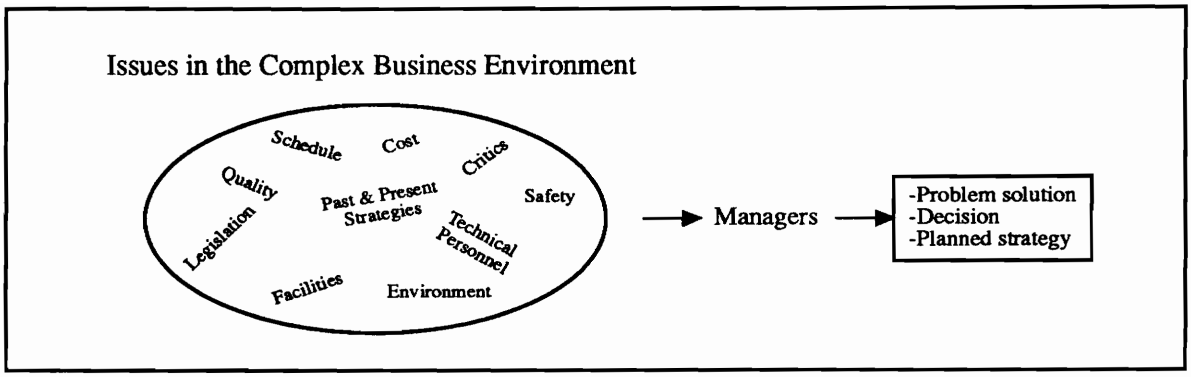


Figure 1. A complex business environment generates complex decision making.

1.3 Research Question

> How can we extend and automate GOMS in such a way that we can use this type of cognitive task analysis for knowledge acquisition for problem solving?

Managers and experts solve problems to make decisions and formulate plans. Problem solving is the overriding process by which decisions and plans are made. Problem solving is modeled by the information processing model of problem solving. Problem solving involves identifying the problem, building a problem space, and evaluating moves through the problem space to the solution.

To solve problems and make decisions, managers need knowledge. This knowledge makes up a problem space. The problem space includes: the goal and subgoals, possible states, operators that move the problem solver from state to state, and constraints on the problem (Voss, Tyler, & Yengo, 1983). Figure 2 is an abstract view of a problem space from which problem solvers make inferences. The top node is the starting point or initial state. The nodes are a subgoal or state[2]. The links are the operators to move from node to node. These operators can be viewed as an action or decision that allows the problem solver to move to the next state. The bottom node is the accomplished goal. Each goal/subgoal (node) and its set of operators (links) collectively represent a chunk of knowledge. At each state (node) a decision is being made based on the problem state and the knowledge available. The problem solver moves through the problem space to solve the problem. This movement involves making decisions about all alternatives described in the problem space. The chain of decisions represents a problem solution or a planned strategy to achieve a goal.

[2] A state is defined as the set of accomplished goals or subgoals, goals and subgoals to be achieved, and the operators available to achieve a goal or subgoal.

The problem space "consists of the information known or potentially available to the solver that may be useful in solving the problem" (Voss, Greene, Post, & Penner, 1983, p. 167). It represents all the knowledge that can be brought to bear in solving the problem. In fact, the problem space is a knowledge base, the "core rules and data that make up the domain knowledge" (Dym & Levitt, 1991, p. 113). A datum is a specific fact plus meaning and a unit of information is data or information compared to a reference (Berube, 1990). Knowledge is more than one piece of information in a pattern from which inferences and predictions can be made.

Both problem spaces and knowledge bases represent or model knowledge as production rules. A production rule takes the form of: If <conditions> then <action> (Kieras & Polson, 1985). The <action> is performed after the <conditions> are met. A condition is to an action as an operator is to a subgoal or as a subgoal is to a goal in the problem-solving model. For example, the goal is achieved after its subgoals are achieved: if <subgoals> achieved then <goal> achieved and at a lower level, if <operators> achieved then <subgoal> achieved.

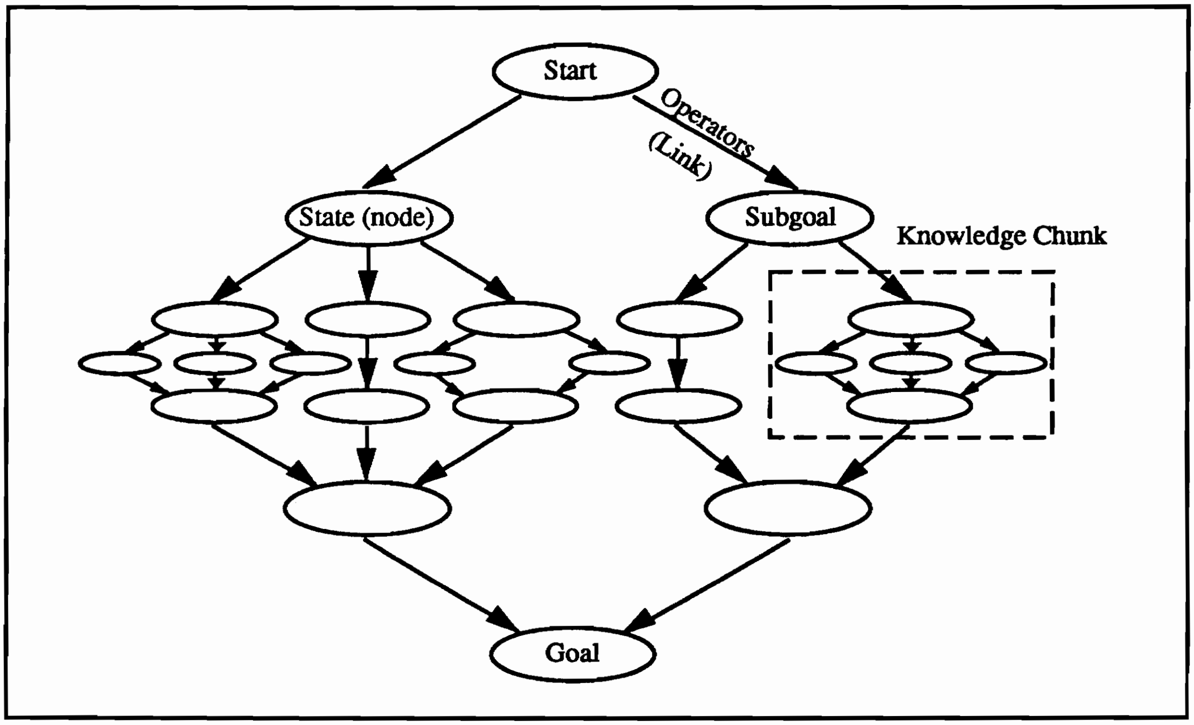


Figure 2. Abstract view of a problem space.
(Abstracted from the ideas presented by Voss, Tyler, & Yengo, 1983; Williams, 1991.)[3]

[3] The following convention is used to reference a figure or table: **unreferenced** signifies ideas of this author not attributable to a single source; **taken from** signifies a figure or table is taken from the source as is; **abstracted from** signifies the figure or table was developed from ideas presented by the source(s); and **adapted from** signifies the figure or table was adapted by this author, the adaption is noted.

The problem space is the foundation of problem solving because 1) it represents the generation of all possible alternatives from which decisions must be made and 2) it shows the impact of a decision on other decisions which must be made (Simon, 1960). To increase the efficiency and effectiveness of problem solving, tools must be built to elicit or extract and integrate the problem-space knowledge from experts and managers. This knowledge needs to be extracted, integrated, stored, and retrieved to aid problem solving. The knowledge represented as rules will determine what actions or decisions need to be made under a given set of conditions.

The extracted and integrated knowledge is the knowledge base. Knowledge acquisition is the extraction of an expert's knowledge. Knowledge acquisition is knowledge seeking. Gathering this knowledge is the most difficult and time-consuming process of developing a knowledge-based application. Knowledge acquisition is the bottleneck of knowledge-based system development (Dauer, 1990; Gaines, 1988; Grefenstette, Ramsey, & Schultz, 1990; Mockler, 1990; Olson & Rueter, 1987; Rowley, 1990). This bottleneck precludes the wide-spread application of knowledge-based technology. Automated knowledge acquisition can assist in overcoming this bottleneck.

Benefits of using an automated knowledge-acquisition tool include increased productivity of the knowledge engineer, reduced skill level required for knowledge acquisition, and extended use of knowledge-based systems (Trippi & Turban, 1990).

1.4 Operationalized Research Question

> How can we elicit expert knowledge from a group of managers, each with their own special knowledge, to support problem solving and decision making about a set of complex and difficult problems such as those encountered in planning?

In solving planning problems such as those encountered in government programs a diverse set of knowledge is needed. There are many people involved in the process and each have their own knowledge. The planning and construction of a new production reactor is an example of such a problem involving many people with a diverse knowledge base.

Each manager has his or her own special knowledge (e.g., safety and quality, environment, business management, construction management, support systems, design, and facilities and equipment) to influence a decision (U. S. Department of Energy: Office of New Production Reactors, 1990). We need to extract, organize, store, retrieve, and integrate their expertise to support decision making now for New Production Reactors (NPR) and in future endeavors similar to NPR.

For managers to make decisions specific to the new production reactor, they must either formally or informally in their minds make a representation of the problem space. This representation is an organization of all the knowledge that can be brought to bear in defining the problem. Specifically, the representation is made of initial conditions, potential intermediate subgoals that must be achieved, kinds of moves that allow one to accomplish these intermediate subgoals, constraints in terms of limiting conditions, and the desired end state or goal (Kieras, 1988; Voss, Greene, Post, & Penner, 1983; Voss, Tyler, & Yengo, 1983; Williams, 1991). With this problem space, managers identify all the decisions they will make. Each node in the problem space is a decision to be made about which link or operator should be performed to achieve the given goal or subgoal. These decisions must take into consideration such constraints as cost, schedule, quality, and critics.

The integration and portrayal of knowledge consisting of the problem space employed by experts to generate a solution is necessary to formulate a planned strategy. A mechanism supporting the integration and portrayal of knowledge will aid decision makers in the non-routine, uncertain aspects of their decision-making responsibilities. A structured methodology guiding the knowledge-acquisition process to construct the problem space needs to be developed and implemented (Weber, Liou, Chen, & Nunamaker, 1990).

1.5 Conceptual Model and Delimitation

> This research will focus on a knowledge-acquisition tool that guides experts through the presently intuitive steps of conducting a cognitive task analysis, performing the knowledge-acquisition process, and eliciting a problem space, all leading to solving and identifying problems.

To increase the efficiency and effectiveness of problem solving, tools must be built to help experts identify problems and then elicit the problem space knowledge from them. This knowledge needs to be extracted, related, stored, and retrieved to aid problem solving. The knowledge-acquisition tool will extract and relate the knowledge by performing the knowledge-acquisition process using a cognitive task analysis technique. As a result of the knowledge acquisition, a representation of the problem space is constructed. This problem space is used to guide managers in developing solutions or plans to solve a problem.

By having the problem-space knowledge integrated and stored, the knowledge can be retrieved to help monitor and verify decisions made. Measurement, data gathering, and analysis in the monitoring and verification process will provide information and knowledge about new more-specific problems in accomplishing the goal. The recursive problem-solving cycle repeats itself until the goal has been achieved and all sub-problems have been solved. Figure 3 is the conceptual model of this research.

This research will focus on designing, developing, and testing a knowledge-acquisition tool which performs the knowledge-acquisition process by employing an extended GOMS cognitive task analysis to construct a problem-space representation. The dotted box of Figure 3 shows the area of the conceptual model the research will focus on.

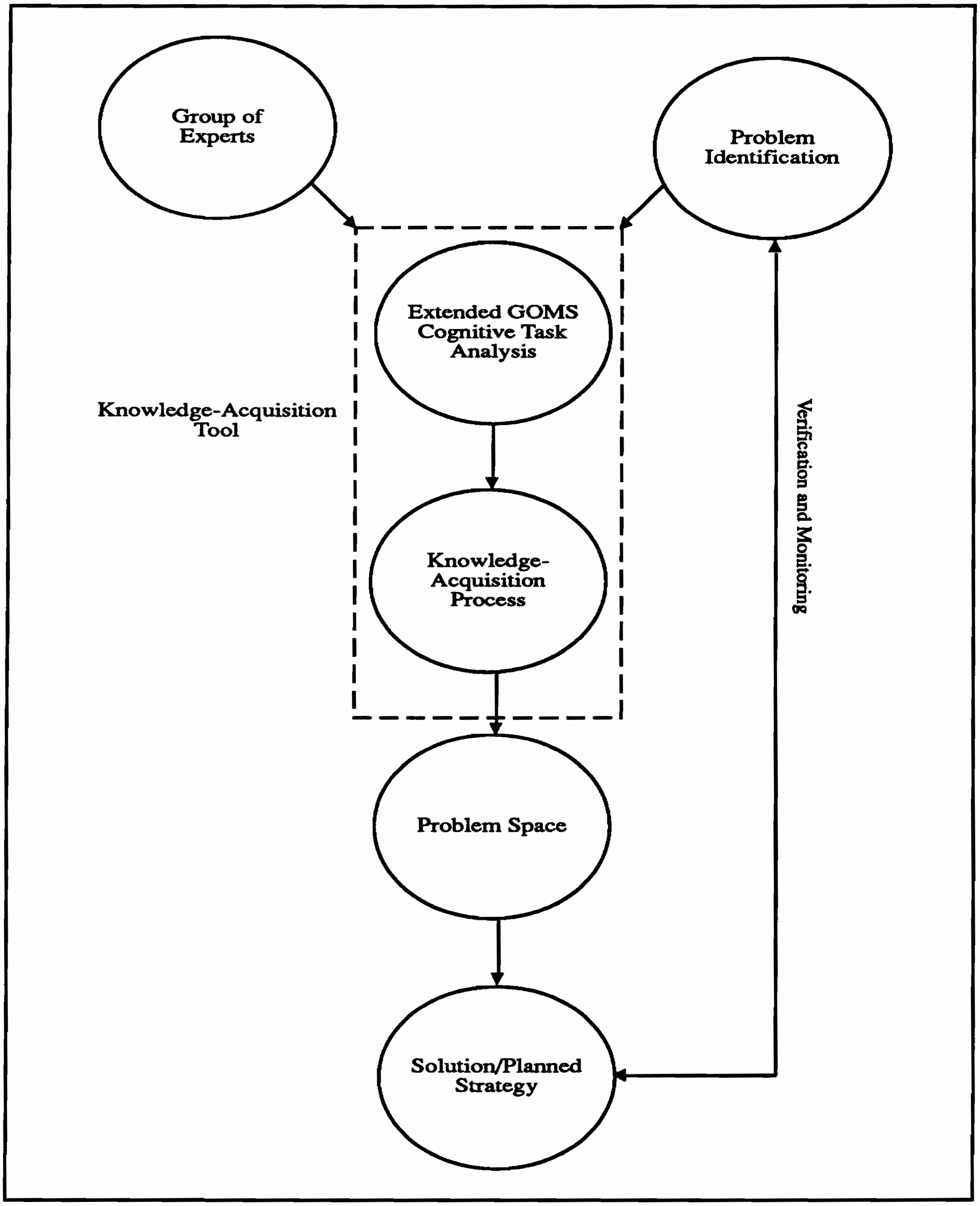

Group of Experts
Problem Identification
Extended GOMS Cognitive Task Analysis
Knowledge-Acquisition Tool
Knowledge-Acquisition Process
Verification and Monitoring
Problem Space
Solution/Planned Strategy

Figure 3. Conceptual model with the research delimitation.

1.6 Research Purpose and Objective

> The purpose of this research is to further the development of automated knowledge-acquisition techniques for knowledge-based representations as they may be used in problem-solving activities. The objective of this research is to design, develop, and test an automated knowledge-acquisition tool using a cognitive task analysis process for guiding individuals in organizing their knowledge to formulate problem spaces from which decisions and plans can be made.

Benefits of using knowledge-based systems range from internal cost savings to consistency in decision making to preserving high-valued organization expertise (Feigenbaum, McCorduck, & Nii, 1990). The wide-spread use of this technology has been hampered by the high cost of the knowledge-base development process. The development process involves eliciting, organizing, representing, refining, and verifying the domain knowledge elicited from an expert or group of experts. The process is time consuming and inefficient. The two most common ways to develop the required knowledge base are:

1) a knowledge engineer works with an expert, or
2) a knowledge engineer becomes an expert (Olson & Rueter, 1987).

In the first case, eliciting the knowledge from the expert is the task. However, this is easier said than done. Numerous existing techniques help the process. In the second case, the knowledge engineer becomes an expert, which is time consuming because the learned expertise takes time and resources. There is a need to facilitate 1) the knowledge engineer's interaction with the expert and/or 2) the expert's capability as a knowledge engineer.

A cognitive task analysis is a process of constructing an explicit representation or model of a person's knowledge (Kieras, 1988). A cognitive task analysis process can be used to elicit from the expert the knowledge needed to develop the problem space. Knowledge is a collection of information in a pattern from which inferences can be made. An automated knowledge-acquisition tool for conducting a cognitive task analysis will facilitate knowledge acquisition for problem spaces by providing a structured, systematic methodology to elicit, organize, represent, refine, and verify the knowledge. This knowledge can support an individual or group of individuals in solving problems by identifying knowledge needs and decisions they need to make.

By developing and implementing an automated tool, the domain expert can easily transfer his or her knowledge; thus, the knowledge-acquisition bottleneck can be removed.

1.7 Sub-Problems

This section partitions my research problem into components. The following three sub-problems partition my conceptual model:

1. Understand current knowledge-acquisition methods.
2. Understand problem solving and problem spaces.
3. Understand cognitive task analysis.

1.8 Outputs

This section defines the tangible results relating one-to-one to the sub-problems.

The following are my three outputs of the sub-problems:

1. A taxonomy of current knowledge-acquisition methods.
2. A conceptual framework of problem solving.
3. A conceptual framework of cognitive task analysis.

1.9 Premises

This section defines the propositions offered as fact to define, support, or put boundaries around the problem statement. The following premises are used to state the source of the problem-space description is valid.

1. The information processing model of problem solving is a valid model.
2. The information processing model of problem solving is valid for both individuals and groups (Prietula, Beauclair, & Lerch, 1990).

1.10 This Research and the NPR Grant

> This research focuses on designing, developing, and testing a knowledge-acquisition tool which is used for integrating and portraying information and knowledge in a grand strategy system.

This research will design, develop, and test an intelligent management tool to aid problem solving, decision making, and planning in a grand strategy system. A grand strategy system is used to manage and plan for the future by understanding the past, present, and future activities; internal and external forces; and plans of the organization. The grand strategy system is an iterative process of documenting and evaluating past strategies; analyzing internal and external forces; formulating planned strategy; and administering, implementing, and verifying status. Each of these steps involves eliciting and integrating information and knowledge. The tool will be used as the "brains" for "Integrating and Portraying Information and Knowledge" as displayed in the dynamic process model of the grand strategy system (Management Systems Laboratories & Virginia Productivity Center, 1992). Figure 4 shows the integration of the research conceptual model with delimitation and the dynamic process model of the grand strategy system.

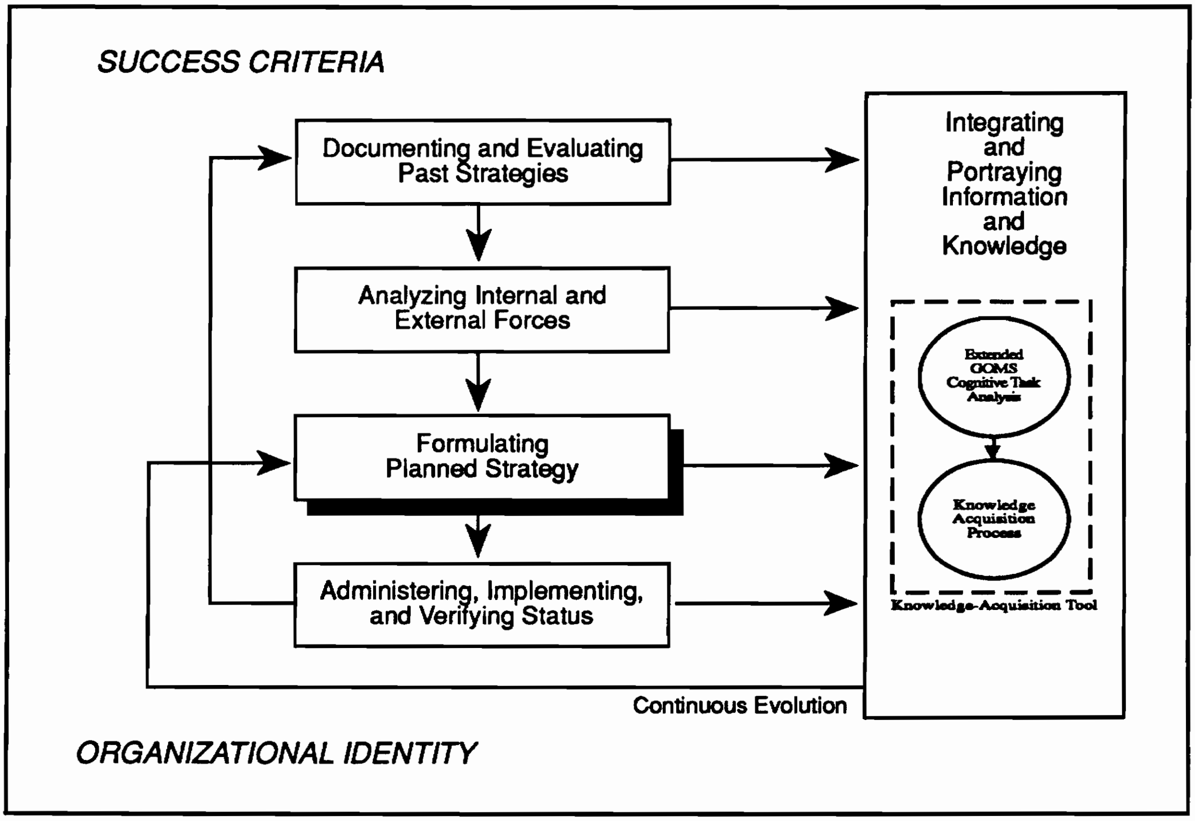

SUCCESS CRITERIA
Documenting and Evaluating Past Strategies
Analyzing Internal and External Forces
Formulating Planned Strategy
Administering, Implementing, and Verifying Status
Integrating and Portraying Information and Knowledge
Extended GOMS Cognitive Task Analysis
Knowledge Acquisition Process
Knowledge-Acquisition Tool
Continuous Evolution
ORGANIZATIONAL IDENTITY

Figure 4. This research and the NPR grant.
(Adapted from Management Systems Laboratories & Virginia Productivity Center, 1992, by adding the conceptual model to the "Integrating and Portraying Information and Knowledge" box.)

1.11 Type of Research

> This research is exploratory development research because it will identify the important issues and concepts to be used in designing and developing an automated knowledge-acquisition tool for problem-space knowledge.

The research to be conducted is exploratory development research. Exploratory development is "research and development toward a specific problem." It is "the exploration of new technologies or concepts that hold promise for application to specific" needs (Driskell & Olmstead, 1989, p. 51).

This research is exploratory research because a tool to elicit the problem-space knowledge has not been developed. Other researchers have attempted, but did not reach the full potential of such a tool because their methods were not based upon theoretical and/or empirical evidence. This research will be integrating basic theoretical and empirical evidence as the foundation of the design and development of the tool.

This research is also a design problem. The goal of this research is to design and develop a knowledge-acquisition tool. Design involves studying, analyzing, evaluating, and integrating the sub-components for a system (Saunders, 1982). This research will study, analyze, and evaluate problem solving, cognitive task analysis, and knowledge acquisition. The results from the above will used in designing an automated knowledge-acquisition tool.

1.12 Research Hypothesis

This section describes the expected relationships between two or more constructs of this research. The following represents the overall supposition of this study:

> A knowledge-acquisition tool employing an extended GOMS cognitive task analysis process will generate consistent and accurate knowledge bases.

1.13 Overall Research Plan

The following seven steps will be the used in guiding the overall completion of this research.

1. Understand current knowledge acquisition. (Review body of knowledge.)
   a. Classify methods based on characteristics.
   b. Develop criteria to analyze methods.
   c. Conduct a comparative analysis of methods based on criteria.

2. Understand problem solving. (Review body of knowledge.)
   a. Research problem solving and a problem-solving model.
   b. Identify components of the problem-solving model, e.g., problem representation and space.

3. Understand cognitive task analysis. (Review body of knowledge.)
   a. Research a model of cognitive task analysis.
   b. Identify components of a cognitive task-analysis method, e.g., GOMS.

4. Select method to use in tool.
   a. Match characteristics of knowledge-acquisition or cognitive task-analysis methods with the problem-solving model.

5. Work with the design team of Kent Williams and John Deighan (programmer) to design constraints[4].
   a. Integrate results from 1, 2, 3, and 4.
   b. Develop process flow diagrams.

6. Refine design and tool.

7. Evaluate and validate tool. This is discussed in greater detail in Chapter 3.

[4] I have contributed to the design and development of the tool in the following ways: 1) completed flowcharts of the process, 2) integrated design characteristics from the literature into the tool, 3) developed the prompt strings and help files, 4) developed and employed a testing plan to debug the system, and 5) iteratively used and tested the tool which provided feedback on the tool itself. The tool was designed by Kent Williams, John Deighan, and myself. John Deighan completed the actual coding of the tool. A description of the tool is given in Appendix 8.

1.14 Definition of Terms

Cognitive Task Analysis - constructing an explicit representation or model of a person's knowledge (Kieras, 1988).

Datum - a specific fact plus meaning.

Decision Making - choosing an alternative.

Domain Knowledge - "knowledge specific to the domain or field in which the problem is defined" (Dym & Levitt, 1991, p. 15).

Expert - person with special knowledge.

Information - data or information compared to a reference (Berube, 1990).

Knowledge - more than one piece of information in a pattern from which explicit inferences and predictions can be made. This knowledge is represented using if <conditions> then <action> production rules.

Knowledge Base - "the core rules and data that make up the domain knowledge" (Dym & Levitt, 1991, p. 113).

Knowledge-Based System - "a computer representation that uses knowledge and problem-solving paradigms on a skill level comparable to that of human experts" (Abdul-gader & Kozar, 1990, p. 61).

Knowledge Engineer - "the person who designs and builds expert systems" (Feigenbaum, McCorduck, & Nii, 1990, p. 319).

Knowledge Acquisition - "the process that extracts knowledge from a source (e.g. a domain expert or textbook) and incorporates it into a knowledge-based system that solves some problem" (Bylander & Chandrasekaran, 1988, p. 65).

Problem Space - "consists of the information known or potentially available to the solver that may be useful in solving the problem" (Voss, Greene, Post, & Penner, 1983, p. 167).

Solution/Planned Strategy - a means to solve a problem, chain of decisions leading to the solution.

State - the set of accomplished goals or subgoals, goals and subgoals to be achieved, and the operators available to achieve a goal.

CHAPTER 2 - BODY OF KNOWLEDGE

2.1 Relevance of the Body of Knowledge

> *Today national wealth arises from knowledge, expertise, innovation, and intellectual capital.*
>
> (Feigenbaum, McCorduck, & Nii, 1990, p. 3).

Managers and experts solve problems and make decisions and plans. Problem solving involves identifying the problem, building a problem space, and evaluating moves through the problem space to the solution. Managers need knowledge to solve problems and make decisions and plans. To increase the efficiency and effectiveness of problem solving, tools must be built to elicit the knowledge from groups of experts and managers. This knowledge needs to be extracted, integrated, stored, and retrieved to aid problem solving.

The following literature review describes problem-solving and knowledge-acquisition methods. Problem solving is investigated because we must understand the problem-solving process before we can help managers find solutions. We must first know the components of problem solving before we can elicit these components. Knowledge-acquisition methods were reviewed because they describe how the knowledge which represents a problem is elicited and organized. This review was used to determine the method used to acquire the problem-space knowledge.

2.2 Problem Solving

Problem solving is investigated because it must be understood before tools can be made to help managers find solutions. The components of problem solving must first be known before the components can be elicited.

2.2.1 Problem Solving Model

Solving problems is a part of life. How people solve problems, the problem-solving process, must be understood before aids to problem solving can be constructed. The Newell and Simon (1972) information processing model of problem solving provides a model to define this process.

According to this model, problem solving has two stages: 1) a representation stage and 2) a solution stage. In the representation stage, the problem solver interprets the problem, develops a problem representation, and develops a problem space. In the solution stage, the problem solver uses an evaluation function to move through the problem space to a solution.

When solving a problem, a person is first presented with a task under a given set of circumstances representing the problem statement. The problem solver interprets the problem statement to construct a problem representation and from this the problem space. The problem representation contains the initial states and goal of the task. The problem space includes the initial states, goal and subgoals, possible intermediate states, operators which move the problem solver from state to state, and constraints on the problem.

In the solution stage, the problem solver moves through the problem space evaluating each state-to-state move and identifying operators to perform the movement. An evaluation function determines which move from state to state should be taken to lead to the goal (Simon, 1975). This choice is based upon the knowledge available at the time of decision.

2.2.2 Problem Solving Content and Process

Problem solving can be divided into the content and the process which acts on the content. The content is the problem space represented as production rules and a database of facts. A production rule takes the form of: If <conditions> then <action> (Kieras & Polson, 1985). The <action> is performed after the <conditions> are met. The database is a set of facts relevant to the problem.

The problem solving process is a production system process by which inferences are made by comparing the production rules and database (Simon, 1975). The process involves comparing the production rules to the database to see if the conditions of a production are satisfied. If a production's conditions are satisfied then it is fired, the <action> is *performed.* If more than one production's conditions are satisfied, then a conflict resolution function is used to choose the appropriate production to fire. An inference is made when a rule is fired, and this firing adds facts to the database. This cycle is repeated until the problem is solved or no more inferences can be made.

2.3 Knowledge Acquisition

Knowledge acquisition is "the process that extracts knowledge from a source (e.g., a domain expert or textbook) and incorporates it into a knowledge-based system that solves some problem" (Bylander & Chandrasekaran, 1988, p. 65). Knowledge acquisition is the iterative process by which knowledge is: 1) elicited, 2) organized, 3) represented, 4) refined, and 5) verified for use in a knowledge-based system to solve problems. Current methods of knowledge acquisition are considered the bottleneck to further use of knowledge-based systems.

Knowledge elicitation is the first step to knowledge acquisition. Knowledge elicitation involves acquiring or drawing the knowledge from sources (e.g., expert, case examples, or reference books). This step is mainly concerned with what knowledge a source has and how to best acquire the knowledge. Knowledge elicitation, the first step, is knowledge seeking. Knowledge organization, the second step, takes the elicited knowledge and organizes it to show the relationships between large clusters of the knowledge. The goal is to take the elicited knowledge and group it as the expert typically does. Knowledge representation, the third knowledge-acquisition step, structures and formats or encodes the organized knowledge into a form such as a production rule acceptable by a specific knowledge-based system. The fourth step, knowledge base refinement, involves checking for inconsistencies, gaps in logic, conflicts, contradictions, and completeness in the knowledge base. Verification, the final step, is done to determine if the knowledge base can yield accurate inferences in solving real world problems within the domain being modeled.

Knowledge acquisition is the bottleneck of knowledge-base development. It is time-consuming, tedious, and prone to error (Abrett & Burstein, 1988; Boose & Bradshaw, 1988; Cleaves, 1988; Cooke & McDonald, 1988; Dauer, 1990; Gaines, 1988; Garg-Janardan & Salvendy, 1988; Grefenstette, Ramsey, & Schultz, 1990; Gruber & Cohen, 1988; LaFrance, 1988; Littman, 1988; Mockler, 1990; Moore & Agogino, 1988; Olson & Rueter, 1987; Regoczei & Plantinga, 1988; Rowley, 1990). An efficient, systematic, structured approach to performing the knowledge-acquisition process is unavailable (Di Piazza, 1990; Hayward, Woelinga, & Breuker, 1988).

2.4 Knowledge-Acquisition Methods

Knowledge-acquisition methods were reviewed because they describe how the knowledge which represents a problem is elicited and organized. This review was used to determine the method used to acquire the problem-space knowledge.

Based upon this research and analysis, three categories of knowledge-acquisition methods were defined: 1) manual, 2) machine aided, and 3) machine learning. This is done based upon the level of the knowledge engineer's interaction in the knowledge-acquisition process.

Knowledge-acquisition methods were investigated because they provide the means to capture an expert's knowledge (i.e., problem-space representation). A single complete source of information of the knowledge-acquisition methods does not exist. Books, journal articles, proposals, and proceedings on knowledge acquisition were collected. The methods were analyzed with respect to five criteria. 1) The method must be general purpose and not constrained by the requirement for a pre-existing domain knowledge base prior to the commencement of knowledge acquisition, making it applicable to a variety of differing domains of application. 2) The method must produce a set of procedural units which lend themselves to processing via a production system, since a production system process is the most widely employed architecture for processing cognitive activities. 3) The method must be systematic to the extent that a process can be formulated and codified for computer implementation. 4) The method must be compatible with the manner in which procedural knowledge is typically recalled. This should supply the element of simplicity to the knowledge-acquisition process. 5) The method must be valid in terms of its being empirically grounded in experimental research attesting to its compatibility with the human information processing system. These criteria provide a means to evaluate and compare the different knowledge-acquisition methods.

Figure 5 presents the taxonomy of knowledge-acquisition methods which was created as a result of the review and subsequent analysis. The first level (manual, machine aided, and machine learning) depicts the required level of a knowledge engineer's interaction. The lower levels group the methods by the specific means they employ to elicit the knowledge.

Manual methods require the knowledge engineer (KE) to be directly involved in the complete process. The knowledge must first be elicited and then manually organized. Once the knowledge is represented, the KE must manually encode the knowledge in a form acceptable to a specific knowledge-base system. Finally, the refinement and verification is done by a manual step-by-step examination of the knowledge base and the inferences derived from the knowledge base. Figure 6 provides an abstract view of the manual knowledge-acquisition process.

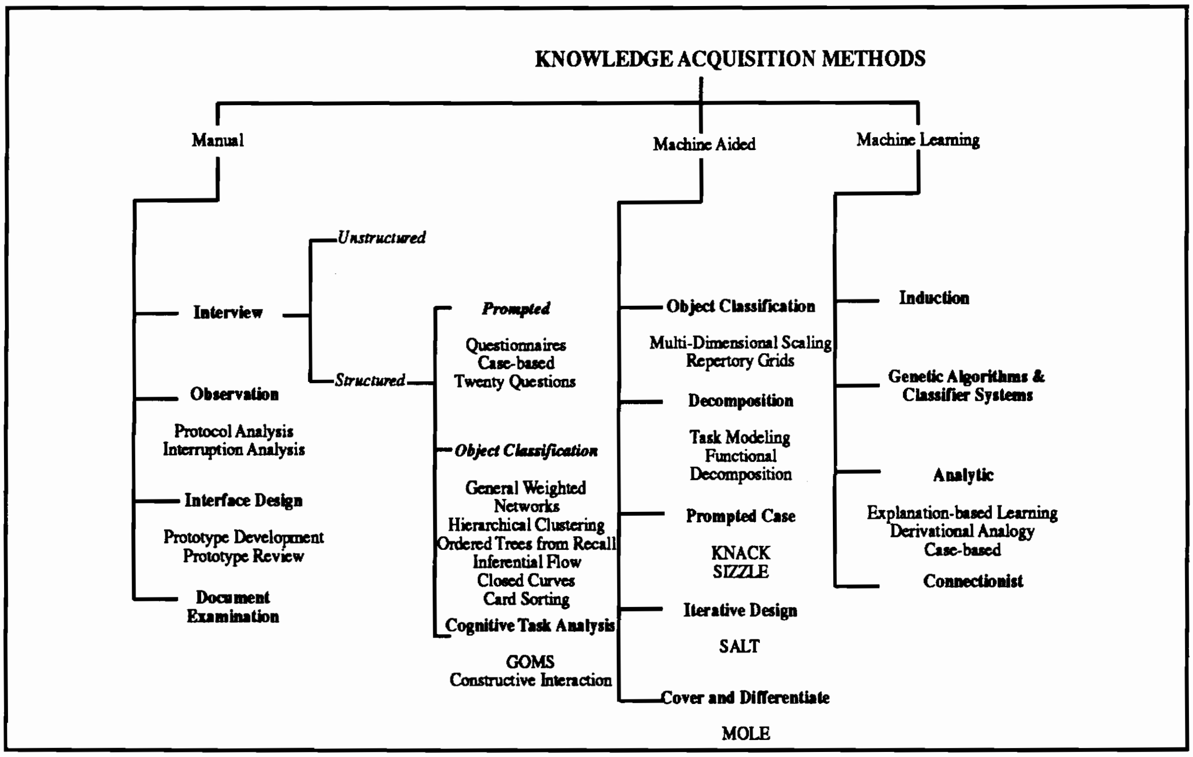


Figure 5. Taxonomy of knowledge acquisition methods.

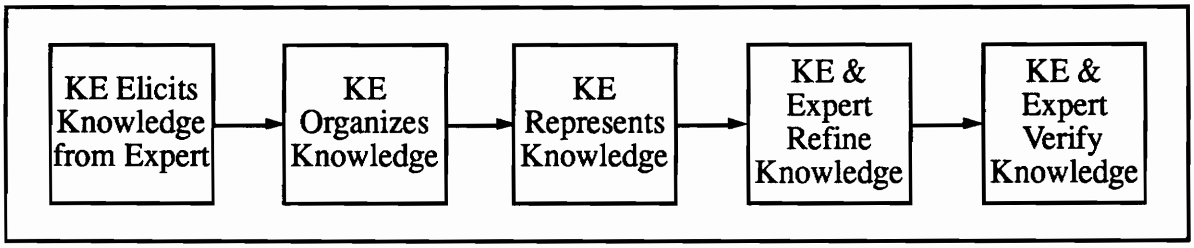


Figure 6. Abstract view of manual knowledge acquisition.

Machine-aided methods elicit, organize, represent, and refine the knowledge interactively with the expert. As the elicitation process proceeds, the machine implicitly organizes and represents the knowledge. Most machine-aided tools provide facilities to interactively refine the knowledge base. Figure 7 provides an abstract view of machine-aided knowledge acquisition. Machine-aided methods are typically automated versions of the manual methods. The automation aides in the organization, representation, and refinement phases.

The machine-learning methods require very little direct interaction on the part of the expert or KE. He or she is responsible for providing or gathering the data to be used by the machine-learning method. Machine-learning methods generate the knowledge from data. These methods automatically organize, represent, and refine the knowledge base. Machine learning is the epitome of knowledge acquisition. Figure 8 provides an abstract view of the machine-learning knowledge-acquisition process.

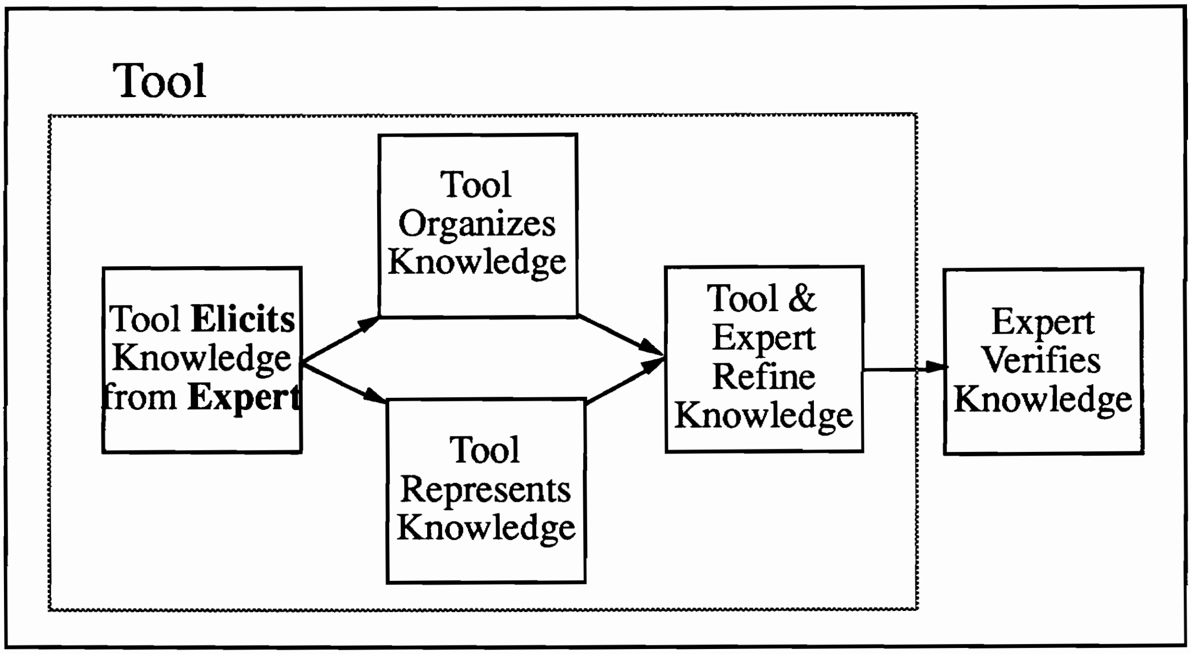

Tool
Tool Elicits Knowledge from Expert
Tool Organizes Knowledge
Tool Represents Knowledge
Tool & Expert Refine Knowledge
Expert Verifies Knowledge

Figure 7. Abstract view of machine-aided knowledge acquisition.

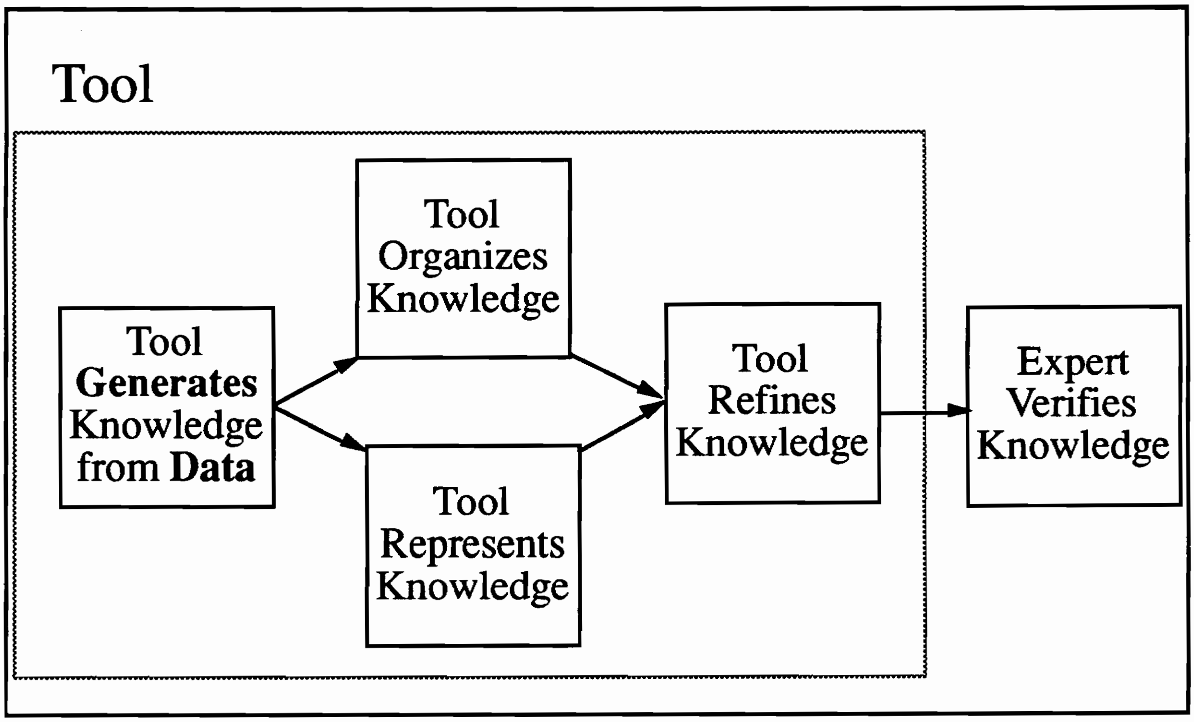

Tool
Tool Generates Knowledge from Data
Tool Organizes Knowledge
Tool Represents Knowledge
Tool Refines Knowledge
Expert Verifies Knowledge

Figure 8. Abstract view of machine-learning knowledge acquisition.

2.4.1 Manual Methods

Manual knowledge-acquisition methods, the first major set of knowledge-acquisition methods, require the KE and expert to complete the knowledge engineering process with very little aid from a machine. In most cases, the KE must manually construct the representation and then enter it in a specific format of the software system shell. The manual methods can be classified as Interview, Observation, Interface Design, and Document Examination. These methods are distinguished by the nature of the task the KE and expert performs.

2.4.1.1 Interview Methods

Interviews are conducted between a KE and an expert or group of experts (O'Leary & Watkins, 1990). The basic process is a question and answer session(s) between the KE and expert. This method can be very time consuming (Dauer, 1990; Olson & Rueter, 1987). Interviewing is also limited by the expert's ability to express himself in a meaningful way to the KE. The KE's knowledge of the domain can also restrict the detail of the knowledge acquired. The more the KE knows about a domain the more detailed questions can be asked and the answers understood. Typically, the KE's domain knowledge grows during the knowledge-acquisition process. The KE reviews earlier portions of the interview to enhance the level of detail elicited.

There are two major categories of methods used for interviewing. They are Structured and Unstructured methods. This categorization is based upon the presence or lack of explicit structure placed on the interviewing process.

2.4.1.1.1 Structured-Interview Methods

Structured-interview methods use closed questions in a structured, planned sequence to elicit domain knowledge (Welbank, 1990). A closed question asks for a specific answer. The following is an example of a closed question: What is the next step of the process? Structured-interview methods include the subclasses of Prompted Interview, Object Classification, and Cognitive Task Analysis.

2.4.1.1.1.1 Prompted Interview

Prompted-interview methods use a "prompt" to guide the expert's response. The structure is in the method chosen to prompt the expert. Examples of prompted interview are: Case-based, Questionnaires, and Twenty Questions.

2.4.1.1.1.1.a Case-Based Interview

When using a case-based interview, the expert is asked to solve a domain case developed by another expert or taken from knowledge of past problems in the domain. For example, a diesel mechanic would be given the case of a faulty generator which precludes starting the engine. As the mechanic solves the problem, the KE would record the knowledge used and the order in which the mechanic used it. The KE gets an understanding of the underlying reasoning process and tactics used to solve the example case (Welbank, 1990). The KE needs to have a good understanding of the case and domain so he or she can follow the expert's process and minimize distractions to the expert (Dauer, 1990). The level of detailed knowledge is dependent upon the example's detail. The more specific the case, the more specific the gathered knowledge. By having many experts solve the same case, the conclusions can be verified before being used in a final application.

2.4.1.1.1.1.b Questionnaires

Questionnaires are the second form of a prompted-interview method. These can be completed by the expert at his or her convenience. They do not necessarily require face-to-face interaction between the expert and KE. Olson and Rueter (1987) give an example of a questionnaire used to uncover the objects of a domain. They use index cards each of which asks the expert to describe a domain variable, the type of values the variable may be assigned, and the range of values. Next, the expert is asked to draw the relationship among the variables. This method is good for uncovering the important domain objects or concepts, but the reasoning of the expert can't be traced. Therefore, the level of detail is low. The results are limited by the expert's ability to explicitly describe the domain in terms of variables. Considerable knowledge on the part of the KE is required in the design of the questionnaire.

2.4.1.1.1.1.c Twenty Questions

Twenty questions is the third example of a prompted-interview method. The KE chooses a domain problem and the expert asks "twenty" yes-no questions trying to diagnose the problem. This is useful for mapping and paring down the search space in diagnostic-classification problems. The KE needs a good understanding of the domain because it might be difficult for him or her to answer the expert's questions, a major difficulty with this method. Additionally, the line of reasoning employed by the expert to generate his questions is not made explicit (Welbank, 1990).

2.4.1.1.1.2 Object Classification

Object classification, the second group of structured-interview methods, is used to classify or group objects or concepts. Object classification methods include:

General Weighted Networks, Hierarchical Clustering, Ordered Trees from Recall, Inferential Flow, Closed Curves, and Card Sorting. Each of these methods provide structure to the interviewing process by asking specific questions about domain objects or items. The expert is asked to describe relations among objects and this information is analyzed to produce a classification of the objects. These methods are used to elicit the way in which the expert clusters, relates, or organizes the domain objects. Top-level detail with little insight into the detailed reasoning process concerning a domain is abstracted from using these methods.

2.4.1.1.1.2.a General Weighted Networks

General weighted networks develop a network of objects and their relationships. The expert gives symmetric distance judgments (e.g., A is to B as B is to A) on all possible pairs of objects. The distance is a judgment call by the expert which says how closely two items are related. These distances form a half-distance matrix used for further analysis. Table 1 shows a half-distance matrix for an example general weighted-network problem.

From the half-distance matrix a minimal connected network (MCN) is constructed by connecting only the items that are most closely related. The MCN is made by first connecting the two items closest to each other as determined by the shortest distance between these two items in the matrix. For example, from Table 1, the goat and sheep or goat and cow would first be joined. For the example presented, the goat and sheep are selected to be joined first. Next, an item not yet on the network with the shortest distance to any item on the network is added to the network. The cow and then the pig would be added to the network next because these are the next two shortest distances in the table from an object already on the network to an object that is not. This step iterates until all items are part of the network. Figure 9 shows the MCN for the matrix given in Table 1.

Next, a minimal elaborated network (MEN) is constructed by adding a link to the network if and only if it is shorter than any path between the two not directly linked objects (Olson & Rueter, 1987). Figure 10 shows the MEN for the example given in Table 1. The dotted lines represent the added links. For example, the shortest path from the MCN (Figure 9) between the goat and horse is five:

{ 1 (goat-sheep) + 1 (sheep-cow) + 3 (cow-horse) }.

However, from Table 1, the distance between the goat and horse is four, therefore the link is added.

These networks are analyzed for dominating concepts and members of cycles (items linked into circles) or fully related objects (Olson & Rueter, 1987). A dominating concept has more links to other concepts than any other one. This can reveal how the expert structures objects and the strengths of the relationships. This process could be difficult when there are a significant number of objects to analyze.

Table 1
Half-Distance Matrix for Example General Weighted Network Problem
(Taken from Olson & Rueter, 1987.)

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

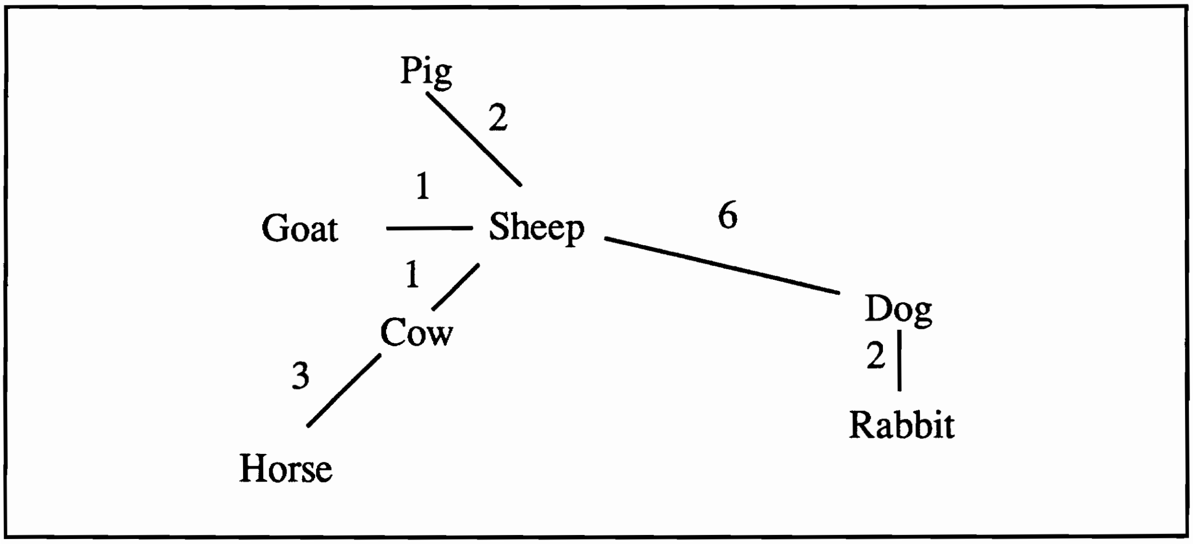


Figure 9. MCN for example general weighted network problem.
(Taken from Olson & Rueter, 1987.)

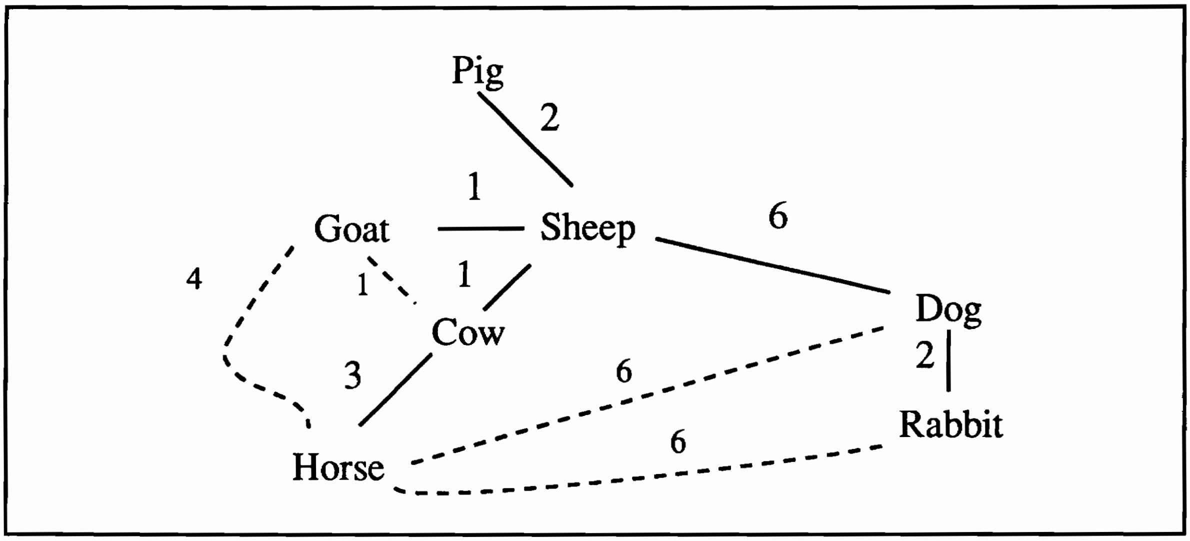


Figure 10. MEN for example general weighted network problem. (Taken from Olson & Rueter, 1987.)

2.4.1.1.1.2.b Hierarchical Clustering

Hierarchical clustering is used to construct clusters and classifications of objects. Hierarchical clustering uses the distance between objects as the basis for comparison by grouping items with the smallest distances between them. From a half-matrix of distances as previously shown in Table 1, items with the smallest distance between them are joined into a cluster(s) forming a level of the hierarchy. A new matrix is computed by using each clustered item as a new "combined item." The distances from a "combined item" to other combined or un-combined items is computed using one of the following rules:

Min (distance from object to each object in an object-cluster),
Max (distance from object to each object in an object-cluster), or
Avg (distance from object to each object in an object-cluster).

For example, cow, sheep, and goat are closest and, therefore, form a cluster. Next, the new distance matrix is computed by using the minimum rule given above, the distance from the new cow-sheep-goat cluster or combined item to the pig is computed as: min { 3 (goat-pig), 3(cow-pig), 2(sheep-pig)} which is 2. Table 2 shows the matrix after the first iteration using the minimum rule. If all items are not connected, then a new higher level of the hierarchy is created and the process continues. The second iteration of this process begins with the values of the matrix shown in Table 2. From the values shown in Table 2, pig would be joined with the cluster cow-sheep-goat in the second level. Dog and rabbit would also be joined in the second level. This can be seen in Figure 11.

As each iteration is performed a level of the hierarchy is being formed. The first clusters form the lower level and each additional cluster is getting closer to the top of the hierarchy. Figure 11 shows the completed hierarchy. The numbers to the left of the hierarchy represent the distances at which the clusters were combined.

The detail of the resultant hierarchical classification depends upon the number of individual clusters, data items, and hierarchy levels. The process assumes an item is or isn't a member of a cluster based upon the value of the distance between items of the cluster, that is, the more similar two items are the closer they are in a cluster at some level of the hierarchy (Olson & Rueter, 1987). As with the other object classification methods, hierarchical clustering is limited by the number of distance values needed to be supplied. As more objects are used, the number of distances increases. For each item, n(n-1)/2 distance relations are needed to fill the half-matrix of distances.

Table 2
Revised Matrix for Hierarchical Clustering Example after Using Minimum Rule (Taken from Olson & Rueter, 1987.)

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

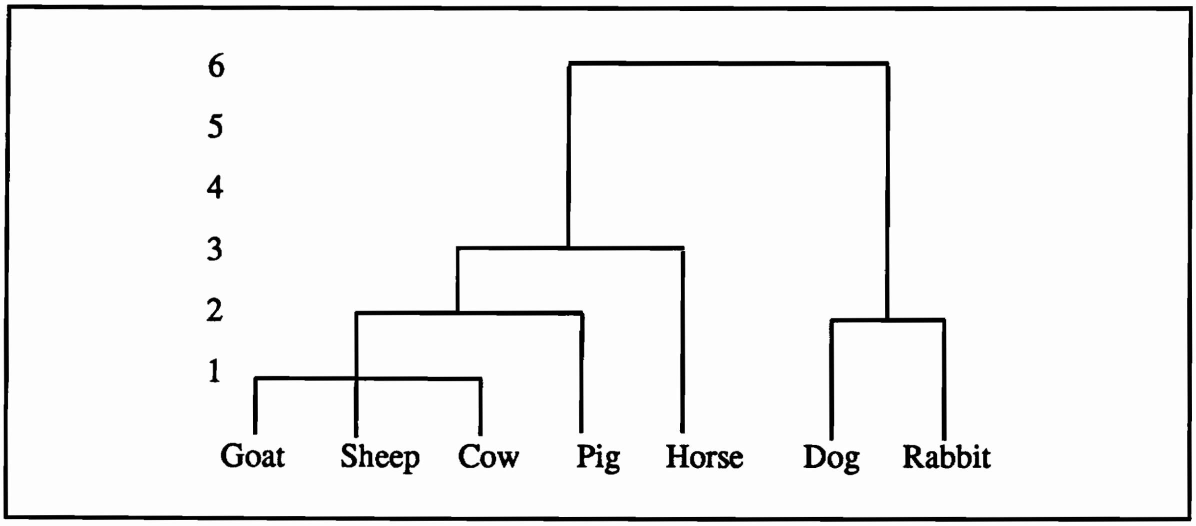


Figure 11. Completed hierarchy for hierarchical clustering problem. (Taken from Olson & Rueter, 1987.)

2.4.1.1.1.2.c Ordered Trees from Recall

Ordered trees from recall assumes items in a cluster are recalled before items from another cluster are recalled (Olson & Rueter, 1987). The expert is asked to recall a list of items ten to twenty times with the KE providing the first item on various trials. The results from the trials are analyzed for clusters of items which are consistently recalled together. Items that are consistently recalled together are grouped into a cluster. As shown in the recall trials of Figure 12, cow and horse; goat, sheep, and pig; and dog and rabbit are consistently grouped. These groups each form the lowest level of the hierarchy. The order in which the items are recalled also indicates groupings. The groups horse and cow; and goat, sheep, and pig are consistently recalled "next" to each other and thus form the next level of the hierarchy. From this analysis an ordered tree is constructed (Olson & Rueter, 1987). Figure 12 displays the recall trials and resulting ordered tree.

The method is limited by the number of items that can be recalled. As with all object classification methods, there is very little, if any, detail regarding the explicit reasoning process employed by the expert.

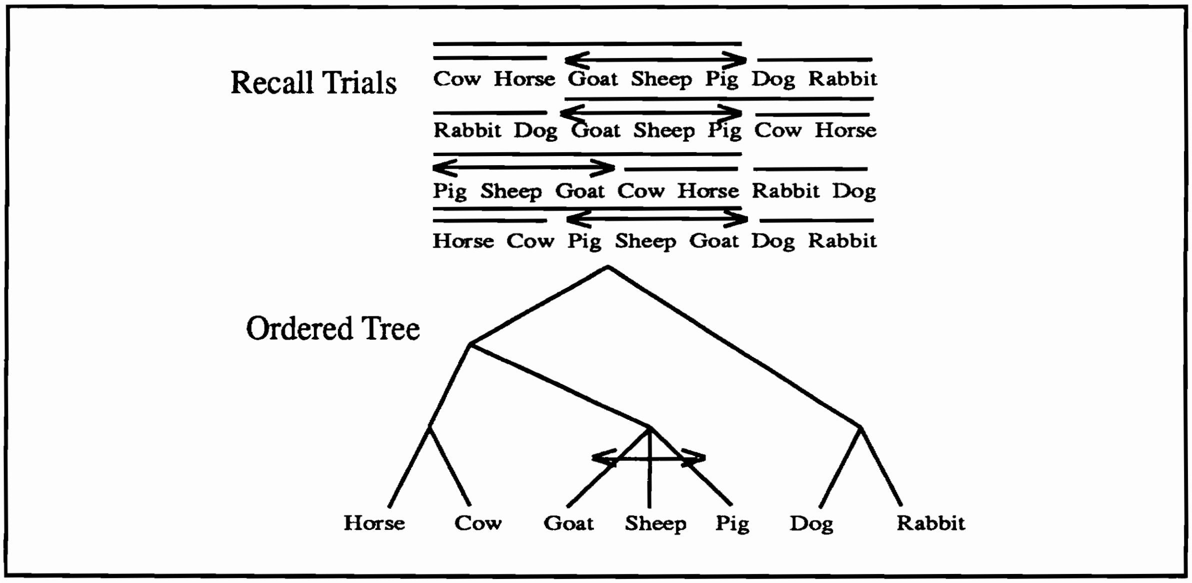

Recall Trials
Cow Horse Goat Sheep Pig Dog Rabbit
Rabbit Dog Goat Sheep Pig Cow Horse
Pig Sheep Goat Cow Horse Rabbit Dog
Horse Cow Pig Sheep Goat Dog Rabbit
Ordered Tree
Horse
Cow
Goat
Sheep
Pig
Dog
Rabbit

Figure 12. Recall trials and ordered tree example.
(Taken from Olson & Rueter, 1987.)

2.4.1.1.1.2.d Inferential Flow

The inferential flow method yields a network expressing cause-and-effect relations among concepts or objects. Starting with a list of key domain objects, the KE questions the expert about the cause-and-effect relationships between two objects. A standard weight between 0.0 and 1.0 is assigned between two objects the first time they are mentioned in a relationship. A network is constructed by assigning these strengths to the links between objects (Olson & Rueter, 1987). Each subsequent time two objects are mentioned together in the expert's reply, the strength is updated to some value between its current value and 1.0. The strengths could be positive or negative for representing direct and inverse relationships, respectively. A complete network is constructed from the combination of objects and strengths between them. Figure 13 depicts an example of an inferential flow network.

2.4.1.1.1.2.e Closed Curves

The closed curve method is used to find the relationships of objects that are encoded in a physical space (Olson & Rueter, 1987). The expert is given a spatial representation of items. He or she then draws a closed curve around the group of objects which belong together physically. This is limited by the interpretation of the closed curves and the ability to draw the relationships.

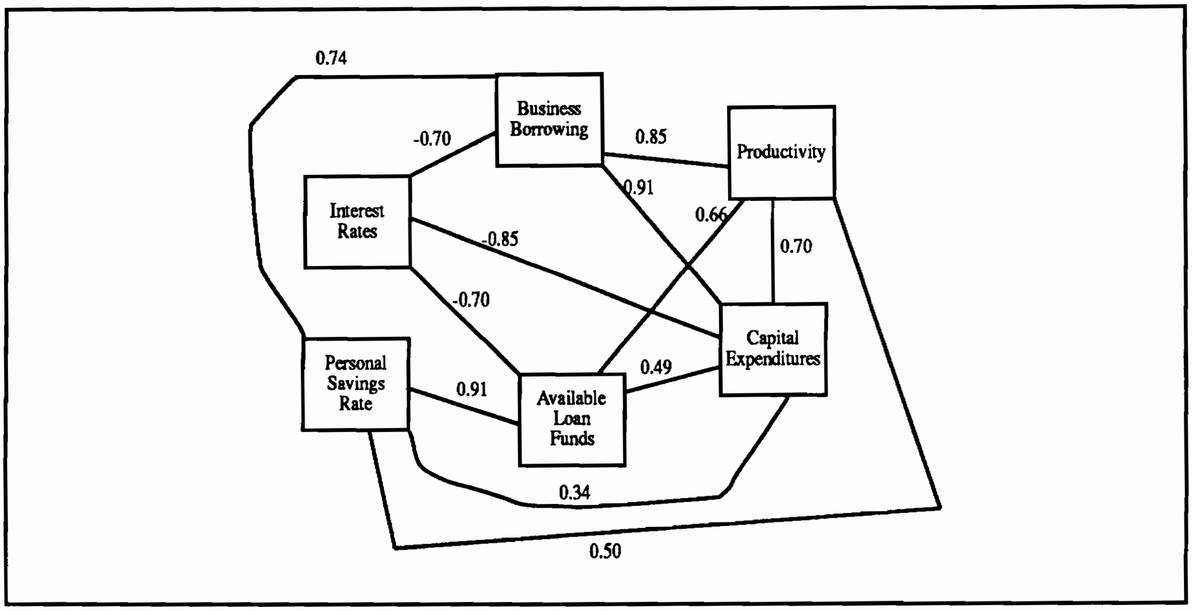


Figure 13. Inferential flow network.
(Taken from Olson & Rueter, 1987.)

2.4.1.1.1.2.f Card Sorting

Card sorting is used as a prompting device for initial knowledge acquisition (Dauer, 1990; Welbank, 1990). The expert is given a deck of index cards prepared by the KE, each card contains a basic domain object or concept. The expert sorts through the cards, placing related concepts in a category based upon his or her domain knowledge (Welbank, 1990). He or she then describes to the KE the interrelationships among the categories (Dauer, 1990). The expert could also be given three cards with concepts written on them to describe how the two are related and the third concept differs. Card sorting is limited by the ability of the expert to explicitly group and describe the relationship of the items.

2.4.1.1.1.3 Cognitive Task Analyses

Cognitive task analyses, the third group of structured-interview methods, are used to construct an explicit model of a persons procedural knowledge used in performing a task (Kieras, 1988, p. 135) and how one models knowledge related to the workings of complex physical systems (Miyake, 1986). Examples of cognitive task analysis methods are GOMS (goals, operators, methods, and selection rules) and Constructive Interaction.

2.4.1.1.1.3.a GOMS

GOMS was developed by Card, Moran, and Newell to construct models of text-editing. GOMS is a hierarchical goal decomposition method. It has been used to analyze manuscript editing (Card, Moran, & Newell, 1983) and human-computer interaction (Kieras, 1988; Kieras & Polson, 1985).

A GOMS model has four sets of components: 1) goals, 2) operators, 3) methods, and 4) selection rules. A goal defines a state to be achieved. An operator is an elementary specific effort or act. A method is a set of steps used to accomplish a goal. A selection rule is used to choose among alternative methods towards achieving a goal. The relationship of these components is shown in Figure 14.

The GOMS process involves specifying the goal to be accomplished. The list of steps needed to be performed to accomplish the goal is specified. This list of steps makes up a method. If more than one method exists for accomplishing a specific goal, then a selection rule is specified. The selection rule describes the condition(s) that determines which method of the alternatives should be used to accomplish the goal. From Figure 14, method-1 or method-2 can be used to achieve subgoal-1. Which method is chosen is based on conditions specified by the selection rule. The process continues in a top-down breadth-first expansion of goals. Each step of a method becomes a new sub-goal; and the process continues until all steps have been expanded. For example, in Figure 14, step 1-1 is made a goal and operator-1 and operator-2 need to be achieved to achieve step 1-1. An operator is a step which cannot be further decomposed (i.e., a primitive step). When all steps are defined as operators/primitives the process stops. A set of production rules used in performing the task is generated from the methods and selection rules.

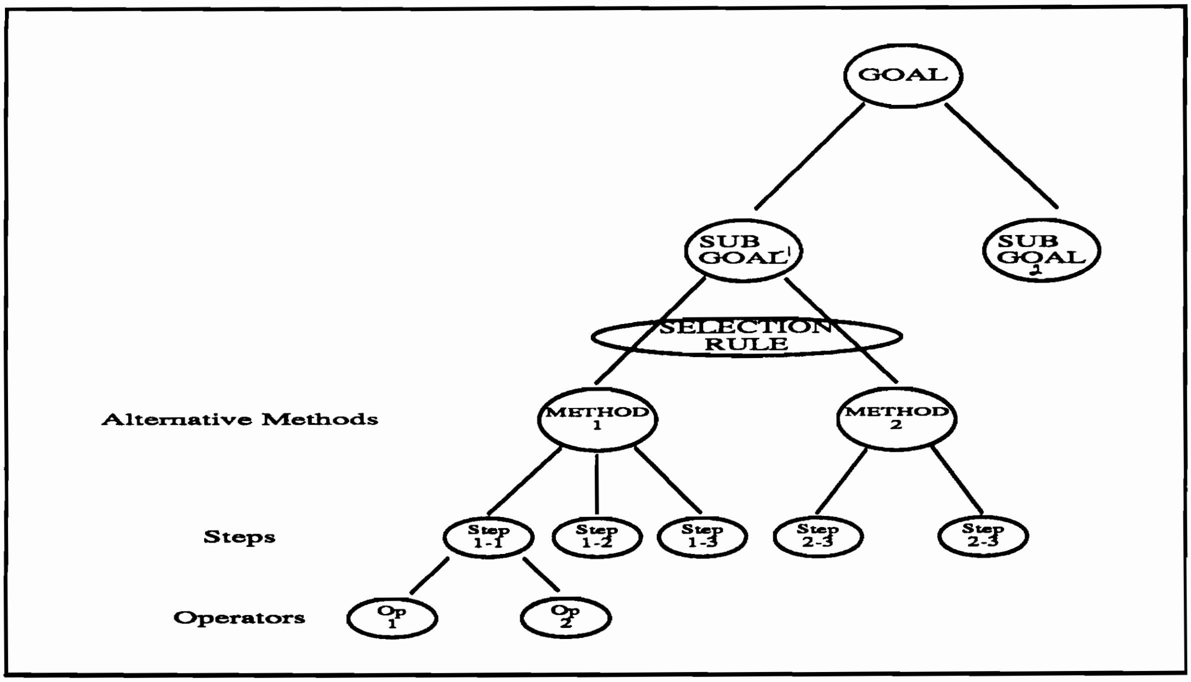


Figure 14. An example of a GOMS hierarchy.
(Abstracted from ideas of Card, Moran, & Newell, 1983; Kieras, 1988; Kieras & Polson, 1985; Williams, 1991.)

2.4.1.1.1.3.b Constructive Interaction

Constructive interaction is a framework which describes the iterative process of understanding. Miyake (1986) developed the method based upon verbal protocols she collected from subjects discussing how a complex physical device works. She found that people proceed in a top-down, breadth-first fashion progressing from one level of understanding to the next level of understanding. In a sense, people continue to ask why or how something works until they fully understand the smallest and deepest sub-components of a system.

Constructive interaction yields a function-mechanism hierarchy. The highest-level function describes the physical system in terms of "what happens" and the mechanism describes the "how it happens". A mechanism is an explanation or set of connected functions. The constructive interaction process iteratively breaks down a system into lower subsystems each of which performs a specific function which is accomplished by a specific mechanism. The mechanism at one level becomes the function to be explained at the next lower level. For example, the function at level n is described by the mechanism at level n+1. The mechanism at level n+1 then becomes the function to be explained by the mechanism at level n+2. There may be alternative mechanisms for enabling a specific function. In this respect, the structure of a function-mechanism hierarchy is similar to that of the goal-subgoal hierarchy and the associated methods hierarchy of the GOMS task-analysis process for modeling procedural tasks.

<u>2.4.1.1.2 Unstructured Interview</u>

The unstructured interview, as its name implies, uses very general and open-ended questions (Welbank, 1990). Open-ended questions ask for general information. Examples of open-ended questions are: How does this process work? How do you perform the process? This method eventually leads into a structured interview as the KE learns more about the domain. It is useful for initial knowledge acquisition.

2.4.1.2 Observation

Observation is the second set of manual knowledge-acquisition methods. The expert performs the task while the KE watches and/or videotapes the process. A transcript is created from the observations. A number of techniques can then be used to analyze the transcript (e.g., a GOMS analysis, discourse analysis, coherence analysis, etc.). Observation methods are useful for identifying problem-solving strategies, studying motor skills, and verifying experts' task descriptions (Welbank, 1990). Observation methods include Protocol Analysis and Interruption Analysis.

2.4.1.2.1 Protocol Analysis

Protocol analysis is used to catch the expert in the act of performing the task being analyzed (Ericson & Simon, 1984). As the expert performs the task, he or she describes aloud what comes into consciousness. The expert actually thinks aloud. The KE transcribes the expert's verbalizations and, on occasion, may prompt the expert by reminding him or her to continue verbalizing. The KE then analyzes his or her notes, the videotape, and the transcription of the videotape to draw inferences about the expert's thought process. By using protocol analysis, very detailed information can be gathered.

2.4.1.2.2 Interruption Analysis

Interruption analysis follows the same basic process as protocol analysis. However, the expert performs the task without verbalizing aloud what he or she is doing until the KE doesn't understand the task (Olson & Rueter, 1987). At which time the KE interrupts and asks the expert to describe what he or she is doing. This method is useful for verification of a developed system. The KE needs to have a good understanding of the domain.

2.4.1.3 Interface Design

Interface design is the third set of manual knowledge-acquisition methods. Interface design methods include Prototype Development and Prototype Review (Welbank, 1990). These methods have the KE and expert work together to describe and evaluate prototype knowledge-based systems.

2.4.1.3.1 Prototype Development

Prototype development is used to gather an initial description of the knowledge-based system. The KE and expert work together to design the knowledge-based system's interface. In the course of this prototype development process, different kinds of information are abstracted: 1) information the system will prompt the user to enter, 2) the order in which the information is requested, and 3) the semantics to use in the prompts. This is a high-level acquisition method. A detailed thought and reasoning process is not acquired.

2.4.1.3.2 Prototype Review

Prototype review is used to verify and extend a developed knowledge-based system. The prototype is shown frame-by-frame and the expert comments on how to improve or change the system. Prototype review is used for knowledge base refinement, the fourth step of the knowledge-acquisition process. Corrections and/or clarifications to the system are noted. Prototype review should not be done early in the development process because credibility of the project could be decreased due to a mismatch between the expert's expectations of the system and the system's capabilities at the time of review.

2.4.1.4 Document Examination

Document examination is the fourth set of manual knowledge-acquisition methods. This is very useful for both initial and detailed knowledge acquisition of domain theory and principles. Reference books such as troubleshooting manuals provide a wealth of information to start system development. Some document examination should be done before working with experts so the KE can gain a good initial understanding of the domain.

2.4.2 Machine-Aided Methods

Machine-aided methods are the second of the three categories of knowledge-acquisition methods. These methods automatically and interactively elicit, organize, represent, and refine knowledge from KEs or domain experts. The organization and representation steps are implicit within the tool's process. Machine-aided methods guide the knowledge-acquisition process by using the manual structured-interview methods in an automated form of a computer program. By using these methods experts can directly transfer their knowledge independent of a KE (Diederich, Ruhmann, & May, 1988). These systems are typically referred to as automated knowledge-acquisition or knowledge-elicitation tools. The methods these machine-aided tools employ have been further divided into Object Classification, Decomposition, Prompted Case, Iterative Design, and Cover-and-Differentiate.

2.4.2.1 Object Classification

Object-classification methods involve, as in the manual object-classification methods, eliciting from the expert knowledge about relationships among objects. Two such machine-aided object-classification tools are Multi-Dimensional Scaling and Repertory Grids.

2.4.2.1.1 Multi-Dimensional Scaling

Multi-dimensional scaling (MDS) has its origins in experimental psychology. The purpose of MDS is to reveal whatever pattern or structure may otherwise lie hidden in a matrix of empirical data (Shepard, 1972). The output of MDS is a spatial model showing the relationship between items relative to some number of dimensions.

The empirical data are given by the expert as a half-matrix of similarity judgments (e.g., A is to B as B is to A) between objects. The similarity judgments are then analyzed using multi-dimensional scaling analysis, which organizes the data to determine the best clusters of objects according to the user supplied similarity matrix. Objects that are more similar (i.e., a large similarity value) should be closer to each other in a multi-dimensional space than objects that are less similar (i.e., small similarity value). The analysis determines the number of dimensions that best account for the similarity judgments between all pairs of objects. A plot representing where objects fall on these dimensions is created. The expert then provides the names for the dimensions.

Figure 15 shows the results of MDS for career or job titles. The two dimensions for the plot are *dependency* on others to do the job (horizontal axis) and *prestige* of the job (vertical axis). From the plot, the jobs "physician" and "psychologist" are less dependent and more prestigious than the job of "laborer" or "clerk."

This method is limited by the interpretation of the plot and the number of similarity judgments the expert is expected to provide, n(n-1)/2 judgments are needed for n objects.

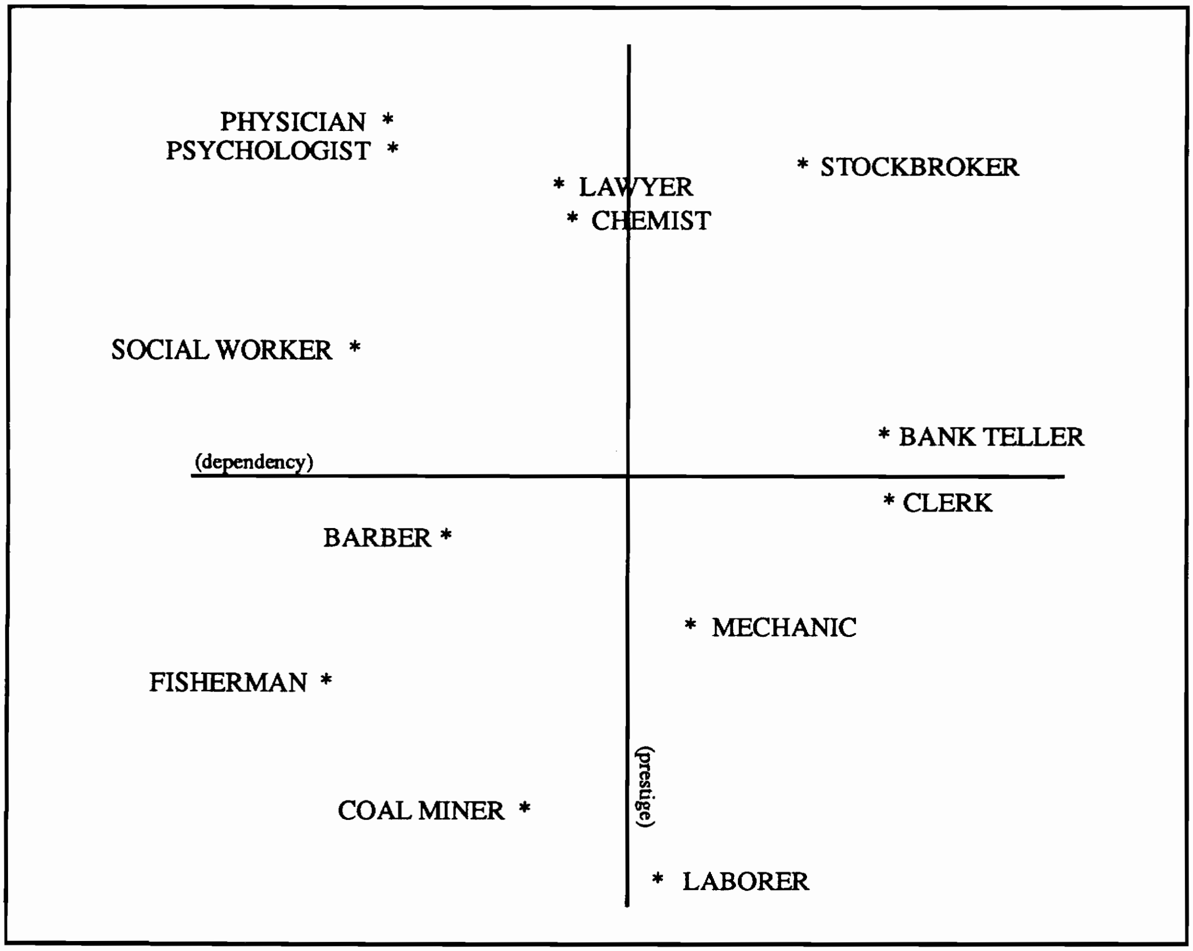
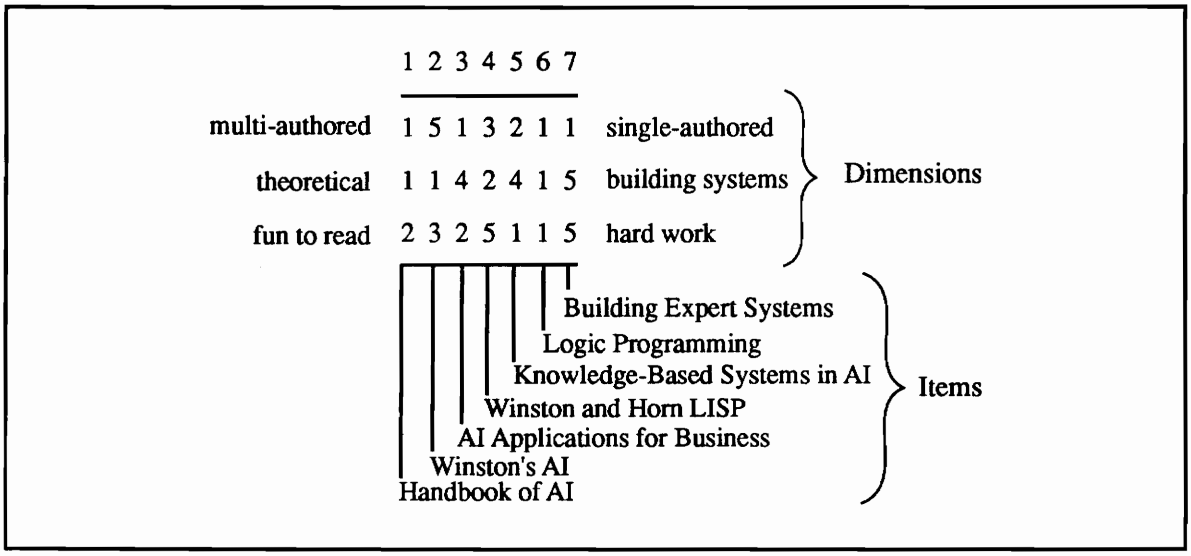


Figure 15. Example plot from MDS.
(Adapted from Burton, 1972, by not placing all occupations from original plot. A simplified version is presented.)

2.4.2.1.2 Repertory Grids

The repertory grid method is based upon Kelly's (1955) personal construct theory. This technique is generally used for classification-diagnosis problems (Boose & Bradshaw, 1988).

The expert is first asked to provide a list of the items to be compared. The expert is then presented with three items. The expert tells which item is different from the other two. He or she is then asked how it is different or what characteristic of the items makes the two similar to each other and different from the third. What makes the two items similar and the third item different defines a dimension. He or she assigns a name to the distinguishing dimension along with a scale, indicating a high and low range. The two similar items are each given a value representing where they fall on the dimension. For example, as shown in Figure 16, items Building Expert Systems and Logic Programming are similar to each other by being multiple authored and are different from Winston's AI which is single authored. Number of authors is the dimension (Shaw & Gaines, 1988). The remaining items are then rated on the same dimension by the expert. This process continues until all items are rated on all of the dimensions identified. Figure 16 is an example of a repertory grid.



	1	2	3	4	5	6	7	
multi-authored	1	5	1	3	2	1	1	single-authored
theoretical	1	1	4	2	4	1	5	building systems
fun to read	2	3	2	5	1	1	5	hard work

Building Expert Systems
Logic Programming
Knowledge-Based Systems in AI
Winston and Horn LISP
AI Applications for Business
Winston's AI
Handbook of AI

Figure 16. Example of a repertory grid.
(Taken from Shaw & Gaines, 1988.)

The machine automatically and implicitly organizes and represents the knowledge. Based on a machine-calculated half-matrix, the hierarchical clustering method (as reviewed in the manual structured interview object classification section) is applied to form a hierarchy of the items (Olson & Rueter, 1987). From these groups, inheritance networks are generated. Boose and Bradshaw use this method in their Aquinas and Axotl knowledge-acquisition tools (Boose & Bradshaw, 1988; Bradshaw & Boose, 1990). The level of detail depends upon the objects, dimensions, and interactions of objects and dimensions.

2.4.2.2 Decomposition

Machine-aided decomposition attempts to decompose an object or concept into its components. Two such methods of machine-aided decomposition include Task Modeling and Functional Decomposition.

2.4.2.2.1 Task Modeling

The task modeling method is used in the Logic Aids Program (LAP) developed by Di Piazza (1990). The tool interviews the expert about assumptions and goals of the expert system. The tool develops and organizes domain rules by employing a structured-interview technique.

The process begins by interviewing the expert for the overall goal of the system for which the knowledge being elicited will be used. This goal is used as the starting point for the following set of questions iteratively asked by LAP of the expert[5]:

[5] For this dialogue and others in the machine-aided section, the following convention is used to help distinguish between machine and expert inputs and outputs: **bold** is used for machine dialogue, **bold-underline** is used for values the expert provided and are being used as part of the system's dialogue, and *italics* are used for the expert's current input.

1) The expert is asked for the conditions under which the expert would perform an action. The goal or action is corrective-action = pump-ftt-to-sea.

   If ________________ = ________________
   corrective-action = pump-ftt-to-sea

   becomes

   If _*trim-condition* = *heavy-forward*
   corrective-action = **pump-ftt-to-sea**

2) The expert is asked (by one or more of eight methods) if the condition is necessary to performing the task. Can the action be completed under different conditions? For example,

   Will *trim-condition* = *heavy-forward* **always lead to** **corrective-action** = **pump-ftt-to-sea?**

   If the answer is yes, then the process continues with step 3. If the answer is no, then the process returns to step 1 to have the expert define another condition for the action by asking:

   What is an alternative conclusion?

3) Is the condition given in step 2 the new goal?

   Is trim-condition = heavy-forward your new goal?

   If the answer is yes, then the process begins with step 1 again and the new goal is the condition. If the answer is no, then the process continues until all goals or actions have conditions which describe when an action should be taken.

As a result of this top-down, goal-decomposition method a set of production rules are created modeling the expert's domain knowledge.

2.4.2.2.2 Functional Decomposition

The functional decomposition method is employed by Pugh and Price (1990) in their Functional Reasoner tool which interviews an expert about the composition of a system, its components, and their interrelated states. This functional-decomposition tool is based on the generic task described by Bylander and Chandrasekaran (1988).

The following set of questions taken from Pugh and Price (1990) are iteratively asked by the system of the expert:

1) The expert is asked for the name of the object.

What is the name of the device you are building? *circuit 1*

2) The expert is asked for the preconditions of the object to work properly.

Please give the general preconditions for [circuit 1] to function properly:
1. *power supply working correctly*
2. *bulbs working correctly*

3) The expert is asked for the list of possible states the device can be in.

Please give the list of states that [circuit 1] can be in:
1. *on*
2. *off*

4) The expert is asked for the causes of a state change for the object.

Given that the [circuit 1] is in the following state: [on]
If <what happens> to [circuit 1] to cause a change of state and/or a side effect to occur.
Causes:
1. *power supply switched off*

5) The expert is asked for the new state the object enters after a cause has acted upon it.

Given that [power supply switched off] has acted upon [circuit 1]
Which of the following states does the [circuit 1] enter?
1. [on]
2. [*off*]

6) The expert is asked for any side effects which might occur due to the state change.

Please list any side effects that occur.
Side Effect:
1. *bulb 1 goes out*
2. *bulb 2 goes out*

7) The expert is asked if there are any more conditions which might facilitate a state change given the object's current state.

Are there any other causes (or list of causes) which facilitate:
1. a state change and/or
2. a side effect when the [circuit 1] is in state [on]? (y/n)

If the answer is yes, then the process continues with step 4 using the current state. If the answer is no, then the process continues with step 8.

8) For this example, the process would continue for state "off" of component "circuit 1" and continues from step 4. If all the states have been defined for the object, then the process continues with step 9.

9) The expert is then asked for any sub-devices for the device.

Please give the list of sub-devices for the [circuit 1] (if any):
1. *power supply*
2. *wire a*
3. *wire b*
4. *wire c*
5. *bulb 1*
6. *bulb 2*

10) The expert is asked for the possible states of the sub-devices given in step 9.

Please give the list of states that [power supply] can be in:
1. *on*
2. *off*

Please give the list of states that [wire a] can be in:
1. *conducting*
2. *nonconducting*

This step continues for each sub-device identified in step 9.

11) The expert is asked for the state of the sub-devices given the higher level object's state

Given that the [circuit 1] is in the following state: [on]
Please input the state of each of the sub-devices.

[power supply]
1. [on]
2. [*off*]

[wire a]
1. [conducting]
2. [*nonconducting*]

This process continues for the remaining sub-devices defined in step 9 (i.e., wire b, wire c, bulb 1, and bulb 2) and every state of the higher level object (i.e., on and off).

12) The expert is then asked for the effects of interaction between sub-devices.

Given that the [circuit 1] is in the following state: [on]
And that the states of the sub-devices are:

1. [power supply]	**State: [on]**
2. [wire a]	**State: [conducting]**
3. [wire b]	**State: [conducting]**
4. [wire c]	**State: [conducting]**
5. [bulb 1]	**State: [lit]**
6. [bulb 2]	**State: [lit]**

When the cause [power supply switched off] on the device [circuit 1] occurs, which of the following sub-devices is affected?

1. [*power supply*]
2. [wire a]
3. [wire b]
4. [wire c]
5. [bulb 1]
6. [bulb 2]

The expert chooses an object and answers the following question for each object selected:

Which of the states does the [power supply] enter?
States
1. [on]
2. [*off*]

This step iteratively occurs until the complete set of potential interactions is elicited. Each iteration displays the state change for each sub-device. After every interaction is done for a given state of the higher level object, the next state of the higher level object is performed for this step.

13) The process begins again at step 9 with each sub-device becoming a device. The process stops when no more sub-devices are defined.

In summary, the overall process involves eliciting the object, preconditions for proper object functioning, a list of states for the object, causes for changes of states, related state changes, sub-devices of the object, states of a sub-device, state status of sub-device given object's state, and the effect of an object state change on sub-devices.

The result of this process is a functional description, a representation of a device described in terms of the function and interaction of each of its components (Pugh & Price, 1990). The level of detail is solely determined by the user in specifying the sub-devices.

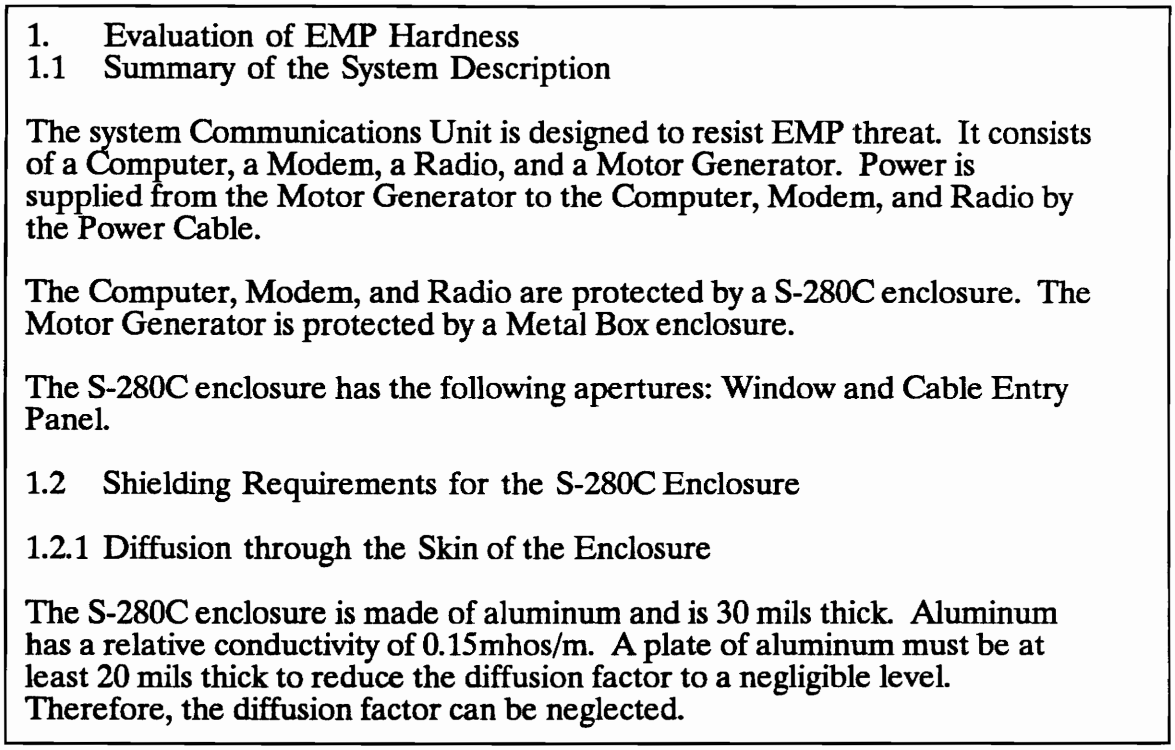
2.4.2.3 Prompted Case

The prompted-case method uses an expert case description interactively adapting this case description to solve a problem. Two examples of systems employing this method include KNACK and SIZZLE.

2.4.2.3.1 KNACK

KNACK builds a knowledge base for developing reports. KNACK requires: a sample report, a domain model, and strategies for acquiring knowledge to aid report writing (Klinker, 1988; Klinker, Bentolila, Genetet, Grimes, & McDermot, 1988; McDermot, 1988). The sample report is a skeleton or case example of the desired report. The domain model contains a description of the domain's concepts, terms, and their interdependencies (Klinker, 1988). Strategies define ways to acquire pieces of information for use in a report.

The expert gives KNACK a sample report which is created in a text editor. An example section of a sample report is given in Figure 17, taken from Klinker (1988). Next, the expert inputs the domain model. The domain model consists of the key concepts used in the report (e.g., enclosure). Each concept is described in terms of attribute-value pairs, as seen in Figure 18.



1. Evaluation of EMP Hardness
1.1 Summary of the System Description

The system Communications Unit is designed to resist EMP threat. It consists of a Computer, a Modem, a Radio, and a Motor Generator. Power is supplied from the Motor Generator to the Computer, Modem, and Radio by the Power Cable.

The Computer, Modem, and Radio are protected by a S-280C enclosure. The Motor Generator is protected by a Metal Box enclosure.

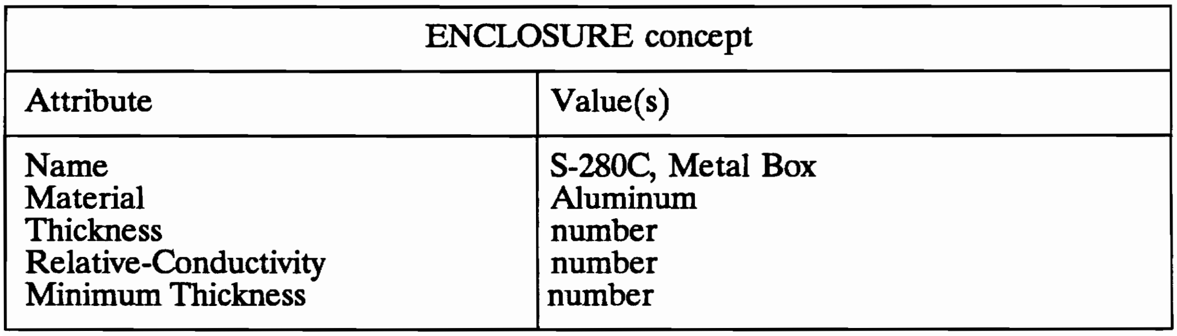
The S-280C enclosure has the following apertures: Window and Cable Entry Panel.

1.2 Shielding Requirements for the S-280C Enclosure

1.2.1 Diffusion through the Skin of the Enclosure

The S-280C enclosure is made of aluminum and is 30 mils thick. Aluminum has a relative conductivity of 0.15mhos/m. A plate of aluminum must be at least 20 mils thick to reduce the diffusion factor to a negligible level. Therefore, the diffusion factor can be neglected.

Figure 17. Example of a sample report for KNACK.
(Taken from Klinker, 1988.)

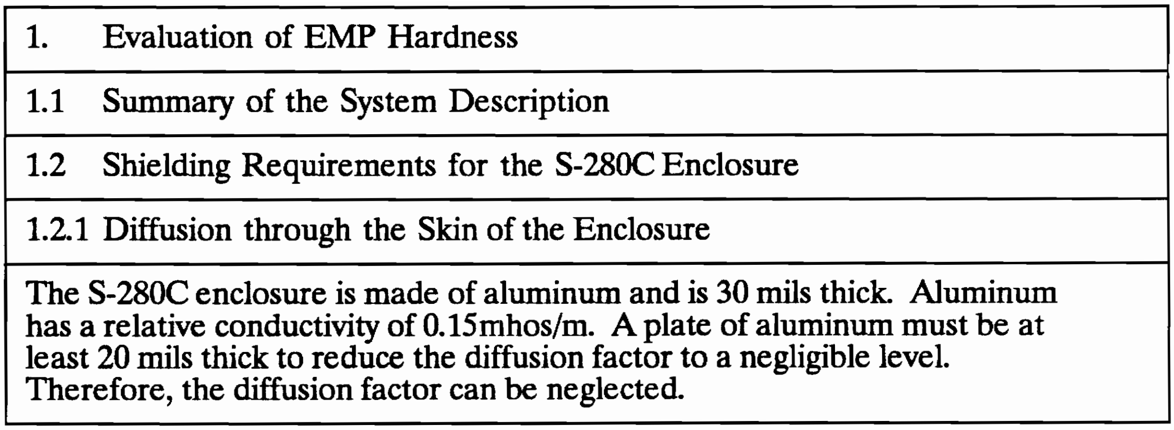


ENCLOSURE concept	
Attribute	Value(s)
Name Material Thickness Relative-Conductivity Minimum Thickness	S-280C, Metal Box Aluminum number number number

Figure 18. Example of a concept definition used in KNACK.
(Taken from Klinker, 1988.)

The relationships between the concepts are also defined by naming the relationship, e.g., comprises or connected with. For example, a system is *comprised* of an enclosure and the enclosure is *comprised* of seams and apertures.

KNACK constructs a skeletal report by fragmenting the sample report. KNACK fragments the sample report by looking for key words and formatting. Key words such as Chapter or Section and formatting such as "1. title" or "1.1 title" are used in the fragmentation. From the example sample report given earlier, the fragments would include those shown in Figure 19.



1. Evaluation of EMP Hardness

1.1 Summary of the System Description

1.2 Shielding Requirements for the S-280C Enclosure

1.2.1 Diffusion through the Skin of the Enclosure

The S-280C enclosure is made of aluminum and is 30 mils thick. Aluminum has a relative conductivity of 0.15mhos/m. A plate of aluminum must be at least 20 mils thick to reduce the diffusion factor to a negligible level. Therefore, the diffusion factor can be neglected.

Figure 19. Example of report fragments made by KNACK.
(Taken from Klinker, 1988.)

Next, KNACK generalizes each fragment. The report is generalized by searching for values of a concept's attribute in the report and replacing the attribute values with a variable name. For example,

> "The S-280C enclosure is made of aluminum and is 30 mils thick. Aluminum has a relative conductivity of 0.15mhos/m. A plate of aluminum must be at least 20 mils thick to reduce the diffusion factor to a negligible level. Therefore, the diffusion factor can be neglected."

would become

> "The <enclosure.name> is made of <enclosure.material> and is <enclosure.thickness> thick. <enclosure.material> has a relative conductivity of <enclosure.conductivity> mhos/m. A plate of <enclosure.material> must be at least <enclosure.minimum-thickness> mils thick to reduce the diffusion factor to a negligible level. Therefore, the diffusion factor can be neglected (Klinker, 1988, p. 145)."

Next, the expert defines strategies to acquire information or values of a concept. Strategies include question, inference, table, menu, graphics, formulas, or database. For example, to elicit how to determine the enclosures of a system KNACK would ask the expert the following:

> **Which strategy can be used to determine the ENCLOSURES of a SYSTEM?**
>
> The strategies are:
> [*question*, **inference, table, menu, graphics, formulas, database, postpone, quit**].

question text	*Please list the enclosures*
possible answers	[*incomplete-set, S-280C, Metal Box*]
default answer	*unknown*

In the above example dialogue, the expert is specifying the "how to get a value" strategy for a system enclosure. The expert is telling KNACK to ask for the enclosure with the question "Please list the enclosures." The expert gives possible responses (e.g., incomplete set, S-280C, or metal box), and, if appropriate, a default answer. The strategies are generalized by replacing specific attribute values defined in the strategy with the variable representing the concept. For example, the above strategy "Please list enclosures" becomes "Please list the enclosures of the <system.name> system."

After the strategies have been defined, the report is interactively checked with the expert. KNACK takes a fragment and replaces the concept variables with its values. The fragment is shown to the expert and the expert makes any changes necessary.

Next, the knowledge base built by the sample report, domain model, and strategies is refined. Two of the heuristics used by KNACK to refine the knowledge base are: 1) each domain model concept must have a strategy for gathering its value and 2) each concept or attribute must be used in a report or strategy.

The output from KNACK is called a WRINGER. A WRINGER is an expert system which acts on the knowledge base created by KNACK. A WRINGER uses the information such as strategies to gather information to write reports.

In summary, KNACK uses the following process: 1) from the domain model and sample report, a generalization of the report is created interactively with the expert; 2) from this generalization the possible values of a concept are placed in the report; 3) strategies to get those possible values are also given; and 4) the knowledge is analyzed for completeness of strategies, concept inclusion, and possible concepts and values. KNACK uses an acquire-and-present problem-solving strategy to search through the report and domain model (McDermot, 1988). The sample report is acquired, generalized, and presented to the expert for revisions.

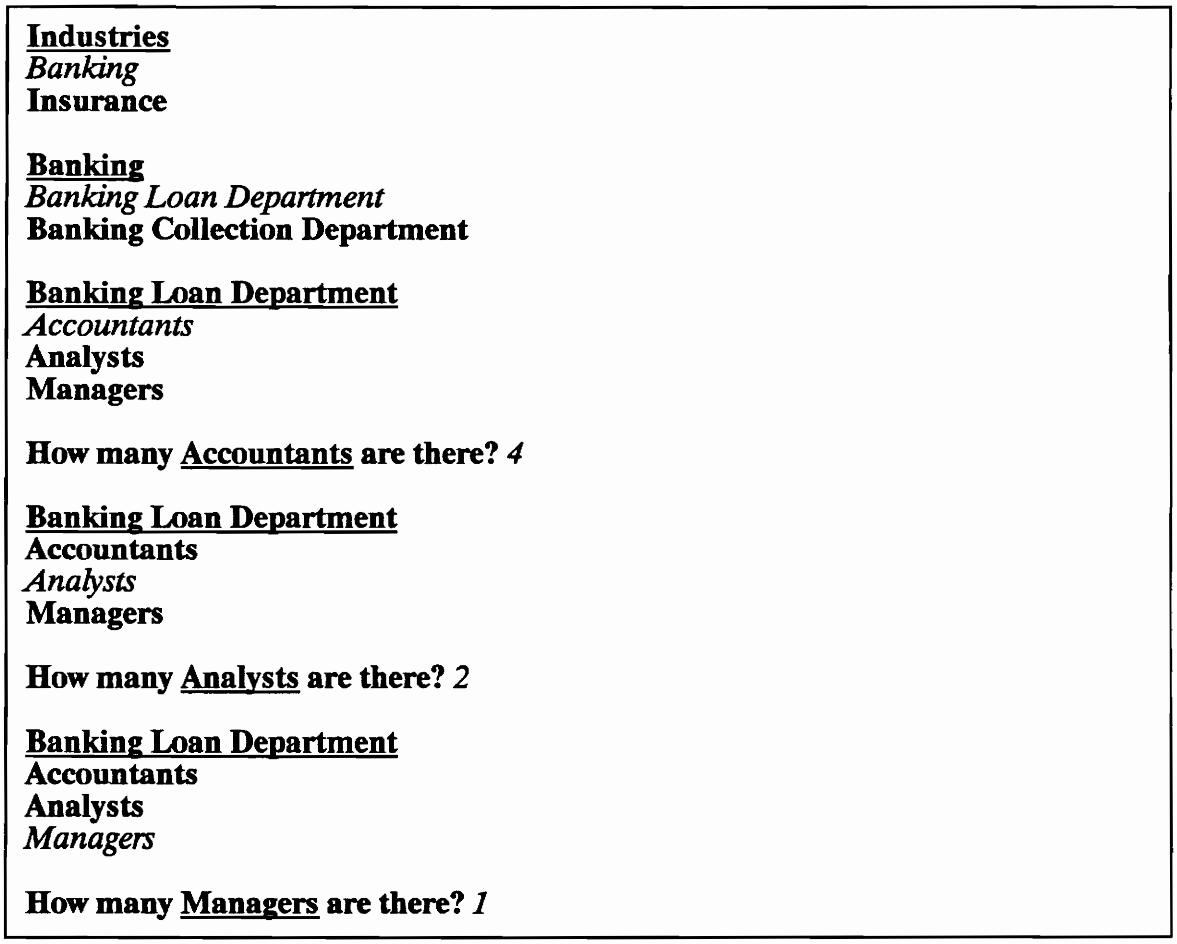
2.4.2.3.2 SIZZLE

SIZZLE uses past cases and extrapolation knowledge to solve a sizing problem (Offutt, 1988). A sizing problem is one of finding the minimum acceptable capacities some system must have, given information about the uses to which that system will be put. SIZZLE solves a problem by acquiring, representing, and applying knowledge to make solution recommendations. SIZZLE extrapolates from a past similar case (i.e., the source) to a target case by taking the source solution and adjusting it to fit the target case. A case is represented as a description-solution pair.

The process begins with the expert inputting the characteristics or uses for the target case. The input or target case description is a set of attributes and values. The expert is presented with a set of prompts for a value of an attribute which describes the case. From the following example shown in Figure 20, the initial attribute is **industries**. Based upon the expert's response, further attribute-value pairs are requested. For the initial attribute, **industries**, the expert has assigned the value *banking*. The expert is then asked for the value of the attribute

department of **banking**. This series of identifying the values for attributes builds the case description used by SIZZLE to find a solution.

From the target case, SIZZLE searches for a source case that matches the target case the closest. The source case is found by matching the attributes and values for each case. The source case with the most similar attribute-value pairs is selected. After the source case is found, SIZZLE adjusts the source's solution to fit the description of the target case. The solution is adjusted by use of rates defined by the expert. The rate encodes the amount of a resource needed by an attribute (e.g., disk space needed per accountant). SIZZLE presents the adapted solution to the expert. Figure 21 shows the source case description and solution SIZZLE used to construct a solution to the target.



Industries
Banking
Insurance

Banking
Banking Loan Department
Banking Collection Department

Banking Loan Department
Accountants
Analysts
Managers

How many Accountants are there? *4*

Banking Loan Department
Accountants
Analysts
Managers

How many Analysts are there? *2*

Banking Loan Department
Accountants
Analysts
Managers

How many Managers are there? *1*

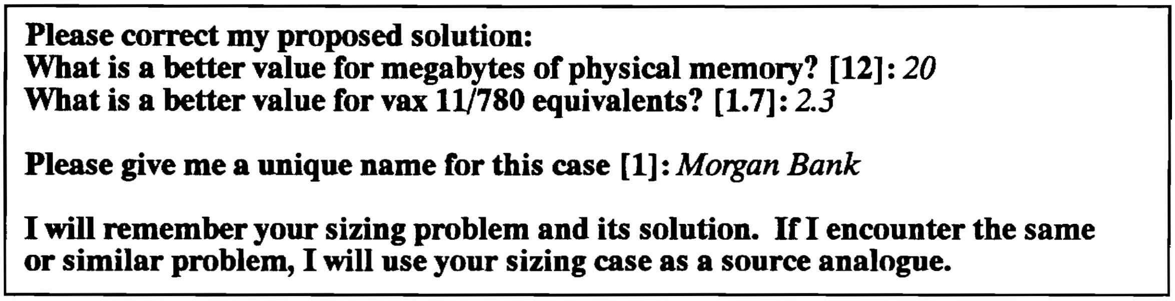
Figure 20. Example target case used by SIZZLE.
(Taken from Offutt, 1988.)

```
Your sizing task reminds me of the Valley Bank Case.

The source sizing solution:
Secretaries  1
Managers     1
Accountants 1

Source Solution:
vax 11/780 equivalents            0.9
megabytes of physical memory    5

Your posed sizing problem:
Accountants 4
Analysts     2
Managers     1

Solution to your sizing problem:
vax 11/780 equivalents            1.7
megabytes of physical memory    12
```

Figure 21. Example source case used and solution constructed by SIZZLE. (Adapted from Offutt, 1988, by presenting a simplified solution.)

After the expert is presented the solution, he or she is to determine if the proposed solution is appropriate. If the solution is accepted, then the session is over. However, if the solution isn't accepted, then the expert adjusts the solution and SIZZLE saves the case description and solution. Figure 22 shows the dialog used when the proposed solution needs to be edited by the expert.

In summary, the expert inputs the problem characteristics to help remind the system of possible past cases to extrapolate from. Once the system is given these characteristics, it searches for a case that can directly solve the problem or can be extrapolated. The user can either accept the solution or modify it. If he or she accepts the solution, then the session ends, otherwise the solution can be modified. The knowledge base is interactively and continuously being constructed and refined during a session. Knowledge added by each case can immediately be used to solve another case.



```
Please correct my proposed solution:
What is a better value for megabytes of physical memory? [12]: 20
What is a better value for vax 11/780 equivalents? [1.7]: 2.3

Please give me a unique name for this case [1]: Morgan Bank

I will remember your sizing problem and its solution.  If I encounter the same
or similar problem, I will use your sizing case as a source analogue.
```

Figure 22. Example of the expert adjusting a solution with SIZZLE.
(Adapted from Offutt, 1988, by presenting a simplified solution.)

2.4.2.4 Iterative Design

Iterative design methods aid the expert in the design process by eliciting, organizing, representing, and refining the gathered design knowledge. SALT is one such example of an iterative design tool.

2.4.2.4.1 SALT

SALT uses a propose-and-revise problem-solving strategy to aid in the design-constraint satisfaction process (Marcus, 1988a; Marcus, 1988b). SALT proposes a solution and then revises it to meet constraints. Constraint satisfaction tasks involve allocating requirements or resources within specified constraints (e.g., scheduling or design).

SALT elicits knowledge about design parameters, constraints, and fixes to violated constraints (McDermot, 1988). The expert is interactively interviewed for the parameters, constraints, and fixes. Parameters are a set of values for variables which define a design alternative. A constraint is a limit on the value of a parameter. A fix is a procedure for altering a parameter when a constraint is violated. SALT elicits, refines the knowledge, and generates rules.

Design parameters are elicited by having the expert fill in a form. A design parameter is defined by specifying its name, procedure for getting a value, precondition to using the parameter, and justification. Procedures include formula calculation and database lookup. The precondition specifies when a parameter should be used. The justification is text to comment on the parameter. Figure 23 is an example of a formula calculation procedure.

1. Name: *Car-Jamb-Return*
2. Precondition: *Door-Opening = Center*
3. Procedure: *Calculation*
4. Formula: *[Platform-width - Opening-Width] /2*
5. Justification: *Center-Opening doors look best when centered on the platform*

Figure 23. Example of a parameter definition in SALT.
(Taken from Marcus, 1988a.)

A constraint for a parameter is defined the same way a parameter is, by filling in a form. When defining a constraint, the expert identifies: the parameter to be constrained, the type of constraint, name of the constraint, precondition, procedure, and justification. The name, precondition, and justification represent the same things as in the parameter definition form. The type of constraint includes such limits as the maximum, minimum, or member of a chosen set. The procedure determines how the value to be associated with the parameter is to be achieved. In Figure 24, the maximum (constraint type) value of Car-Jamb-Return (constrained value) is Panel-Width multiplied by Stringer-Quantity (formula procedure) when Door-Opening equals Side (precondition).

Next, fixes or remedies to violated constraints are elicited from the expert. A constraint definition includes the name of the constraint violated, the parameter or value to change, how to change the value, the preference for the change, and the reason for preference. The constraint violated is the name of the constraint the fix is attempting to correct. The value to change specifies the parameter and value the fix is going to change. How to change the parameter value specifies the procedure used to change the value. The preference rating is used to distinguish between alternative fixes to the same violated constraint. The lower the value of the preference rating, the more it is preferred. In Figure 25, the parameter Stringer-Quantity is increased by a value of 1 every time the Maximum-Car-Jamb-Return constraint is violated, if this fix is preferred over other fixes.

1. Constrained value:	*Car-Jamb-Return*
2. Constraint type:	*Maximum*
3. Constraint name:	*Maximum-Car-Jamb-Return*
4. Precondition:	*Door-Opening = Side*
5. Procedure:	*Calculation*
6. Formula:	*Panel-Width * Stringer-Quantity*
7. Justification:	*This procedure is taken from installation manual I, p. 12b.*

Figure 24. Example of a constraint definition in SALT.
(Taken from Marcus, 1988a.)

1. Violated Constraint:	*Maximum-Car-Jamb-Return*
2. Value to change:	*Stringer-Quantity*
3. Change type:	*Increase*
4. Step type:	*By-Step*
5. Step size:	*1*
6. Preference rating:	*4*
7. Reason for preference:	*Changes minor equipment sizing*

Figure 25. Example of a fix definition in SALT.
(Taken from Marcus, 1988a.)

When the expert is finished defining procedures, constraints, and fixes, the system refines the knowledge base for completeness, compilability, and convergence. Completeness is assessed by making sure every parameter has a procedure and constraint, and every constraint has a fix. Compilability is assessed by checking for overlaps in preconditions that produce counter-productive rules, the ability to find at least one path to a solution, and checking for loops or self referencing procedures. Convergence is assessed by checking the interaction of fixes to ensure violated constraints can be fixed so a solution can be reached. The purpose is to make sure the system doesn't stop because the interactions won't allow a solution to be designed.

The knowledge base can be used to aid designers in developing a design description. This design description is represented using a dependency network with a node representing a parameter and with links for the relations (e.g., contributes to, constrains, suggests revision of) between nodes. An example of a model built by SALT is given in Figure 26.

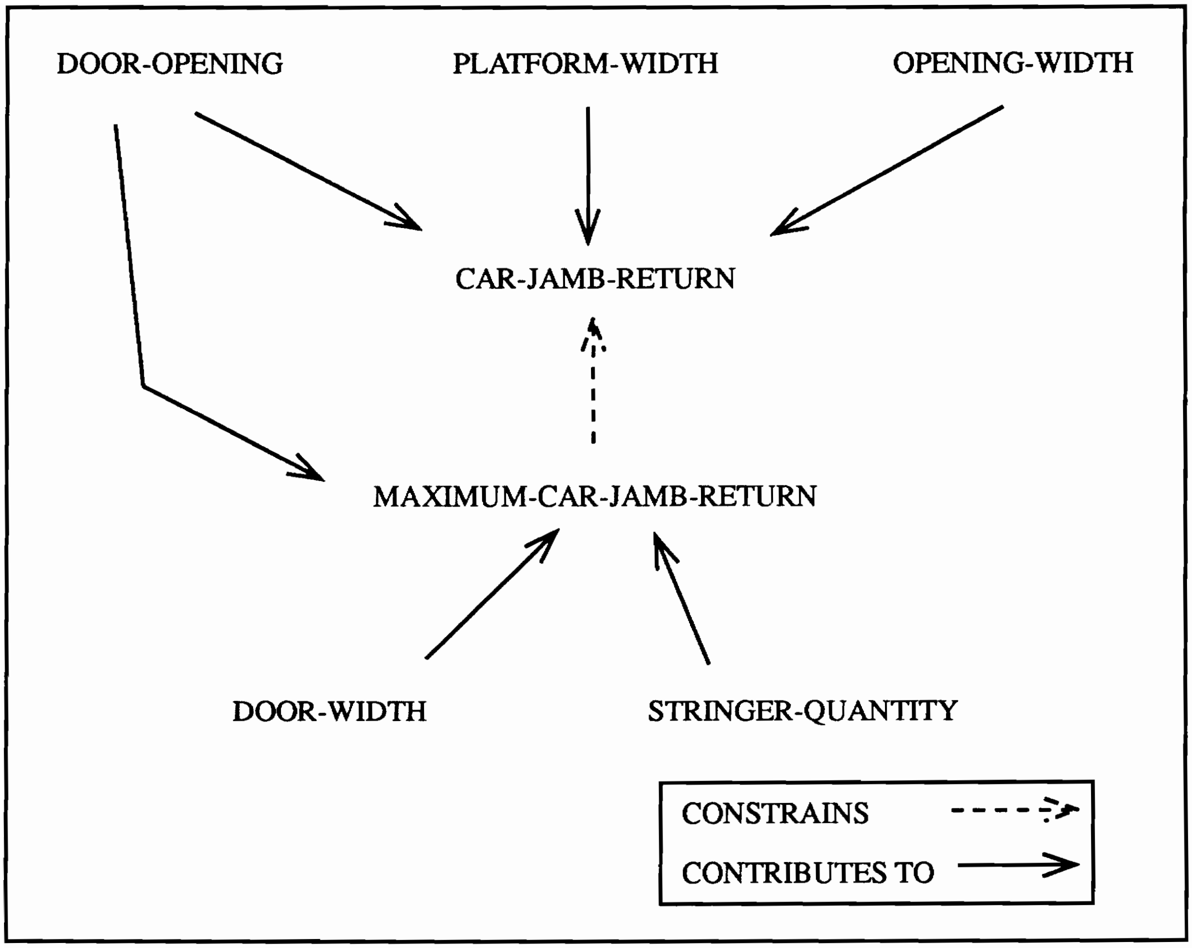


Figure 26. Example of a design model built by SALT.
(Taken from Marcus, 1988a.)

2.4.2.5 Cover and Differentiate

Cover and differentiate is a problem-solving process in which the expert specifies: 1) candidates to cover a potential problem solution and 2) information used to differentiate the candidates (Eshelman, 1988). MOLE is an example of a machine-aided method which uses a cover-and-differentiate process.

2.4.2.5.1 MOLE

MOLE is an automated knowledge-acquisition tool that employs the cover-and-differentiate strategy to diagnose problems (Eshelman, 1988). MOLE is the improvement of an earlier tool, MORE, which followed the same strategy (Kahn, 1988). MOLE interviews the expert for complaints, explanations for the complaints, and knowledge to differentiate the explanations (Eshelman, Ehret, McDermot, & Tan, 1988; McDermot, 1988). The knowledge is refined dynamically by evaluating system performance in solving cases.

The following steps are used by MOLE to implement the knowledge-acquisition process. The process and examples are taken from Eshelman (1988).

1. The expert inputs a list of possible complaints or symptoms to be explained. Attribute-value pairs are used to describe a complaint.

   List possible complaints or symptoms that might need to be diagnosed:
   Complaint:
   >> *loss-in-gas*
   Status: *new*
   Method: *ask*
   Default Value: *none*

Complaint:
>> *high-fly-flash-flow*
Status: *new*
Method: *ask*
Default Value: *none*

2. MOLE asks for covering knowledge: states or events which explain/cover the symptoms from 1, this is done until every symptom has an explanation.

List possible explanations for <u>loss-in-gas</u>:

Possible explanation for <u>loss-in-gas</u>:

low-heat-transfer
Status: *new*
Method: *infer*
Default Value: *none*

excess-air high
Status: *new*
Method: *infer*
Default Value: *none*

3. MOLE asks for covering knowledge for an explanation. The explanations from step 2 become the next level of "complaints" and explanations for these are elicited by returning to step 2. This process continues until all complaints are explained and the expert says a complaint doesn't need to be explained. The following represents how this occurs.

List possible explanations for <u>low-heat-transfer</u>:

Possible explanation for <u>low-heat-transfer</u>:

misbalance-of-convection
Status: *new*
Method: *infer*
Default Value: *none*

low-radiation
Status: *new*
Method: *infer*
Default Value: *none*

4. MOLE asks for differentiating knowledge. Differentiating knowledge is knowledge which distinguishes candidate explanations for a symptom, that is, what explanation works under a given condition.

   Indicate which of the following tends to rule out excess-air high as the explanation for loss-in-gas?
   1. *high-fly-flash-flow*
   2. *dark-ash*

5. MOLE asks for preferences of explanations. In the following example, low-flame-temperature is preferred over the other two explanations.

   Given the presence of small-red-flame, rank the following explanations of low-radiation:

   low-flame-temperature [1]
   low-moisture-content [1] *2*
   large-particles [1] *2*

6. MOLE looks for weaknesses in the knowledge base (i.e., refines) dynamically by having the expert give a test case and correct diagnosis. MOLE solves the case, if the diagnosis is wrong, then MOLE tries to find and correct the error in the knowledge base

   Dark-ash is explained by low-grindability-relative-to-setting.
   Is this correct?
   >> *no*
   dark-ash is explained as follows:
   1. dark-ash is explained by large-particles
   2. large-particles is explained by low-grindability-relative-to-setting
   Indicate the first incorrect step:
   >> *1*

   What is the correct explanation for dark-ash:
   1. large-particles
   2. excess-air high
   >> *none*

   In order to explain dark-ash some hypothesis must be accepted.

   Enter the explanation for dark-ash:

   excess-air low

   excess-air low
   Status: *old*
   Method: *infer*
   Default Value: *none*

In summary, the overall process involves having the expert specify: 1) the list of symptoms to be explained, 2) explanations for "symptoms" and "symptom-explanations", and 3) when an explanation is preferred. The elicitation and refinement of the knowledge is interactive, while the organization and representation are implicitly handled by the machine.

2.4.3 Machine-Learning Methods

The third set of knowledge-acquisition methods are the machine-learning methods. The machine does the majority of the knowledge-acquisition process, it organizes, compiles, and refines automatically after it is given data. Very little human interaction is required in comparison to manual and machine-aided methods. One drawback of these methods is they are subject to noise and fluctuations in the data. (Kodratoff, Manago, & Blythe, 1988; Pettit & Pettit, 1988). The four primary paradigms employed by machine-learning systems have been classified by Carbonell (1990) as the inductive, the analytic, the genetic, and the connectionist paradigms and are adopted for this review. A more elaborate classification however has been offered by Michalski's (1991) Inferential Learning Theory.

2.4.3.1 The Inductive Paradigm

Induction generates rules by synthesizing a general concept description from a sequence of positive and negative instances of the concept. The purpose is to build a concept description so the positive examples can be later classified and the negative examples not classified. The basic induction process starts with a training set of examples which are concept instances and their descriptions (Gennari, Langley, & Fisher, 1990; Trippi & Turban, 1990). Table 3 provides an example of a data set used for inducing a concept. Induction can be incremental, constructing a description one example at a time, or batch, when a description is built by all examples at once. ID3 is an example of a batch-induction system (Quinlan, 1983).

Table 3
Example Induction Data Set
(Taken from Quinlan, 1983.)

	Attributes		
Class	**Height**	**Hair Color**	**Eye Color**
a	short	blond	blue
b	tall	blond	brown
a	tall	red	blue
b	short	dark	blue
b	tall	dark	blue
a	tall	blond	blue
b	tall	dark	brown
b	short	blond	brown

Clusterings that group the instances, definitions of each cluster, and a hierarchical organization of clusters are sought by the algorithm (Gennari, Langley, & Fisher, 1990). A decision tree representing the hierarchy of objects in the concept results from using an induction system (Broner, King, & Nevo, 1990; Quinlan, 1988).

ID3 constructs a decision tree by determining which attribute provides the most information, that is, has the biggest impact on distinguishing between the different instances. Table 4 gives the calculations used by ID3 to construct a decision tree.

Table 4
Formulas Used for ID3 Induction Method
(Abstracted from Quinlan, 1983.)

Value	Formula	What value means or stands for
a_i	N/A	attribute a_i (e.g., height, hair color, eye color)
$a_i v_j$	N/A	attribute value $_j$ of attribute a_i (e.g., short and tall of height; blond, red, and dark of hair color; and blue and brown of eye color)
p_x	# in class x total number of examples	proportion of examples in class x (e.g., 3/8 of the examples are in class a and 5/8 of the examples are in class b)
$p_x a_i v_j$	# in class x for an attribute-value pair total number of examples with attribute-value pair x = the total number of classes (e.g., 1 to 2 for a or b); i = the number of attributes (e.g., 1 to 3 for height, hair color, eye color); j = the number of attribute values for attribute i, (e.g., 1 to 2, for values short and tall of attribute height)	proportion of examples in class x (e.g., a or b) for attribute i value j (e.g., for short height p_a= 1/3 and p_b=2/3; for tall height p_a= 2/5 and p_b=3/5)
M(C)	-sum(p_x * $\log_2 p_x$)	the information content of the classes a and b
IC($a_i v_j$)	- sum($p_x a_i v_j$ * $\log_2 p_x a_i v_j$)	information content for each given attribute-value pair
B(C,a_i)	sum($p_x a_i v_i$ * IC($a_i v_j$))	expected information content for each attribute
IG(a_i)	B(C,a_i) - M(C)	information gained by using attribute a_i of the set of C objects as a test

The process for developing a decision tree used by induction is to calculate the values given in Table 4 and choose the attribute with the greatest information gained value to be the next attribute test. The first attribute is the root. Each subsequently-selected attribute is the test for the subsequent level of the tree. This process continues until all attributes are accounted for or all examples have been classified.

Using the formulas given in Table 4 and the data in Table 3, the following values are computed. There are two classes, a and b. The proportion, p_x, for each class is p_a= 3/8 and p_b=5/8. M(C) is equal to 0.9544 = - (3/8 * $\log_2$ 3/8) - (5/8 * $\log_2$ 5/8). The information content for the attribute values is given in Table 5.

Next, the information gained by using a given attribute is calculated based upon the results of the information content. The results of the information gained for the example given in Table 3 are given in Table 6.

Based on the results of Table 6, hair color is the first attribute used because it has the highest information gained. Next, eye color is used. Attributes are chosen to be a test until all examples are clustered in the hierarchy. Figure 27 shows the resulting hierarchy using the ID3 induction algorithm on the data set given in Table 3. The nodes represent a test of an attribute and the branches the possible results or values of the test. For example from Figure 27, the top node tests the hair attribute. If the instance has blond hair, then it is traced to the right branch or side of the tree.

Table 5
Information Content Values for Induction Example

Attribute Value	**# of Examples with Attribute Value**	p_a	p_b	**Information Content** $IC(a_iv_j) = -\mathrm{sum}(p_xa_iv_j * \log_2 p_xa_iv_j)$
short	3	1/3	2/3	0.918 = -(1/3 * log$_2$ 1/3) - (2/3 * log$_2$ 2/3)
tall	5	2/5	3/5	0.971 = -(2/5 * log$_2$ 2/5) - (3/5 * log$_2$ 3/5)
blond	4	2/4	2/4	1.0 = -(2/4 * log$_2$ 2/4) - (2/4 * log$_2$ 2/4)
red	1	1/1	0/1	0.0 = -(1/1 * log$_2$ 1/1) - (0/1 * log$_2$ 0/1)
dark	3	0/3	3/3	0.0 = -(0/3 * log$_2$ 0/3) - (3/3 * log$_2$ 3/3)
blue	5	3/5	2/5	0.971 = -(3/5 * log$_2$ 3/5) - (2/5 * log$_2$ 2/5)
brown	3	0/3	3/3	0.0 = -(0/3 * log$_2$ 0/3) - (3/3 * log$_2$ 3/3)

Table 6
Information Gained Values for Induction Example

Attribute	**B(C,a$_i$)**	**IG**
Height	.951 = 3/8(.918) + 5/8(.971)	.0034 = .9544 - .951
Hair Color	.5 = 4/8(1) + 1/8(0) + 3/8(0)	.4544 = .9544 - .5
Eye Color	.606 = 5/8(.971) + 3/8(0)	.3487 = .9544 - .606

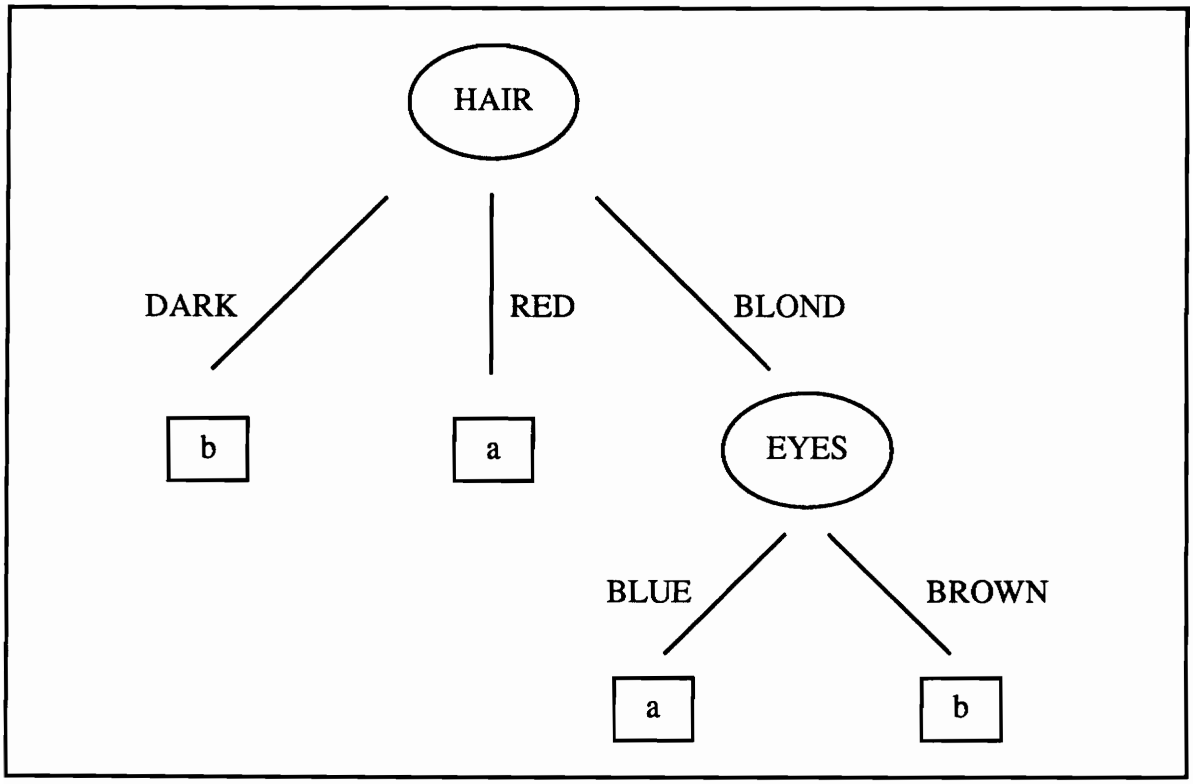


Figure 27. Induction hierarchy created by ID3.

Induction is used primarily for classification due to the decision-tree hierarchy created and is limited by the need to interpret the results of the decision tree (Cendrowska, 1988; Quinlan, 1988). The quality of the results depends upon the order of presented examples and number of distinguishing examples. Induction algorithms differ in the ways the instances are sorted and the decision tree is constructed.

2.4.3.2 Genetic Algorithms Paradigm and Classifier Systems

Genetic algorithms model the biological behavior of genes and their chromosomes and are used to increase the performance of classifier systems by discovering rules. Classifier systems are most useful when there is a continuous stream of environmental data, need for real time action, inexactly defined goals, and little reinforcement (Holland, Holyoak, Nisbett, & Thagard, 1987). Classifier systems have three levels of activities: 1) a performance system that interacts with the environment; 2) a credit assignment system used to determine which rules are effective; and 3) a rule discovery system. These components and their relationships are shown in Figure 28.

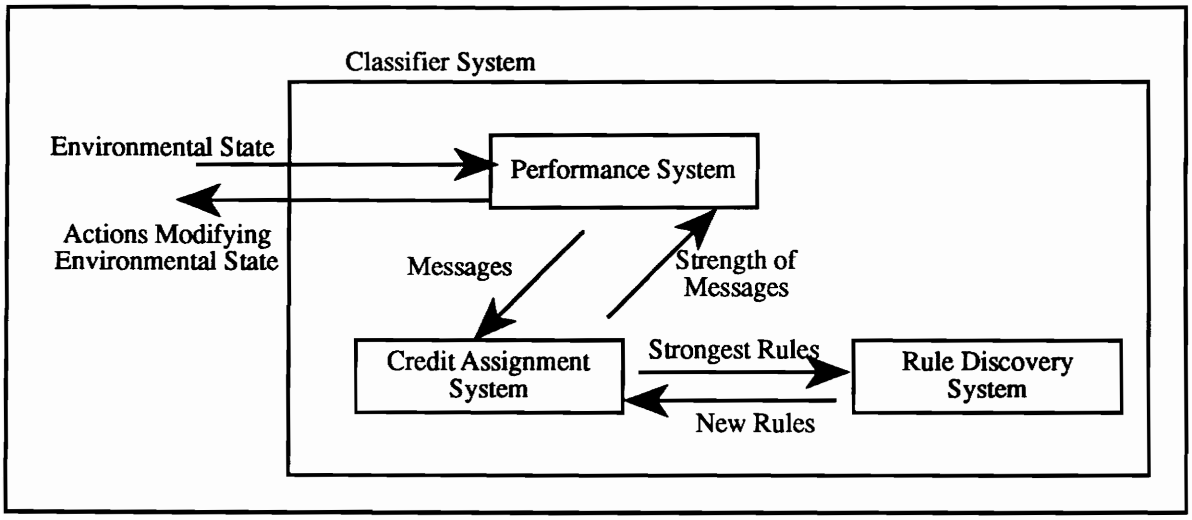


Figure 28. Components of a genetic algorithm system.
(Abstracted from Holland, Holyoak, Nisbett, & Thagard, 1987.)

The performance system has four basic parts: 1) an input interface, that translates the current environmental state into a message; 2) classifiers, which are the rules used by the system to process messages; 3) a message list containing all current messages; and 4) an output interface, which translates some messages into actions which modify the environmental state. The performance system receives the messages from the input interface, processes the messages, and produces a message for the output interface. Messages are processed by the following steps: 1) all messages are compared to all conditions of classifiers and matches are recorded, 2) for each match, messages are posted as specified in the action part of the matching classifiers replacing the old message list, and 3) the messages on the new messages list are translated to requirements of the output interface. Messages are posted based on the strength of their associated classifier as determined by the credit assignment system.

The credit assignment system uses the bucket brigade algorithm to assign a strength to each classifier. The strength represents the usefulness of a classifier to the system. This strength is used to determine which messages get posted to the message list for the next recursive step. The strength is adjusted every time-cycle of the process.

The genetic component takes the strongest classifiers given by the bucket brigade to make new, more-effective rules. The rules are represented by vectors of detectors and reactions in the form of 0 (no, otherwise), 1(yes), or #(don't care) (e.g., (detectors/reactions) or (1,1,0,0,1,#,# / 0,0,1,1)). The detectors are the conditions and the reactions are the actions of a production rule form. Detectors are counted from the right (e.g. $d_6d_5d_4d_3d_2d_1$). Table 7 is an example of detectors taken from Booker, Goldberg, and Holland (1990) to simply show the relationship between a detector, its value, and a message from the input interface.

Table 7
Example Detectors and Their Values for a Genetic Algorithm System
(Taken from Booker, Goldberg, & Holland, 1990.)

Detector	**Value & Its Meaning**
d_1	1 = object is moving; 0 = otherwise
(d_2,d_3)	(0,0) = object centered in vision field
(d_2,d_3)	(1,0) = object left of center in vision field
(d_2,d_3)	(0,1) = object right of center in vision field
d_4	1 = system adjacent to the object; 0 = otherwise
d_5	1 = object is large; 0 = otherwise
d_6	1 = object is striped; 0=otherwise

From Table 7, when the message is received that a large, moving, striped object is adjacent and right of center, the input interface forms a detector, which is 111011. The detector is matched by the performance system against the classifiers in the system. For each match of the detector vector, the reactor vector, of the classifiers is posted to the message list. A reactor might be to move to the left and avoid contact with the object.

New rules are created by applying genetic operators such as cross over, mutation, and inversion to the existing strong rules (Pettit & Pettit, 1988). Cross over involves interchanging conditions and actions between rules, **1001/111** and 0101/010 become **1001/**010 and 0101/**111**. Mutation occurs by changing a random detector value, 1001/000 could become 1101/000. Inversion is the swapping of random detector values, 1110/010 could become 1011/010. The weakest rules are replaced by the new rules created with the genetic algorithms. Refinement is continually performed by a rule's strength being computed by the bucket brigade component.

2.4.3.3 Analytic Paradigm

Analytic machine-learning methods improve the efficiency of a system by using past problem-solving examples and/or domain theory (Carbonell, 1990). Analytic methods include: Explanation-Based Learning, Learning by Derivational Analogy, and Case-Based Learning.

2.4.3.3.1 Explanation-based learning

Explanation-based learning (EBL) uses domain theory and a concept example to acquire search control rules from a problem solving trace to improve problem solving (Minton, Carbonell, Etzioni, Knoblock, & Kuokka, 1987). Minton, Carbonell, Etzioni, Knoblock, and Kuokka (1987) define the inputs of an EBL system as

> **Target concept**: A concept to be learned
> **Training example**: An example of the target concept
> **Domain theory**: A set of rules and facts to be used in explaining how the training example is an example of the target concept
> **Operationality criterion**: A predicate over descriptions, specifying the form in which the learned description must be expressed.

The domain theory is organized as "<fact> IF <conditions>." The fact is true if the conditions are met. The fact is represented as a predicate and the conditions as one or more conjunctive predicates. The operationality criteria is the set of predicates specifying when the learning process should stop (Adeli & Yeh, 1990). A predicate is a rule for establishing a fact or relationship. A predicate is true if the fact or relationship described by the predicate is true, otherwise it is false. For example, if julie is a girl then male(julie) would be false. Figure 29 provides an example of the inputs. From Figure 29, the predicate is_elder_brother_of (B,A) is true if the conjunctive of conditions or predicates [have_same_parent(B,A), and male(B), and older_than(B,A)] is true.

Target Concept
Is_elder_brother_of(_,_)

Training Example
1. is_elder_brother_of(john, tom)
2. is_parent_of(dan, john)
3. is_parent_of(dan, tom)
4. born_in(john, 1960)
5. born_in(tom, 1970)
6. male(john)
7. has_brown_hair(john)

Domain Theory
1. is-elder-brother-of(B, A) if
    have_same_parent(B, A), and male(B), and older_than(B, A).
2. is_elder_sister_of(B, A) if
    have_same_parent(B, A), and female(B), and older_than(B, A).
3. have_same_parent(M, N) if
    is_parent_of(X, M), and is_parent_of(Y, N), and same_person(X, Y).
4. older_than(X, Y) if
    born_earlier_than(X, Y).
5. born_earlier_than(U, V) if
    born_in(U, Year1), and born_in(V, Year2), and less(Year1, Year2).

Operationality criterion
1. born_in(_,_)
2. is_parent_of(_,_)
3. less(_,_)
4. same_person(_,_)
5. male(_)
6. female(_)

Figure 29. Example inputs to an EBL system.
(Taken from Adeli & Yeh, 1990.)

EBL is a two-step process: 1) explaining and 2) generalizing (Adeli & Yeh, 1990). The explanation stage involves constructing a proof tree. The generalization stage is done to generate the final output or operational description.

The explanation stage constructs an explanation or proof that the target concept is valid (Minton, Carbonell, Etzioni, Knoblock, & Kuokka, 1987). An explanation or proof tree is built by using the domain theory necessary to prove the target concept. The domain theory is searched for the target concept or predicate is_elder_brother_of(_,_). Next, the conditions necessary to proving the target concept predicate are found. For example, from the domain theory to prove is_elder_brother_of(john, tom), the following must first be proved: have_same_parent(john, tom), and male(john), and older_than(john, tom). These conditions represent the second level of the tree found in Figure 30. Each condition needs to be proved true. From the domain theory, to prove have_same_parent(john, tom), the following must first be proved: is_parent_of(dan, john), is_parent_of(dan, tom), and same_person(dan, dan). To prove older_than(john, tom), born_earlier_than(john, tom) must be proved true. Next, born_earlier_than(john, tom) is proven by born_in(john, 1960), born_in(tom, 1970), and 1960<1970. This process continues with each of the conditions for each subsequent proof as long as each predicate can be further proven; that is, the predicate has conditions to be proven before the predicate can be accepted as true. The final proof tree built in the explanation stage is given in Figure 30.

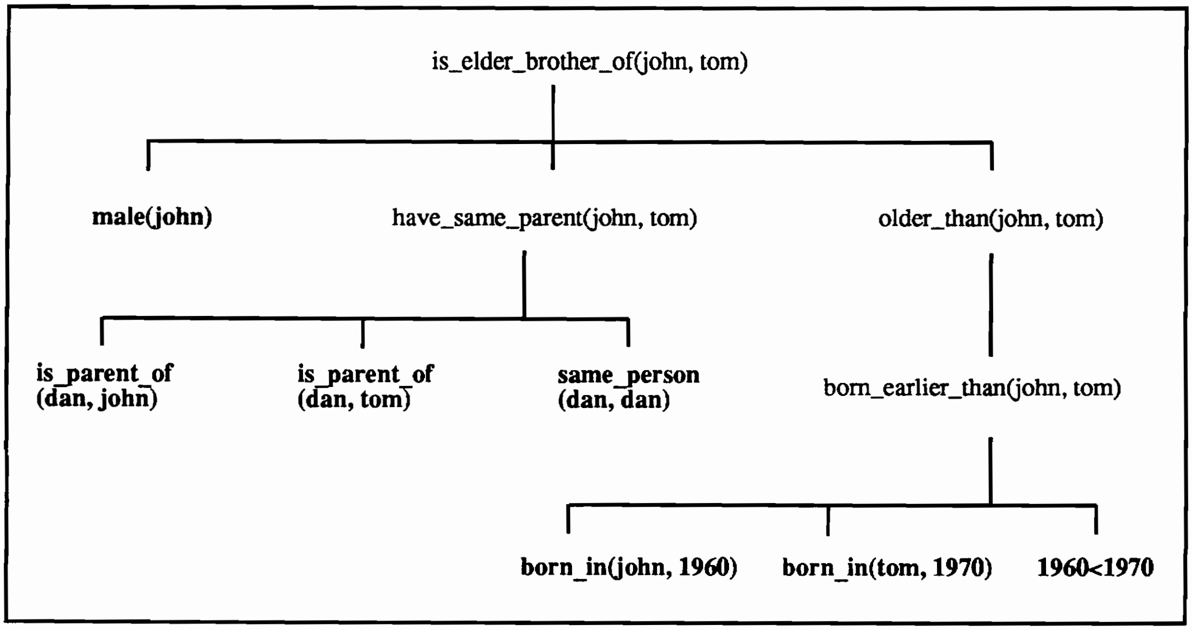


Figure 30. EBL proof tree for example given in Figure 29.
(Taken from Adeli & Yeh, 1990.)

The first step to generalization is variable substitution. For example, from the tree constructed in Figure 30, john would be replaced by X, tom by Y, 1960 by year1, 1970 by year2, and dan by P1/P2. The second and final step of generalization is to take the leaf nodes of the tree shown in Figure 30 and combine them to form the operational description. For the example given in Figures 29 and 30, the operational description is:

```
is_elder_brother_of(X,Y) if
                    male(x), and
                    is_parent_of(P1,X) and born_in(X,Year1) and
                    is_parent_of(P2,Y) and born_in(Y,Year2) and
                    same_person(P1,P2) and less(Year1,Year2).
```

The purpose of EBL is to find the important characteristics of an event (Pazzani, 1988). Explanation-based learning is limited by the ability to accurately describe the domain and example in the same predicates.

2.4.3.3.2 Learning by Derivational Analogy

Derivational analogy uses top-down decomposition to conduct a new plan based upon previous designs. A design plan is "replayed" to solve the new plan by selecting and adapting the old plan to fit the new. A design plan is an executable record of decisions made in the course of solving the original design problem (Mostow, 1990) and is represented by goals, preconditions, justifications, alternatives, and selection criteria.

The process begins with "recognition" of a candidate design plan which is close to the target. The source is "adapted" to the target. The adaptation is then "evaluated" to validate the results. Finally, the design plan is "consolidated" to integrate the results into memory (Bhansali & Harandi, 1990). Figure 31 provides a conceptual framework for derivational analogy.

Recognition is the process of retrieving a source plan which is "close" to the target plan. Bhansali and Harandi (1990) use the following heuristics in their system for recognition:

- the target and source are of the same functional class
- the inputs and outputs are the same or similar
- structure of the processes are the same, (e.g., recursive)
- arguments are of the same type.

Plan adaptation involves identifying violated constraints, deciding which steps to modify, changing the plan's steps to fit the new problem, and propagating the results to identify any other violated constraints (Mostow, 1990). The purpose of this step is to use an adapted version of the source plan to solve the target plan.

Evaluation of the adapted plan is done to validate the results of the adaptation stage. Mostow (1990) provides four overall criteria to validate an adapted plan: executable, correct, successful, and desirable. A plan is executable if it is syntactically correct. A plan is correct if it meets the specification given in the target plan. Success of a plan is judged by the plans ability to achieve its original purpose. Finally, a plan is desirable if it satisfies the criteria used in selecting it in the recognition stage. If any of these four criteria are not met, then the adapted plan is re-adapted or a new source plan is selected.

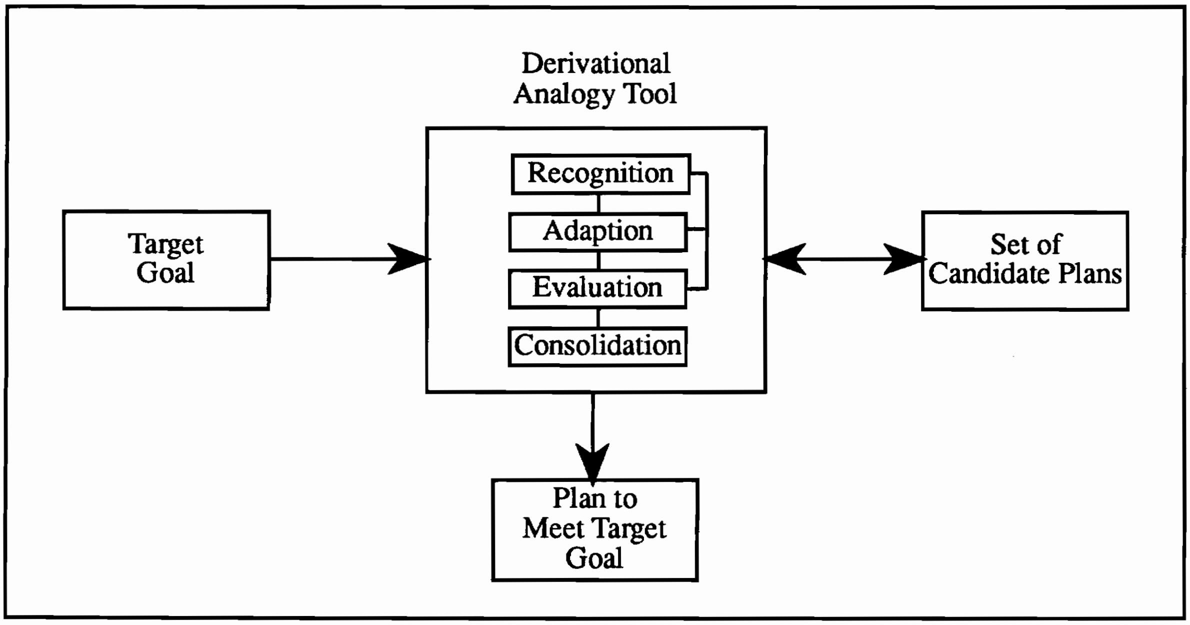


Figure 31. Derivational analogy conceptual framework.
(Abstracted from Mostow, 1990; Bhansali & Harandi, 1990.)

The recognition, adaptation, and evaluation stages recursively occur on the main goal and subgoals of the target and source plans. Consolidation is done to save the adapted plan for future use. The results of the adaptation are integrated into the system's memory to be used on other plans. Derivational analogy assumes "good" previous plans exist to use and learn from. Consequently, the initial cases need to be developed.

2.4.3.3.3 Case-Based

Case-based methods explain why previous knowledge or expectations failed to apply and correct failed expectations (Schank, 1986; Schank & Leake, 1990). The case-based process consists of anomaly detection, search, accepting, tweaking, and integration. An anomaly is something that is abnormal or out of the ordinary, a deviation from the expected or common rule, a failed expectation. Previous cases or explanations are represented using explanation patterns (XP). XPs have 6 components:

1) a representation of the anomaly that the pattern explains,
2) a set of states of the world under which the pattern is likely to be a valid explanation,
3) a set of states of the world under which the pattern is likely to be useful, even if it isn't immediately applicable,
4) a pattern of beliefs, with relationships between them, that show why the event being explained might have been explained,
5) a proverb or rule of thumb that summarizes the situation for use in planning,
6) a set of prior episodes that have been explained by the pattern (Schank & Leake, 1990).

Table 8 is an example of an XP.

Table 8
Example of an Explanation Pattern Used in Case-Based Learning
(Taken from Schank & Leake, 1990.)

Component	Example Value
Anomaly	untimely death
States of the world XP is valid	untimely death or death heavily insured with relatives who didn't love him or beneficiary is suspicious character
States of the world XP is useful	deceased was rich; relatives who didn't love him or beneficiary is suspicious character
Pattern of beliefs	- beneficiary dislikes policy holder - dislike makes beneficiary want to harm policy-holder beneficiary has goal to get a lot of money - inheriting is a plan for getting inheritance - insurance means that inheritance will include a lot of money - inheriting requires that the policy-holder dies beneficiary kills the insured to harm him and to get money
Rule of thumb	a good way to get rich and get rid of someone you don't like at the same time
Prior episodes explained by XP	deaths seen in movies, mafia killings

The following is a five-step process used in SWALE, a case-based program:

1) Anomaly Detection

Anomaly detection occurs when the system attempts to place a new fact into its memory and the input doesn't fit with existing knowledge. The input won't integrate with existing knowledge because: it doesn't satisfy an existing expectation, it can't be accepted in terms of known patterns, other circumstances do not make the fact more likely, or its role-fillers aren't reasonable. Existing XPs can not directly explain each fact of the input.

The anomaly is classified or indexed according to its type; for example, the anomaly could be role-filler of wrong category, premature event, planning problems, or novel causal connection. This anomaly index describes the type of anomaly detected and is used in subsequent steps.

2) XP Search

Existing explanations are searched to use as possible explanations for the anomaly detected in step 1. Three searches are used: routine, unusual features, and folkloric explanation. A routine search uses the type of anomaly found in step 1 to search existing XPs. If no XP can be found based on type, then the XPs are searched based on unusual or extraordinary features of the input case. Finally, SWALE attempts to find a folkloric explanation which might explain the case. Folkloric explanations are "unbelievable" explanations classified only by the type of event to be explained.

3) XP Accepting

This step determines if the XP retrieved from step 2 is acceptable to the case being explained. An explanation of an expectation failure must show a flaw in the reasoning that led to the expectation. An XP is accepted if it is relevant, believable, and provides the level of detail needed. An XP is relevant if it answers or addresses the question raised by the anomaly detection. An XP is believable if it fits into the understanding of the world known by SWALE, the XP could be confirmed, assumed, or unacceptable. The level of detail for an XP is evaluated based upon the need for the level of detail.

4) XP Tweaking

Tweaking is done to adapt XPs that are inapplicable to current anomalies. Tweaking is done by matching the anomaly index to a strategy. Strategies attempt to create new explanations by finding XPs that don't have the problem making the original explanation to be unacceptable. Four strategies are: abandon belief, substitute alternate theme, substitute anticipation, and find connecting XP (Kass, Leake, & Owens, 1986). The abandon-belief strategy eliminates the problematic belief because the belief is irrelevant. The substitute-alternate-theme strategy replaces the action described in the anomaly by the possible action described in XPs. The substitute-anticipation strategy is used when the anomaly is detected because the anomaly event occurred before it was supposed to. This strategy substitutes the belief that the role player was thinking about the event and this thought caused the anomaly. The find-connecting-XP strategy attempts to find a causal chain between beliefs that appear to be unrelated. New explanations are then checked to see if they are acceptable.

5) XP Integration

A new XP is integrated into memory by generalization and indexing. An XP is generalized by substituting general purpose variables (e.g., x and y) in the chain of XPs that explained the anomaly. Indexing of an XP is done by determining the category of anomaly explained (type and major role player) and the conditions under which the anomaly occurred.

This paragraph summarizes the process used in case-based machine learning. Anomaly detection occurs when an expectation consistent with existing memory fails. Explanation patterns are searched for an explanation which accounts for the failed expectation. An attempt to apply the explanation pattern is made. If it is successful with or without revision, then the memory is updated with the new explanation pattern. However, if the pattern was not accepted, then strategies are applied to revise a potential explanation pattern to fit the anomaly. Integration occurs when a new explanation pattern is accepted (Schank & Leake, 1990). Figure 32 provides the top-level process-flow diagram for cased-based machine learning.

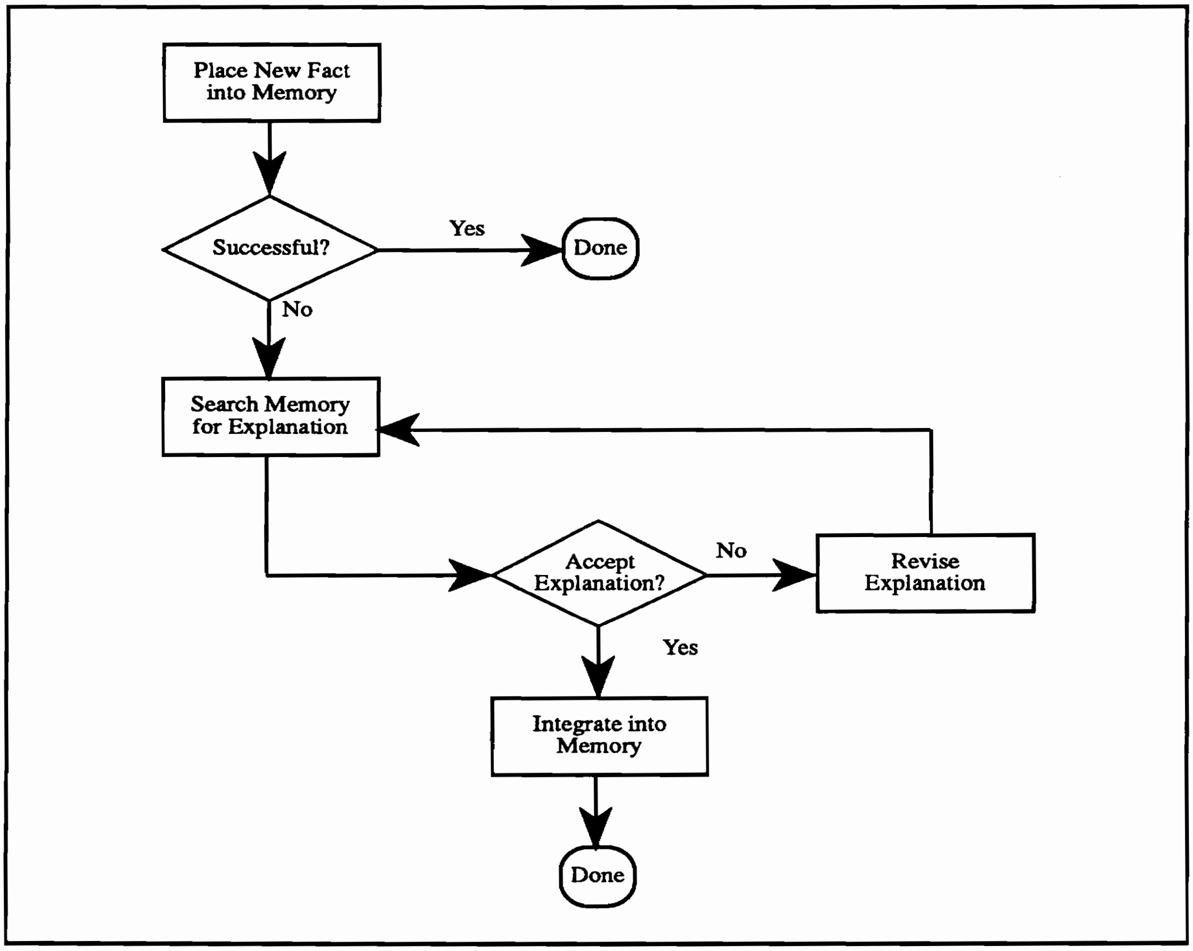


Figure 32. Case-based high-level process-flow diagram.
(Abstracted from Schank, 1986; Schank & Leake, 1990; Kass, Leake, & Owens, 1986.)

2.4.3.4 Connectionist Paradigm

The connectionist paradigm uses neural networks to develop rules for pattern recognition (McClelland, Rumelhart, & Hinton, 1986). Neural networks have input, hidden, and output layers. The input layer receives signals from sources external to the system and the output layer outputs signals out of the system. The input and output layers are coded to define what each of its signals represent (Caudill, 1990). The hidden layer are units whose inputs and outputs are internal to the system.

An input unit translates a signal from the external environment to an output which it sends to the other units in the system. Each of the units receives an input and produces an output. The relationships of the units determine what the system knows, how it responds to an input, and the output signal given by the output layer. For example, if a system having two units, u_1 (input) and u_2(output) receives an input, then u_1 sends a "message" to u_2, which then outputs a signal depending upon the signal it received from u_1.

Rumelhart, Hinton, and McClelland (1986) formally define eight major components of a neural network:

1) **set of processing units**, u_i: represents the domain concepts or characteristics which are used to classify or recognize patterns of the inputs

2) **state of activation**, A(t): represents the state of the system at time *t* and is represented by a vector of $a_i(t)$'s which are the activations for the individual units

3) **output function for each unit**, f_i: produces an output $o_i(t)$ based upon the units activation at time *t*

4) pattern of connectivity among units, matrix W(t): represents how the different units are related, noted by w_{ij}, if w_{ij} is positive then unit *j* excites unit *i*, if w_{ij} is negative then unit *j* inhibits unit *i*, if it is zero then unit *j* has no direct relationship with unit *i*

5) propagation rule for propagating activation through the network: takes the output vector and connectivity matrices and combines them to form the net_i which is the net input to unit $_i$

6) activation rule: combines the inputs impinging on a unit with the current state of that unit to produce a new level of activation for the unit, a function F takes the net input to the unit and the unit's activation state to produce a new state of activation

7) learning rule: modifies patterns of connectivity by experience, this is done in one of three ways: develop new connections, loss of existing connections, or modification of connection strengths, w_{ij} of units.

8) environment: defines the environment within which the system must operate.

Figure 33 is an example of an abstract neural network.

A pattern-association learning goal of neural networks is to find a set of connections so whenever a particular pattern reappears on the input set of units, the associated pattern will appear on the output set of units (Rumelhart, Hinton, & McClelland, 1986). The idea is to develop the connections so when an input pattern is given a predetermined or learned output pattern results.

Learning in a neural network can occur by three methods: supervised, reinforcement, or unsupervised. Supervised learning takes place by specifying the desired output vector, reinforcement uses a scalar evaluation of the output, and unsupervised constructs patterns based on the input pattern only (Hinton, 1990).

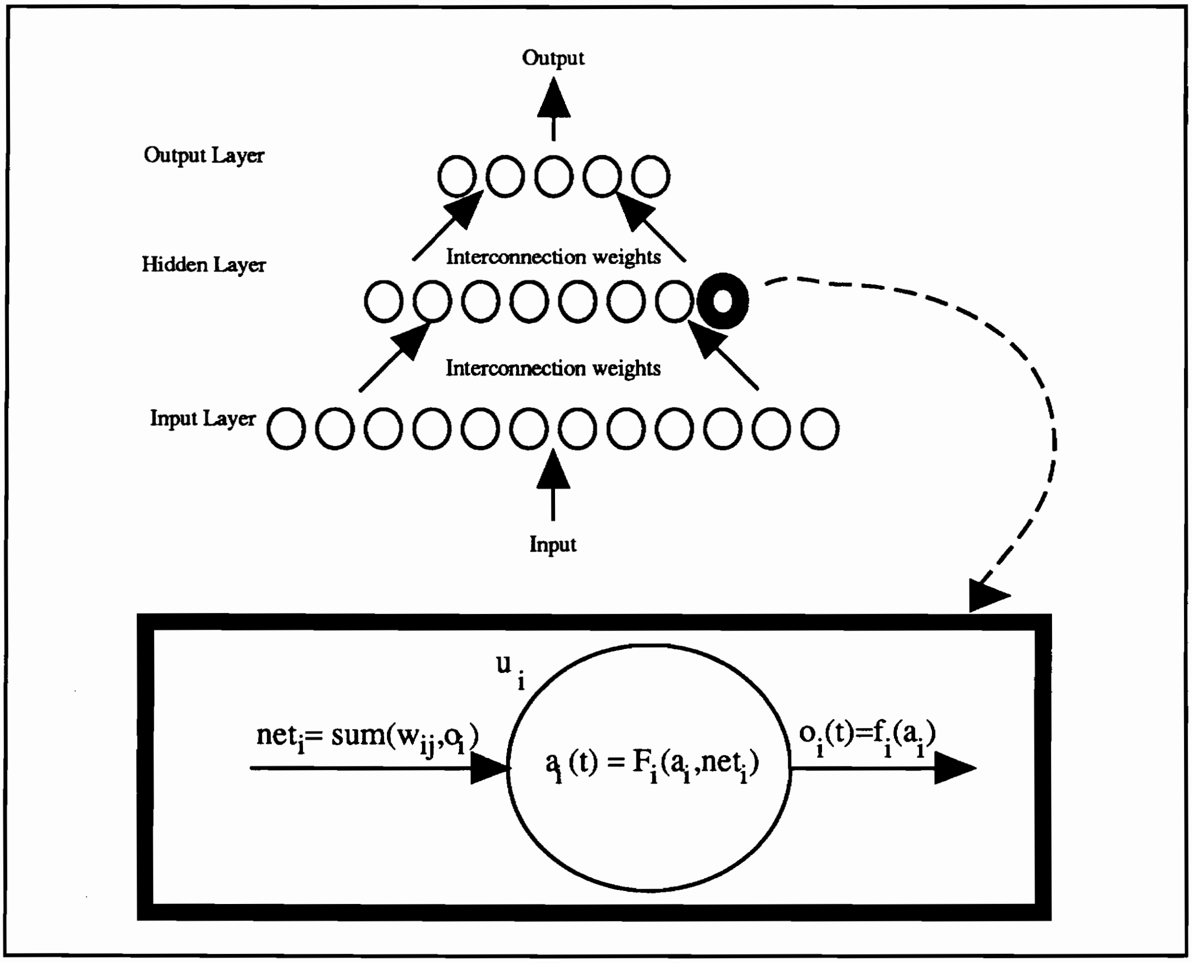


Figure 33. Abstract neural network.
(Abstracted from Hinton, 1990; Rumelhart, Hinton, & McClelland, 1986.)

The overall process for supervised learning in a neural network begins with the definition of the input layer through which the input pattern will be received. The input layer is defined by assigning a unit to each possible characteristic of the input pattern. Next, the output layer is defined by matching the output layer units with the desired output pattern. The network is "activated" by calculating the state of activation using the propagation rule, pattern of connectivity, output functions, and activation rules. The output pattern of the network is compared to the desired output pattern. If the patterns match, then the process stops, else the interconnection weights are adjusted by the learning rule. Hinton (1990) provides an example of a learning rule based on the error between the desired and produced. The error between desired and produced patterns is used to determine the change in the interconnection weights of the units. This process is iteratively done until stability is reached in the network. Figure 34 portrays this learning process.

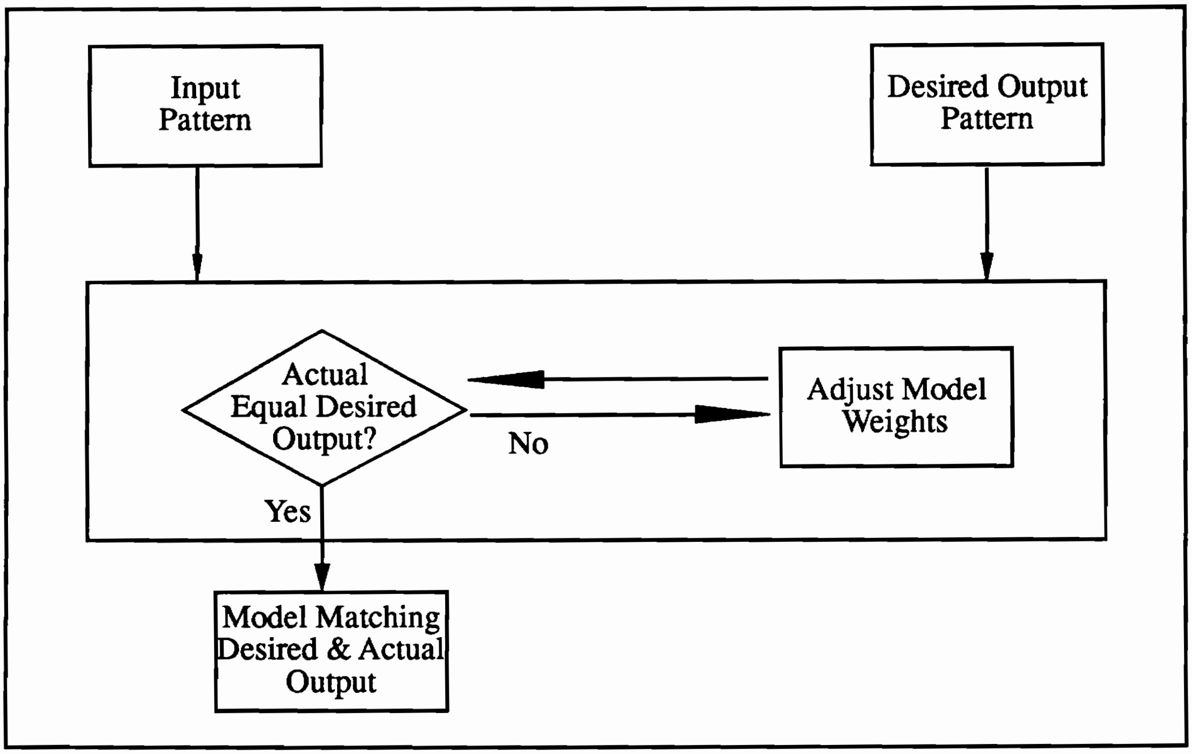


Figure 34. Abstract representation of the connectionist paradigm learning process. (Abstracted from Hinton, 1990.)

2.5 Analysis, Conclusions, and Recommendations

The methods are analyzed to determine which method should be used to perform the knowledge acquisition of problem spaces. The analysis of the methods reviewed shall progress through a process of elimination of methods that do not meet each of the five design criteria defined at the outset.

The methods were analyzed and eliminated with respect to five design criteria. 1) The method must be general purpose and not constrained by the requirement for a preexisting, domain, knowledge base prior to the commencement of knowledge acquisition, making it applicable to a variety of differing domains of application. 2) The method must produce a set of procedural units which lend themselves to processing via a production system, since a production system process is the most widely employed architecture for processing cognitive activities. 3) The method must be systematic to the extent that a process can be formulated and codified for computer implementation. 4) The method must be compatible with the manner in which procedural knowledge is typically recalled. This should supply the element of simplicity to the knowledge-acquisition process. 5) The method must be valid in terms of its being empirically grounded in experimental research attesting to its compatibility with the human information processing system.

Based on this review, the methods are classified into manual, machine aided, and machine learning. The analysis begins with the machine learning category.

The machine learning methods (except induction and connectionist paradigms) are knowledge intensive requiring some base domain knowledge prior to the commencement of knowledge acquisition. It is difficult to imagine how either the induction or connectionist paradigm can be interfaced for the user to simplify the users recall of knowledge.

Next, machine-aided methods are analyzed. None of the machine-aided methods meet the five criteria defined earlier. KNACK and SIZZLE were eliminated because they are knowledge intensive, requiring preexisting domain knowledge prior to their application as knowledge-acquisition tools. Each was designed for specific tasks.

Multi-Dimensional Scaling (MDS) and the Repertory Grid methods are eliminated since these methods apply primarily to object classification. They do not produce a hierarchy of procedural units for modeling process knowledge.

MOLE and SALT are not general purpose although each appears as if it could be extended to cover a variety of applications. The major concern with these methods is their incompatibility with the recall of procedural knowledge.

The methods implemented in LAP and Functional Reasoner employ a top-down decomposition of a task or physical system respectively. Functional Reasoner is eliminated because it was designed to model the functioning of components and their interactions, and is not procedurally based. It is not apparent that LAP facilitates the recall of procedural knowledge with simplicity due to the formal logical structure of its interface.

Manual methods are analyzed against the criteria. The methods described in "Unstructured Interview", "Observation", "Interface Design", and "Document Examination" are eliminated. These methods are not systematized to the extent that a process can be formulated and codified for computer implementation, given the state of the art in intelligent systems. The "Object Classification" methods are eliminated because they are designed to yield a network of objects and their relations as opposed to procedural knowledge. "Prompted Interviews" are eliminated because they require background, domain knowledge to perform the knowledge-acquisition process.

The remaining manual methods are classified as cognitive task analysis. These methods are Constructive Interaction and GOMS. The constructive interaction method yields a top-down breadth first decomposition of the function of a complex device. It appears as if it could be automated as a general-purpose method for knowledge acquisition in that it serves to decompose any device into a function-mechanism hierarchy. The method was developed from protocols of how individuals understand complex physical devices as opposed to task procedures, therefore it is eliminated from consideration.

The GOMS analysis technique shares similarities with that method employed by LAP in that it is general purpose, yields production rules and employs a production-system process for execution and refinement of the knowledge acquired. GOMS has not yet been implemented as an automated aid for knowledge acquisition. However, Kieras (1988) has developed a rather systematic manual interview technique which lends itself to automation. The GOMS analysis technique is also compatible with the recall of procedural knowledge since it is based upon the production theory of learning and memory which has received substantial empirical support in the research literature (Klahr, Langley, & Neches, 1987; Anderson, 1981 and 1983). Consequently, GOMS meets all criteria specified for the design of a knowledge-acquisition tool that can model knowledge of procedural domains with a reasonable degree of ease in terms of user interaction. It is intuitively quite simple for a user to recall the goals he or she wishes to achieve in the conduct of a procedural task and to specify the steps required to achieve the goal. This recall paradigm integrated with a top-down breadth first decomposition of steps is the simple heuristic employed in conducting a GOMS cognitive task analysis. Since most if not all task-related knowledge can be proceduralized in this way, GOMS has an extremely broad range for potential application. Therefore, the GOMS cognitive task analysis technique was selected for the development of an automated knowledge-acquisition tool.

Three conclusions are made from this review of the body of knowledge:

1) A means to capture the problem-space knowledge needs to be developed because the current knowledge-acquisition methods do not directly map the problem space as defined by the information processing model of problem solving.

2) A cognitive task analysis method, i.e. GOMS, will map the problem space knowledge.

3) Machine-aided tools provide the most benefit to the knowledge-acquisition process because the elicitation and refinement steps are interactively performed with the expert and the organization and representation steps are automatically performed by the machine. Therefore, the dependence on the KE is lessened.

The following recommendation for achieving the purpose of this research is made from these conclusions:

> Automation of the GOMS cognitive task analysis method in a machine-aided tool.

GOMS has been primarily used for interface design, however it could be adjusted to fit other needs. In fact, GOMS could be used to map the knowledge used in solving a problem. A relationship between the GOMS model and the information processing model of problem solving is developed.

The components of a problem space are: initial states; goals and subgoals; possible intermediate states; operators, which move the problem solver from state to state; constraints on the problem; and the evaluation function. The components of a GOMS model are: goals, operators, methods, and selection rules. The goal of the problem-solving model and GOMS are the same. A subgoal of problem solving is the same as a step of a method in GOMS. The problem-solving evaluation function is represented by the selection rules in GOMS. Finally, the operators of problem solving are represented in GOMS as the accomplishment of a subgoal because the accomplishment of a subgoal moves the state of the problem from one node to the next. At the lowest level of the problem-solving model, the operators are the same as the operators of the GOMS model because the operator represents a specific action to be taken.

CHAPTER 3 - TESTING AND EVALUATION METHODOLOGY

This chapter details the process and procedures employed in testing and evaluating the developed knowledge-acquisition tool called Cognitive Analysis Tool (CAT).

Overview and Purpose

There are several iterative phases to developing a tool such as the one tested. The first phase included conceptual analysis, requirements analysis and definition, system design, and system development. The second phase was the evaluation of the tool. This evaluation gave insight into the tool's performance and consequently into how to improve the tool. The information and knowledge gained from this evaluation was used to iterate through the design cycle and improve the tool. The evaluation phase focused upon investigating if accurate and consistent knowledge bases can be generated by using the tool. Therefore, a highly-proceduralized task such as interacting with a device that has only one correct way to accomplish a goal was modeled. The primitive level of the goal hierarchy defined the actions required to interact with the device to accomplish the various subgoals. The high-level goal and sub-goals were used to structure and segment the elementary, primitive actions.

The tool's purpose is to help people solve problems by eliciting, organizing, and representing their knowledge; that is, developing a problem space. This problem space is a knowledge base. Developing a knowledge base at this initial evaluation will give insight into the tool's ability to generate accurate and consistent representations of well-structured domains. If the tool can generate accurate and consistent knowledge bases in well-structured domains, then, by analogy, the inference can be made that the tool can be used to generate knowledge bases in ill-structured domains. However, the tool must be tested at the well-structured domain level to determine if the tool can represent well-structured domains.

The purpose of the evaluation methodology is to investigate the tool's performance relative to meeting the research objective. That is, can an automated knowledge-acquisition tool using an extended GOMS process guide individuals in organizing their knowledge to formulate problem spaces from which decisions and plans can be made? To determine if the tool works, the knowledge bases generated by using the tool must be evaluated. The tool works if the developed knowledge bases are accurate with a baseline model and consistent between subjects.

Since this research is exploratory development, a baseline is established to investigate the feasibility and practicality of the tool (Driskell & Olmstead, 1989). This baseline will also serve as a benchmark to evaluate future improvements to this tool. Current literature does not contain adequate metrics to evaluate the effectiveness of a knowledge-acquisition method (Mitta, 1989).

3.1 Questions to be Answered

The following are the questions answered by this experiment.

1) How accurate and consistent are the knowledge bases developed using the knowledge-acquisition tool?

2) How can the tool be improved or changed to facilitate the knowledge-acquisition process?

The results of the first question will indicate the degree to which the research question has been answered. The results of the second question will provide suggested improvements to the tool.

3.2 Experimental Plan

3.2.1 Overview of the Methodology

The methodology involves comparing each subjects' knowledge base as generated by the tool with 1) a baseline knowledge base and 2) the knowledge bases of the other subjects. The baseline knowledge base is taken from Kieras (1988) and adapted to facilitate testing of the tool. To determine the accuracy of the knowledge bases, the baseline is used as the basis of comparison for the knowledge bases constructed by domain experts who interact with the knowledge-acquisition tool. To determine the consistency of the knowledge bases, the knowledge base generated by each subject is compared with each of the other subjects' knowledge bases. Consistency is a measure of the variation in the knowledge bases generated. The tool must be able to generate similar models from different experts on the same well-structured domain. The consistency measure is calculated to determine if a common knowledge base can be generated from multiple experts in a domain in which a baseline knowledge base is not available. If a common knowledge base can be generated in this instance, then the inference can be made that an accurate knowledge base is being generated. Figure 35 displays the conceptual model of the methodology.

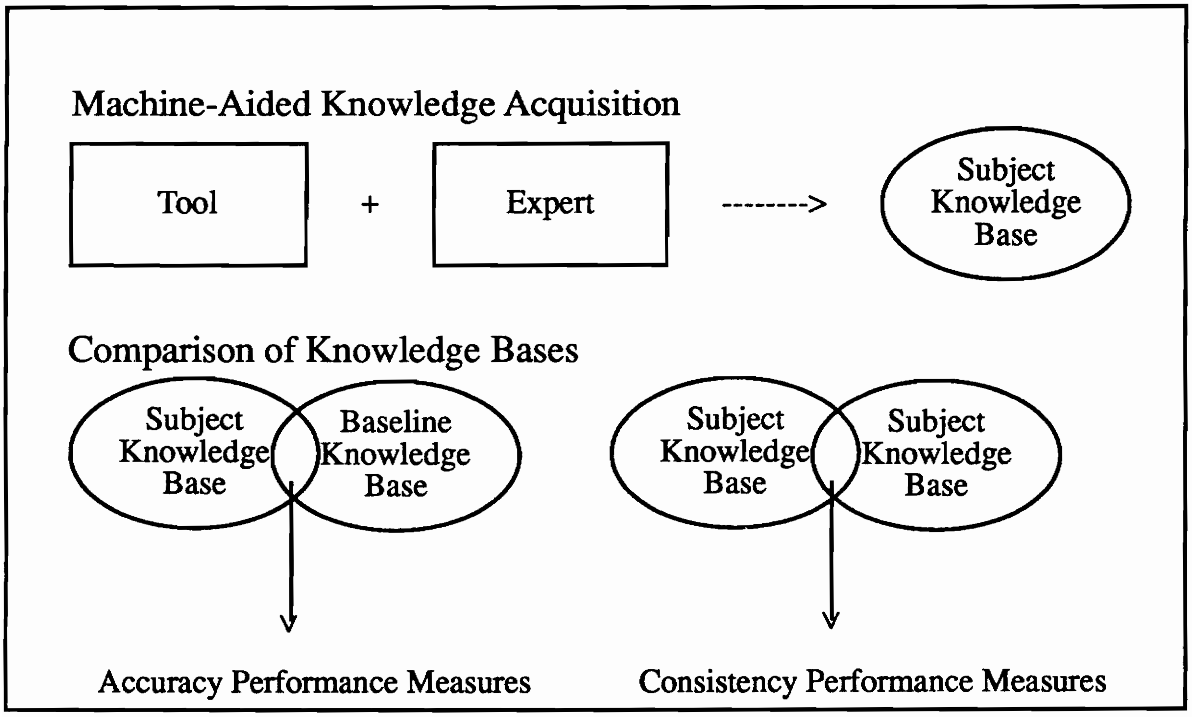


Figure 35. Conceptual model of the methodology.

3.2.2 Possible Sources of Variance for this Experiment

This section describes six possible sources of variance for this experiment. Each variance source will be described in terms of how it could influence the experiment, its different value levels (when appropriate), and how it was controlled and accounted for in the experiment.

1. Method for knowledge-acquisition.

The method used to acquire knowledge from the expert is a source of variance because the method used can influence the knowledge bases that are generated. There are any number of levels for this source of variance; that is, any of the methods identified by the analysis and classification of existing methods given in Chapter 2 could be employed for eliciting knowledge. The knowledge-acquisition method was controlled for variance by selecting one method for automation as a result of the analysis relative to the design criteria specified.

Given the purpose of this experiment is to test the tool's performance, the knowledge-acquisition tool was the only level to the knowledge-acquisition method variable.

2. Knowledge-acquisition tool.

The knowledge-acquisition tool is a major source of variance for this experiment because the tool is what is being tested and evaluated. The tool was controlled by using one design of the tool in the evaluation.

The tool's design characteristics that could influence the variance of the knowledge acquired were the process used to elicit the goal structure, interface, and prompts used. The process used to elicit the goal structure is top-down, breadth-first expansion of goals into subgoals. There are other processes such as bottom-up or depth-first. However, Miyake (1986) provided empirical evidence that people decompose their understanding employing a top-down, breadth-first process. In designing the tool, the interface and prompts were developed following the guidelines of brevity, consistency, flexibility, compatibility, and responsiveness as specified by Williges and Williges (1984) and evaluated by Williams, Hamel, and Shrestha (1987).

3. Type of experimental task.

The type of task is a source of variance because different task types could affect the ease with which knowledge can be elicited in a structured manner. This variance was controlled by using a well-structured task.

There are two high-level categories of tasks for which knowledge can be elicited: 1) well-structured tasks and 2) ill-structured tasks. A well-structured task such as the one described in 3.5 Task Definition provides a task that is performed in a highly proceduralized manner. A well-structured task will facilitate the evaluation of the knowledge bases generated by the tool and subsequently the tool itself. A well-structured task will facilitate the evaluation because there is little, if any, variation in how the task should be performed. That is, the task must be completed by a defined set of actions. Knowing these defined actions will allow the accuracy of the models to be calculated. An ill-structured task may not have any one correct knowledge base and will not allow accuracy calculations of the generated knowledge bases.

4. Source of baseline knowledge base for experimental task.

The source of the baseline model is used in the calculation of the accuracy measure and therefore is a variance source. This variable was controlled by using a single model defined in Kieras (1988).

The baseline knowledge base was used to compare the tool-generated knowledge bases. The baseline knowledge base provides a reference to investigate if the tool can be used to generate accurate models. The Kieras (1988) baseline knowledge base was chosen because it is a well-published and accepted model of how to perform the task. This model was also created using the GOMS cognitive task analysis method which provided the framework from which the tool was designed.

5. Subjects used as experts.

The subjects used as experts for the experiment are an obvious source of variance. The subjects' expertise is a significant source of variation. Homogeneity of variance between subjects was partially controlled for by their ability to perform the task to be modeled. That is, subjects' expertise was assessed by a screening task. If a potential subject could not perform the screening task, then he or she was not used as a subject for the experiment.

A subject can be characterized by his or her ability to perform the task, ability to recall knowledge on how to do the task, and how he or she structures knowledge about the task. Each subject was screened for his or her ability to perform the task. How a subject recalls his or her knowledge cannot be manipulated and was assumed to be the same for every subject. How each subject structured his or her knowledge was assessed by the data reduction procedures described in 3.7 Data Collection and Reduction.

6. Experimental procedures.

How the experiment was conducted is also a major source of variance. The experimental procedures were designed to minimize any variation in the conduct of the experiment. The procedures employed for each session are presented in Appendices 2-4. Each step was performed for each subject and checked off when performed. Any deviation from these procedures was noted. Possible deviations were open-ended questions asked by the subject for clarification purposes.

3.2.3 Assumptions

One assumption made relative to potential sources of variation between subjects was that:

> Each subject had the same ability to recall their knowledge on how to perform the task given that they could successfully complete the screening task.

This assumption is necessary since the tool was designed to capture expertise across a large population of experts independent of nuances in their recall capabilities.

3.2.4 Research Model

The research model for this experiment was developed by defining the ideal experiment, removing variables which were infeasible due to the constraints/practicalities, and integrating the overall purpose of the evaluation. The ideal experiment would use various levels of how the knowledge is acquired, various versions of the tool, different levels of tasks, and different sources of the baseline knowledge base. However, the practicalities impose the following constraints: 1) there are too many methods to evaluate and 2) an ill-structured task would not have a single knowledge base, therefore, not allowing reliable comparisons to be made. The overall purpose of this experiment was to test the current version of the developed knowledge-acquisition tool to gain insight into its performance and generate potential improvements. By controlling the other sources of variance, the performance measure values reflect the tool's ability to elicit knowledge.

Based on the above discussion, the research model is given in Figure 36. The dependent variables are the performance measures. The independent variable is the knowledge-acquisition tool.

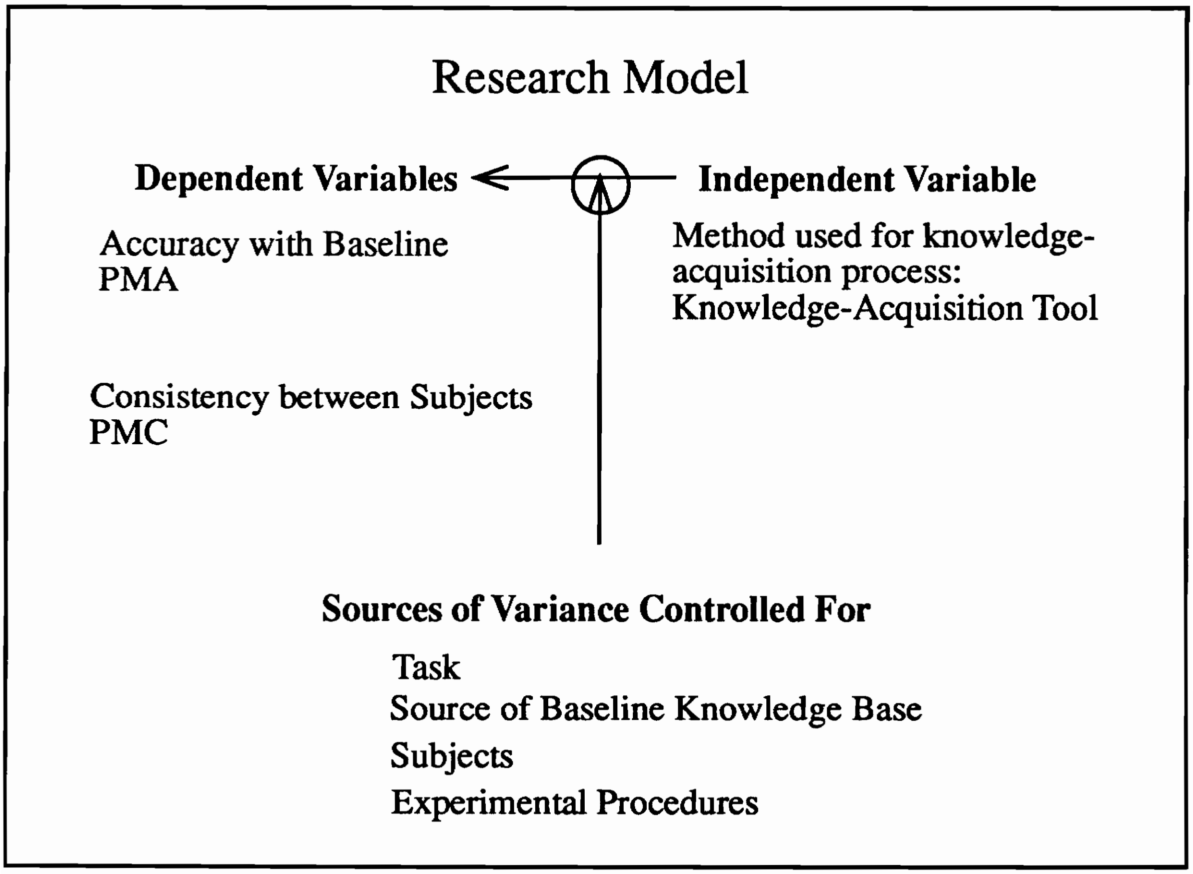

Research Model
Dependent Variables
Independent Variable
Accuracy with Baseline
PMA
Method used for knowledge-acquisition process:
Knowledge-Acquisition Tool
Consistency between Subjects
PMC
Sources of Variance Controlled For
Task
Source of Baseline Knowledge Base
Subjects
Experimental Procedures

Figure 36. Research model.

3.2.4.1 Performance Measures

The performance measures for this experiment were used to assess the capabilities of the tool to elicit accurate and reliable knowledge bases. The precise methods for calculating the performance measures are given in section 3.8.1 Calculation Procedures for Performance Measures. There were two dependent measures based upon what each knowledge base was compared to: 1) subject model versus the baseline model and 2) subject model versus other subject models. The subject-versus-baseline comparison described how accurate the models were relative to the baseline. The subject-versus-subject comparison described how consistent the knowledge bases were across all subjects. A variability measure was calculated for each performance measure. The variability measures were calculated as the standard deviation of the performance measures. The following are the performance measures:

Primitive Method Accuracy (PMA): Primitive Method Accuracy is calculated as the percent match between a subject generated primitive method and the corresponding baseline primitive method.

Primitive Method Consistency (PMC): Primitive Method Consistency is calculated as the percent match between a subject generated primitive step of a given primitive method and the corresponding primitive step of a given primitive method across all subjects.

3.3 Experimental Subjects

3.3.1 Subjects - Who They Are

The subjects were chosen based upon their ability to perform the task to be modeled. Secretaries, typists, information officers, and students from Management Systems Laboratories and the Industrial and Systems Engineering Department were the primary source of subjects.

3.3.2 Sample Size

The subjects were chosen based upon their ability to perform the task to be modeled. The sample size was forty (40). A total of forty-two (42) subjects were run. One subject's data was not used due to the subject being in a hurry and not being representative of a typical subject. Another subject's data could not be collected due to a fault in the tool during the experimental session.

3.3.3 Criteria for Selection

The subjects were selected based upon their ability to perform the task to be modeled. Prospective subjects had to perform the task before being selected as a subject for the experiment. If a prospective subject was able to perform the task, then he or she was selected as a subject for the experiment. The criteria for homogeneity in the sample is the subject's ability to perform the screening task.

3.4 Experimental Apparatus

For the screening of subjects, the following hardware and software were used:

Apple Macintosh
System Software 7
Mouse
Microsoft Word version 5.0.

For the machine-aided session of the experiment the machine-aided tool, Cognitive Analysis Tool (CAT) version 0.02, was run with the following hardware and software:

DOS version 5.0
Microsoft Windows version 3.0
Two button mouse
Win 386 IBM compatible
VGA color monitor
8 mb ram.

Appendix 8 contains a description of CAT.

3.5 Definition of Task to be Modeled

The task modeled for this experiment was moving text in a word-processing file by using the cut and paste commands. The task as defined by Kieras (1988) is to move a specified piece of text which is either a word or arbitrary text. The task is complete when the text is moved.

3.5.1 Baseline Knowledge Base of Task

The baseline model is taken from Kieras (1988). This model was constructed using the GOMS analysis technique. A model of the knowledge base is shown in Figure 37. The rule base is given in Appendix 1. The model was adapted by adding the methods for cutting and pasting text using the command keys and by removing the method for selecting a word by double-clicking on the word with the mouse. These changes were made to facilitate the evaluation of the tool. By adding the command methods, the tool's process for eliciting alternative methods could be investigated. The double-clicking method was removed because most people don't know this method.

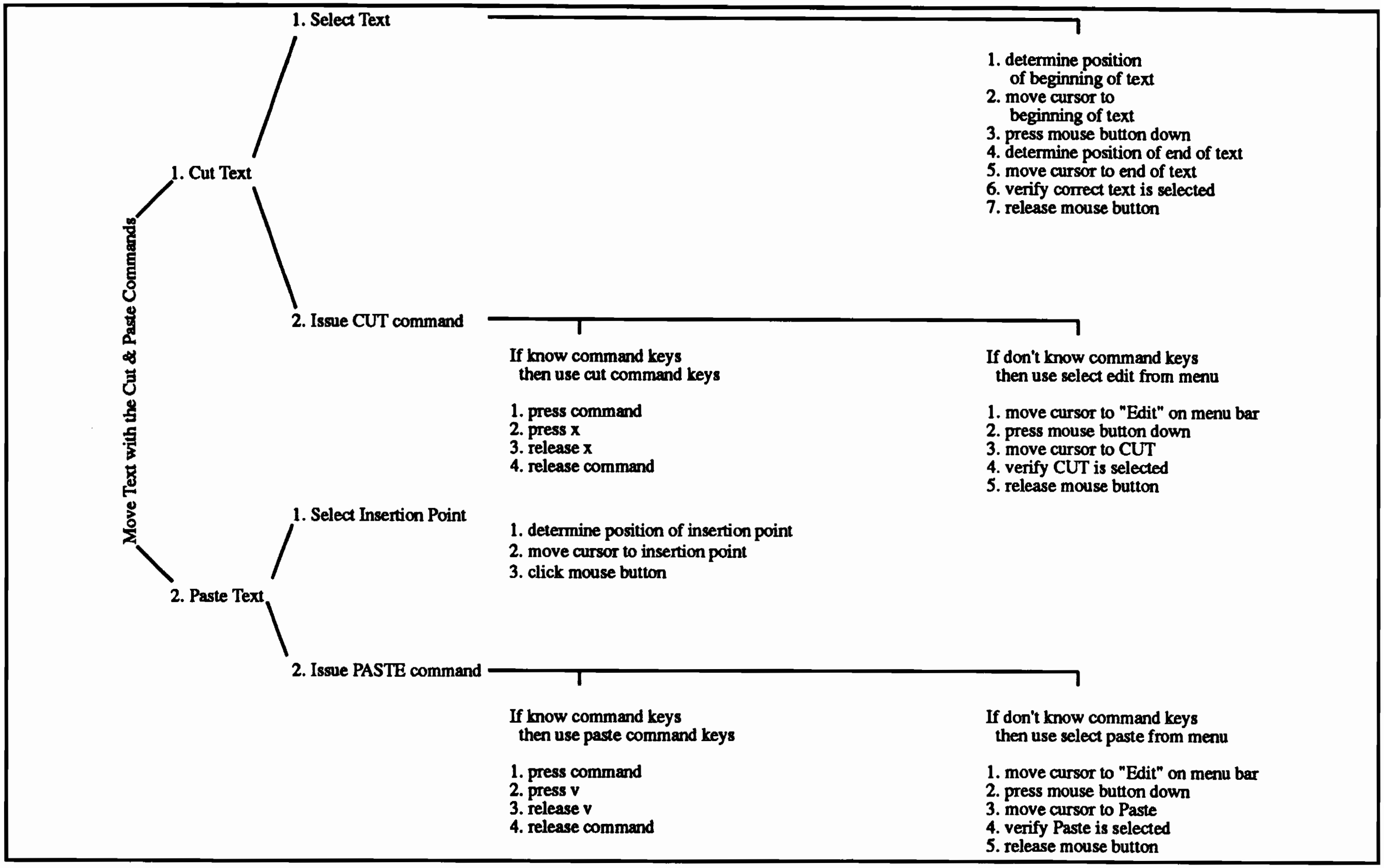

Move Text with the Cut & Paste Commands
1. Cut Text
1. Select Text
1. determine position
of beginning of text
2. move cursor to
beginning of text
3. press mouse button down
4. determine position of end of text
5. move cursor to end of text
6. verify correct text is selected
7. release mouse button
2. Issue CUT command
If know command keys
then use cut command keys
1. press command
2. press x
3. release x
4. release command
If don't know command keys
then use select edit from menu
1. move cursor to "Edit" on menu bar
2. press mouse button down
3. move cursor to CUT
4. verify CUT is selected
5. release mouse button
2. Paste Text
1. Select Insertion Point
1. determine position of insertion point
2. move cursor to insertion point
3. click mouse button
2. Issue PASTE command
If know command keys
then use paste command keys
1. press command
2. press v
3. release v
4. release command
If don't know command keys
then use select paste from menu
1. move cursor to "Edit" on menu bar
2. press mouse button down
3. move cursor to Paste
4. verify Paste is selected
5. release mouse button

Figure 37. Model of baseline knowledge base for the experimental task.
(Adapted from Kieras 1988 by adding the rules for the cut and paste key commands and by removing the rule for selecting a word by double-clicking.)

3.6 Experimental Procedure

The experimental session was conducted in three stages: 1) subject performance of the screening task, 2) machine-aided tool familiarization, and 3) machine-aided tool knowledge acquisition. Session one was performed to verify the subject's ability to perform the task being modeled. Session two, the familiarization session, was needed to allow the subject to become familiar with the tool before using it for the machine-aided knowledge-acquisition session. Session three was the interaction of experimental subjects with the tool. A checklist for the procedure used for each session is given in Appendices 2-4. Figure 38 contains the overall flowchart used for this experiment. Figure 39 contains the detailed flowchart used for processing subjects. A pilot study was run to test and change the procedures for the experiment. Appendix 6 contains the results of the pilot study.

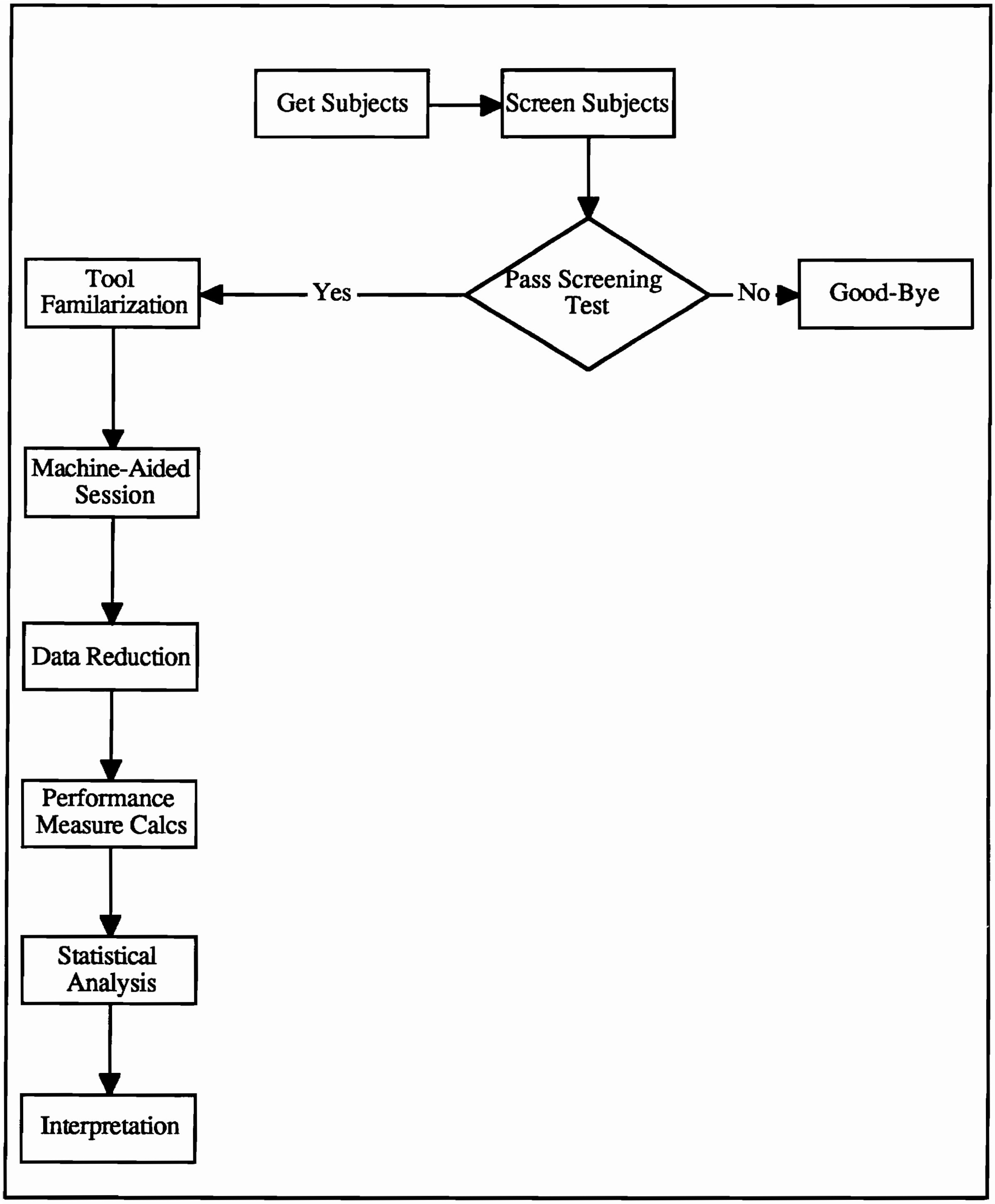


Figure 38. Flowchart of experimental procedure.

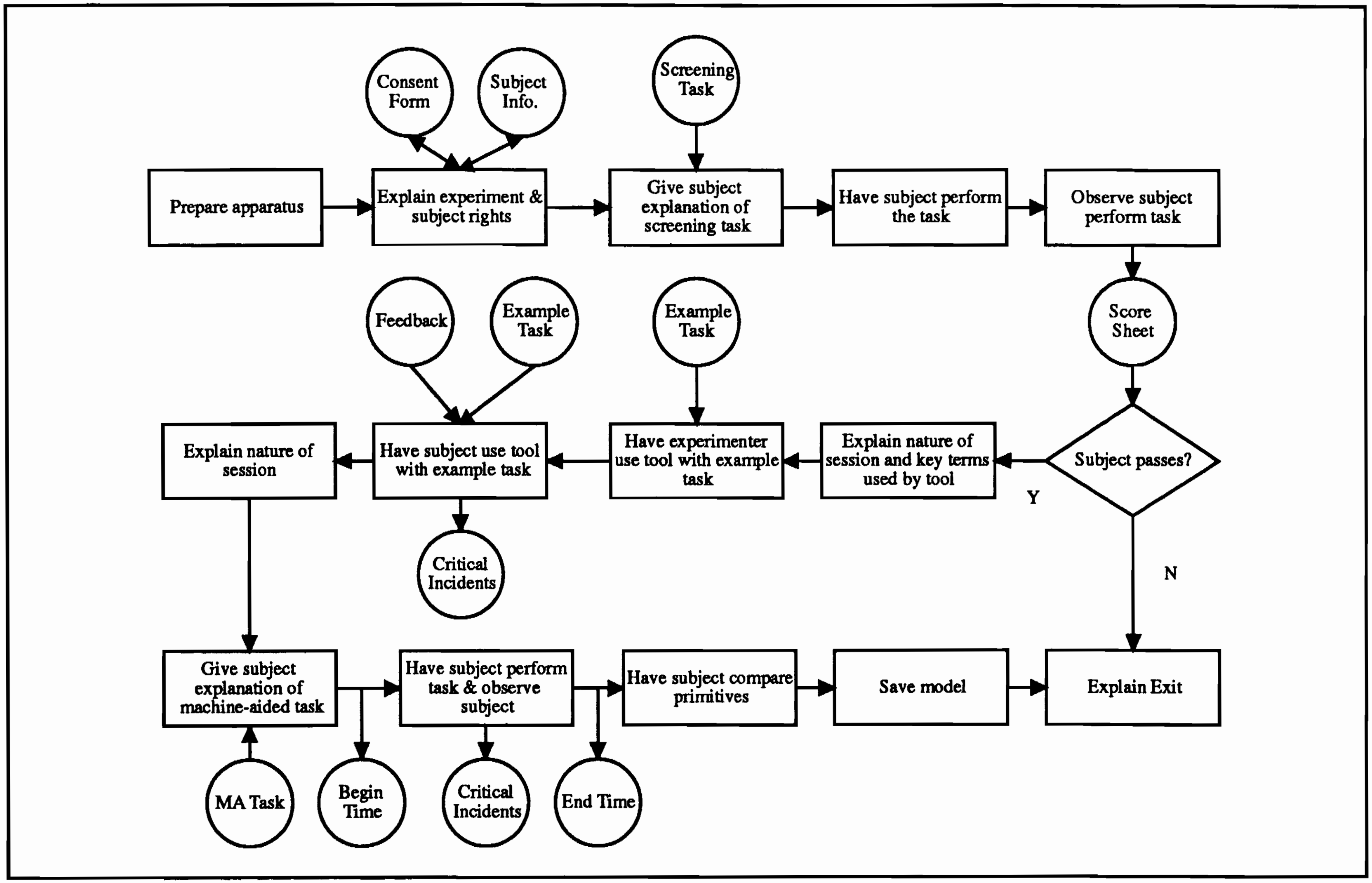

Prepare apparatus
Explain experiment & subject rights
Consent Form
Subject Info.
Screening Task
Give subject explanation of screening task
Have subject perform the task
Observe subject perform task
Score Sheet
Subject passes?
Y
N
Explain nature of session and key terms used by tool
Example Task
Have experimenter use tool with example task
Feedback
Example Task
Have subject use tool with example task
Critical Incidents
Explain nature of session
Give subject explanation of machine-aided task
MA Task
Begin Time
Have subject perform task & observe subject
Critical Incidents
End Time
Have subject compare primitives
Save model
Explain Exit

Figure 39. Detailed flowchart used for processing subjects.

3.6.1 Subject Performance of the Screening Task

The purpose of this session was to screen test subjects relative to their level of expertise. Only those who could perform the task to be modeled at a given level of performance were chosen as subjects for the experiment. The subject passed the screening task if the subject could perform each step of the screening task as defined in Appendix 2 without using the help facility or user manual; that is, if the subject could retrieve from memory the knowledge to perform all the alternative methods described in the baseline model. A checklist was used to determine if the subject performed the task as specified. Appendix 2 contains the procedure, task sheet, and subject performance scoring sheet for this session.

This session was also used to explain the overall process for the experiment and to have the subject sign the consent form. The consent form is given in Appendix 5.

3.6.2 Machine-Aided Familiarization Session

This session consisted of explaining the functions and commands of the tool. The session had three parts: 1) explanation of the tool by the experimenter, 2) guided example completed by the experimenter, and 3) subject use of the tool with an example problem and feedback from the experimenter. The purpose of this session was to provide the subject the "how to use it" knowledge of the tool. Appendix 3 contains the explanation of the tool and the sample tasks employed for demonstrating the tool. Any critical incidents, such as the subject asking the experimenter for help or an observation by the experimenter of the subject having difficulty, were noted by the experimenter. These observations gave insight into possible tool improvements. The results from these observations are presented in section 4.3 as suggested changes to be made to the tool.

3.6.3 Machine-Aided Knowledge-Acquisition Session

The machine-aided tool was used by each subject to guide the knowledge acquisition process on the task defined in section 3.5 Task Definition. The task explanation for this session is given in Appendix 4. Any critical incidents, such as the subject asking the experimenter for help or an observation by the experimenter of the subject having difficulty, were noted by the experimenter. These observations gave insight into possible tool improvements. The results from these observations are also presented in section 4.3 as suggested changes to be made to the tool.

The following facilitations were made by the experimenter to help each subject understand how to use the tool when the subject was having difficulty:

1) if the subject started to define primitive steps for accomplishing the top level goal, then the following was said: "You are probably starting at too low a level of detail. Think about the high-level steps necessary to accomplish the top-level goal. You will be able to define the primitive steps for each high-level goal later on. You can look at the models used in the previous session to help you."

2) if the subject defined two primitive actions per step, then the following was said: "You should define only one primitive action per step. There is a way to define concurrent actions."

3.6.4 Logistic Issues

The sessions were done in the Pointe West Commons conference room in one sitting. The subjects were paid five dollars ($5.00) for participating in the experiment.

3.7 Data Collection and Reduction

The data collected were the knowledge bases elicited from the subjects about the experimental task. The first step to data collection was the construction of the rules generated from the knowledge-acquisition sessions. The rules were automatically generated by the machine in the machine-aided sessions. The second step to data collection was to have each subject match each primitive he or she defined with a primitive from the baseline model. This comparison and matching was done to ensure the user-generated semantics were consistent with the names ascribed to each primitive in the baseline model. Instead of having the experimenter make judgments about semantic similarity or dissimilarity, the subjects were asked to make this judgment since they would understand how the terms they generated were to be interpreted.

The first step of data reduction was to gather the primitive methods generated by each subject. The primitive methods contain the collection of primitive steps enacted to accomplish a goal. Primitive methods are needed because this collection of primitive steps determines how the task is to be completed and represents the knowledge units that define how to do the task.

The following procedure was employed to construct the list of rules containing primitive steps:

1) Print the text output developed by the tool from an experimental session. The tool output consists of: a) a list of rules containing the goal/subgoal names, method names, selection rules (if appropriate), and steps; and b) a list of primitives.

2) Select a primitive from the primitive list.

3) Search each rule or method to see if the rule contains the primitive as a step. If the rule contains the primitive, then highlight the rule.

4) Do steps 2 and 3 until there are no more primitives.

5) Collect the highlighted rules or methods. This set makes up the list of rules that contain primitives. These rules can contain both primitives and non-primitives as steps.

The second data reduction step was to transform each subjects' list of rules into baseline methods. The six baseline methods were: select text, cut text with menu, cut text with keys, new position, paste text with menu, and paste text with keys. The second data reduction step was done to allow the comparison of each subject's knowledge base to baseline primitive methods. The primitive methods define the set of primitive steps or elementary actions required to complete the task. The construction of this primitive method list was done because subjects had different goal hierarchies. Subjects composed and decomposed rules differently.

Decomposition occurs when one method in the baseline knowledge base is decomposed into two or more sub-goals by the subject. The following shows an example of a decomposed method:

```
GOAL "cut text with menu"
        1. "move cursor to edit"
        2. "press mouse button down"
        3. "select cut"

GOAL "select cut"
        3a. "move cursor to cut"
        3b. "release mouse button".
```

The above two methods are really just the one method below:

```
GOAL "cut text with menu"
        1.  "move cursor to edit"
        2.  "press mouse button down"
        3a. "move cursor to cut"
        3b. "release mouse button"
```

In this case, the subject decomposed the one baseline method of "cut text with menu" into the two subgoals "cut text with menu" and "select cut".

An example of a composed method is the following:

```
GOAL "put text in new position"
        1.      locate new position of text
        2.      click the mouse button
        3.      press apple key
        4.      press v key
        5.      release v key
        6.      release apple key.
```

In this case, the subject chunked or combined the following two baseline methods into one method:

```
GOAL "new position"
        1.      locate new position of text
        2.      click the mouse button

GOAL "paste text with keys"
        3.      press apple key
        4.      press v key
        5.      release v key
        6.      release apple key.
```

As a matter of convenience in calculating the performance measures, the subject-generated methods were composed and decomposed into baseline methods. The experimenter composed two subject generated methods into a single baseline method and decomposed a single subject method into two baseline methods as required to compare a subject's methods to the baseline methods. Knowledge composition and decomposition can at occur at different stages in the development of a knowledge base (Anderson, 1983). Knowledge composition and decomposition do not reflect any inaccuracies in the subject generated knowledge base.

3.8 Data Analysis

Data analysis occurred in two stages. The first stage of data analysis was to calculate the performance measure values. The second stage was to calculate the variability scores (i.e., standard deviation) of the performance measures.

3.8.1 Calculation Procedures for Performance Measures

The primitive-method accuracy was calculated for each given primitive method of each subject across all methods. Each primitive step of a primitive method generated by each subject was compared to the primitive steps of the corresponding primitive method identified in the baseline model. The baseline primitive methods were: select text, cut text with menu, cut text with keys, new position, paste text with menu, and paste text with keys. Each one of these primitive methods had a set of primitives as defined in the baseline model to accomplish the method. The primitive method accuracy was calculated by comparing each primitive step in a subject's primitive method to the steps of the corresponding baseline primitive method. The number of baseline primitive steps of a given method that the subject defined in his or her corresponding primitive method was counted, as was the number of steps in the primitive baseline method. The accuracy for a given method of a subject was computed by dividing the number of baseline primitive steps of the given method that the subject defined in his or her corresponding primitive method by the number of steps in the primitive baseline method. This process was completed for each baseline method for each subject. The primitive method accuracy for each subject was calculated by taking the average accuracy for the subject across the six baseline primitive methods. The unit of analysis for the accuracy measure was a method. A method represents a knowledge chunk or rule. Table 9 is an example of the accuracy calculations.

Primitive method consistency was calculated for each primitive step of a given method across all subjects. Consistency describes how consistent subjects were in including or excluding a primitive step in a given method. The first step was to construct a list of all the primitives each subject defined for a given primitive method. This list of primitives was generated for each primitive method across all methods, that is, all of the primitives defined by all of the subjects for a given method were part of this list. Each primitive step of a primitive method generated by each subject was analyzed in terms of its presence or absence in the given method across all subjects. The number of subjects who had the primitive step in the given method was counted, as was the number of subjects who didn't have the primitive in their model for the given method. The consistency of a primitive step of a given method was calculated by dividing the maximum of either the number of subjects who had the primitive step in the given method or the number of subjects who didn't have the primitive step in the given method by the total number of subjects. This process was completed for each primitive step of a primitive method across all methods. Table 10 is an example of consistency calculations. Appendix 7 contains the actual data reduction sheets used in calculating the performance measure values.

Table 9

Example Calculations for the Accuracy Performance Measure

Method: new position	**Subject**			
Primitive Method Steps	#1	#2	#3	#4
determine new position	1	0	0	1
move cursor to new position	1	1	1	0
click mouse button	1	0	1	1
# of baseline primitives defined by subject for this given method	3	1	2	2
# of baseline primitives in this given method	3	3	3	3
Accuracy	3/3	1/3	2/3	2/3
Method: paste text with keys	**Subject**			
Primitive Method Steps	#1	#2	#3	#4
press command	1	0	1	1
press V	1	0	1	1
release V	1	0	0	1
release command	1	0	0	0
# of baseline primitives defined by subject for this given method	4	0	2	3
# of baseline primitives in this given method	4	4	4	4
Accuracy	4/4	0/4	2/4	3/4
Primitive Method Accuracy	$\frac{3/3+4/4}{2}$	$\frac{1/3+0/4}{2}$	$\frac{2/3+2/4}{2}$	$\frac{2/3+3/4}{2}$

Note: Table values: "0" = subject didn't define the primitive in this method, "1" = subject did define the primitive in this method.

Table 10
Example Calculations for the Consistency Performance Measure

Method: new position	**Subject**				**# of subjects who had step for this method**	**# of subjects who did not have step for this method**	**Subject Generated Primitive Method Consistency**
Primitive Method Steps	#1	#2	#3	#4			
determine new position	1	0	0	1	2	2	**0.500**
move cursor to new position	1	1	1	0	3	1	**0.750**
click mouse button	1	0	1	1	3	1	**0.750**
verify have correct position	0	1	0	0	1	3	**0.750**
Method: paste text with keys	**Subject**				**# of subjects who had step for this method**	**# of subjects who did not have step for this method**	**Subject Generated Primitive Method Consistency**
Primitive Method Steps	#1	#2	#3	#4			
press command	1	1	1	0	3	1	**0.750**
press V	1	1	1	0	3	1	**0.750**
release V	1	0	0	0	1	3	**0.750**
release command	1	1	0	0	2	2	**0.500**
verify text is inserted	0	0	0	1	1	3	**0.750**

Note: Table values: "0" = subject didn't define the primitive in this method, "1" = subject did define the primitive in this method.

3.8.2 Statistical Analysis

The statistical analysis used for this experiment was to establish the mean and the standard deviation of the measures. The analysis yields the variability of the models generated with the tool. Two statistical analyses were performed:

1. variability of the accuracy as compared to the baseline, and
2. variability of models between subjects, independent of the baseline model.

From these standard deviations the confidence of predicting the sample standard deviation as being representative of the true population was determined. By using a sample size of 40, the standard deviation calculated is 17% within the true population value with a confidence of 90% (Greenwood & Sandomire, 1950, p. 258).

CHAPTER 4 - ANALYSIS OF RESULTS AND INTERPRETATION

This chapter describes and interprets the analysis of the data from the experiment.

4.1 Analysis of Results

The knowledge bases generated by the subjects were analyzed for accuracy and consistency. Accuracy measures were computed by comparing the primitive-level methods and their corresponding steps to the baseline model's primitive-level methods for each method and each subject. Consistency measures were computed by comparing the primitive-level methods generated by each subject to the corresponding primitive-level methods generated by every other subject.

4.1.1 Baseline Comparison--Accuracy

The results from the analysis of the accuracy of the knowledge bases are given in Table 11. The accuracy analysis was conducted on three sets of measures based on the type of primitive within a method: mental, physical, and combined mental and physical. Mental primitives are the cognitive processes enabled while doing a task, or the implicit mental actions. For example, "determine position of the beginning of text" and "verifying correct text is selected" are mental primitives. Physical primitives are the observable physical or explicit overt behaviors activated to accomplish a task. For example, "press a key" is a physical primitive. The combined measure includes both the physical and mental primitives. The primitives used for the accuracy analysis were only the primitives defined in the baseline model, any other primitive defined by a subject that was not part of the baseline model was not included in the accuracy calculations. The non-baseline primitives are accounted for in the consistency measures, that is, between-subject comparisons.

The average number of physical primitive steps defined for a given method as compared to the baseline model was 82.7%. The average number of mental primitive steps defined for a given method as compared to the baseline model was 28.3%. The average number of combined steps (i.e., both mental and physical) defined for a given method as compared to the baseline model was 72.8%. Across all methods, mental and physical primitive accuracy was significantly different with a paired t-test ($t = -14.433$, df1 = 39, df2 = 39, at $p < 0.0001$).

4.1.2 Between-Subjects Comparison--Consistency

The results from the analysis of the consistency of the knowledge bases are given in Table 12. The consistency analysis, as in the accuracy-measure case, was divided into three sets of measures based on the type of primitive: mental, physical, and combined mental and physical. All primitive steps defined in the subjects' knowledge bases were included in the analysis.

On average, 87.1% of the subjects were consistent in **including** a physical primitive step of a primitive method across all methods of their models. On average, 89.7% of the subjects were consistent in **excluding** the mental primitive steps of a primitive method across all methods of their models. Overall, 88.5% subjects were consistent in defining, including or excluding from their model, a primitive step of any given method. That is, 88.5% of the primitive steps of a method were consistently part of or **not** part of a subject's method across all methods of their models. Table 13 contains confidence intervals for the accuracy and consistency measures.

Further analysis was done to investigate the number of primitive steps a subject defined in their model in addition to the primitive steps defined in the given baseline method. On average, a subject added 0.046 physical primitive steps to a given baseline method and 0.099 mental primitive steps to a given baseline method. A subject, on average, added 0.079 primitive steps (i.e., both mental and physical) to a given baseline method. These additional steps added further detail to the models and were not incorrect actions to complete the task. Table 14 contains the results of this analysis.

Table 11

Summary of Accuracy Results of Subject-Versus-Baseline Comparison - Number of Baseline Primitive Steps of a Method a Subject Identified in Their Model

Method	Primitive Type	Average	Standard Deviation
Select Text			
	Mental	0.358	0.266
	Physical	0.925	0.141
	Both	0.682	0.131
Cut Text with Menu			
	Mental	0.100	0.304
	Physical	0.900	0.276
	Both	0.740	0.236
Cut Text with Keys			
	Mental	NA	NA
	Physical	0.769	0.332
	Both	0.769	0.332
New Position			
	Mental	0.650	0.483
	Physical	0.700	0.389
	Both	0.683	0.346
Paste Text with Menu			
	Mental	0.025	0.158
	Physical	0.900	0.276
	Both	0.725	0.225
Paste Text with Keys			
	Mental	NA	NA
	Physical	0.769	0.332
	Both	0.769	0.332
Across all Methods			
	Mental	0.283	0.207
	Physical	0.827	0.177
	Both	0.728	0.167

Note: NA (not applicable) is given for methods which did not contain mental primitives as part of the given baseline primitive method.

Table 12
Summary of Consistency Results of Subject-Versus-Subject Comparison

Method	Primitive Type	Average	Standard Deviation
Select Text			
	Mental	0.872	0.096
	Physical	0.930	0.057
	Both	0.894	0.085
Cut Text with Menu			
	Mental	0.922	0.070
	Physical	0.910	0.034
	Both	0.917	0.057
Cut Text with Keys			
	Mental	0.906	0.047
	Physical	0.838	0.160
	Both	0.865	0.128
New Position			
	Mental	0.750	0.141
	Physical	0.783	0.191
	Both	0.770	0.154
Paste Text with Menu			
	Mental	0.925	0.071
	Physical	0.905	0.027
	Both	0.917	0.057
Paste Text with Keys			
	Mental	0.875	0.000
	Physical	0.838	0.160
	Both	0.850	0.128
Across all Methods			
	Mental	0.897	0.083
	Physical	0.871	0.120
	Both	0.885	0.102

Table 13
Confidence Intervals for Performance Measures

Measure	**Confidence Level (%)**	**Lower Limit**	**Upper Limit**
Physical Consistency			
	90	0.834	0.908
	95	0.826	0.916
	99	0.810	0.931
Mental Consistency			
	90	0.869	0.918
	95	0.864	0.923
	99	0.854	0.934
Both Consistency			
	90	0.861	0.904
	95	0.857	0.909
	99	0.849	0.917
Physical Accuracy			
	90	0.780	0.874
	95	0.770	0.884
	99	0.903	0.751
Mental Accuracy			
	90	0.228	0.339
	95	0.217	0.350
	99	0.194	0.372
Both Accuracy			
	90	0.683	0.773
	95	0.674	0.782
	99	0.656	0.800

Table 14
Number of Steps Added to a Method in the Baseline Model by a Subject

Type of Primitive	Average	Variability (Standard Deviation)
Physical	0.046	0.113
Mental	0.099	0.129
Combined	0.079	0.103

4.2 Interpretation of Results

The accuracy of the models generated by subjects as compared to the baseline model was 72.8%. For modeling physical or explicit actions, the tool is highly accurate, 82.7%. The overall inaccuracy can be mostly accounted for by the absence of mental primitives generated by test subjects. For any given method, only 28.3% of the mental primitives identified in the baseline model were generated by the test subjects. While using an expert and a knowledge engineer the time to create one rule of a knowledge base has been estimated to take one hour (Teknowledge, Inc., 1984). By having an expert use this tool, which is able to generate 72.8% accurate models, there is a potential savings of three-quarters of a knowledge engineer's time. Accuracy depends on actions defined as primitives and the baseline model used for the comparison. The ultimate judgment as to what makes an accuracy score "good" or "bad" depends on the use of the developed knowledge base.

In defining an action as a primitive, the subject or analyst is making a judgment on the level of detail required to describe the task, as determined by the level at which a step is made a primitive. The analyst makes a judgment as to what is a primitive. One analyst's primitive may be another's subgoal. This judgment is an interpretation of the level of detail needed to describe how to perform the task. These interpretations will vary.

The judgments on the level of detail needed to describe the task depend on the use of the developed knowledge bases. If the knowledge bases are to be used to transfer knowledge to people naive in the domain, (that is, have very little domain knowledge) then the models need to be more detailed and the primitives need to be defined at lower levels. If the knowledge bases are to be used to communicate the knowledge to people with intermediate domain knowledge, then the designation of primitives can be made at higher levels. That is, the lower steps can be assumed to be known by the users. This is probably the case when dealing with group problem-solving activities.

Also, for complementary actions such as pressing and releasing a key, the releasing a key primitive was not defined as often as the pressing a key primitive. The methods to "cut" and "paste" text with the "short-cut keys" has these complementary actions as primitive steps:

```
GOAL "cut/paste text with keys"
        1.      press command
        2.      press x/v
        3.      release x/v
        4.      release command
```

The "press" primitive describes the action of pushing the key down on the keyboard and the "release" primitive describes the action of allowing the key to return to its normal position. The primitives "press command" and "press x or v" were each defined by 36 subjects. However, only 24 subjects included the primitive step "release x or v" and only 27 subjects included the primitive "release the command key." The use of semantics could account for this. Some of the steps were not further decomposed into lower level activities. For example, the step "select edit" could have been broken down into lower level primitives of "move cursor to edit" and "press mouse button down." Verification of the knowledge base would catch the gaps in detail.

The mental primitives as described by Kieras (1988) are "non-observed and hypothetical, inferred by the theorist or analyst" and act as placeholders for mental activities (p. 138). The mental primitives were the non-behavioral steps taken to accomplish the task and were implicitly performed by the subjects. Since the test subjects were not cognitive analysts, it is unlikely that they would have defined many of the mental primitive implicit actions such as "determine position of beginning of text" or the more cognitive-based primitives such as "recall" and "store in long-term memory" or "store in short-term memory" as prescribed by Kieras (1988). The use of mental primitives is especially important when predicting the execution time of a cognitive task. For generating computational models which simulate cognitive processes in terms of execution time, a cognitive analyst would need to edit the models generated using this tool. It may be possible to infer certain metal primitives given knowledge of overt primitive actions and automatically insert the mental primitive steps. The inference and insertion of mental primitives is a matter for further analysis and research. However, the tool is quite capable of eliciting the explicit actions necessary to accomplish a task. This is displayed by the higher accuracy in the physical primitives.

A major issue to be addressed in evaluating the accuracy of automated aids for generating cognitive models is the question of how one defines accuracy. For the case at hand, the experimenter defined accuracy relative to a baseline model generated by an authority in cognitive complexity analysis, Dr. David Kieras. However, in the process of data reduction many questions arose relative to the accuracy of this model. For example, the method "issue paste command with the menu" describes the following primitive steps:

1) move cursor to edit on menu bar,
2) press mouse button down,
3) move cursor to paste,
4) verify paste is selected, and
5) release mouse button.

One could legitimately modify this method to include the steps "locate edit on menu bar in display" before step one of Kieras's model and "verify cursor on edit menu" immediately following step one of the Kieras model. Other primitive steps also could be added. Would these additional steps improve the accuracy of the baseline model?

It appears the accuracy of a model depends upon the use to which the model will be put. If the model is to be used to simulate a cognitive process in terms of execution time, a highly-detailed definition of all mental primitives or operators is required. However, if one wishes to describe a task to a non-cognitive psychologist, the explication of mental operators would more than likely be confusing. Consequently, the level of detail of the model generated is ultimately tied to accuracy, which in turn is conditional upon the use to which the model shall be put. Accuracy of a model is therefore relative and a judgment call on the part of the application developer. A strategy for evaluation of such machine-aided systems must therefore take into consideration the application of the models to be generated. Dependent upon the specific application, there may be a minimal model that would contain the required steps to execute a task for a given application. Many test subjects did define additional steps not included in the baseline model. These additional steps were not wrong, that is, the model could still be executed correctly to perform the task.

The goal-methods hierarchy of GOMS is by analogy similar to the function-mechanism hierarchy of constructive interaction developed by Miyake (1986). The GOMS framework consists of a hierarchy of goals and methods to achieve a goal, whereas, the constructive interaction framework consists of a hierarchy of functions (what happens) and mechanisms (how it happens). Consequently, it is quite possible that the process and framework employed to generate procedural task models can be used to generate qualitative models of complex physical devices.

The models developed were highly consistent, 88.5%. Consistency shows that the models the subjects developed were on average 88.5% similar. That is, their models contained 88.5% of the same primitive steps for any given primitive method. The consistency measure was calculated to determine if a common knowledge base can be generated from multiple experts in a domain in which a baseline knowledge base is not available. From the consistency results, the inference can be made that an accurate knowledge base can be generated in a domain that does not have a baseline model for comparison. The inconsistencies can be explained again by the way each subject interprets the level of detail needed to define the task activities. For example, only one person defined the activities of "move finger to the x key" and "press x." Other subjects simply defined the step "press x." Each subject had the ability to perform the task. The difference in the level of detail would be clarified during the verification stage.

There are five stages to knowledge acquisition: 1) elicitation, 2) organization, 3) representation, 4) refinement, and 5) verification. (See 2.3 Knowledge Acquisition for discussion of these steps.) The purpose of this experiment was to test the tool's capability in eliciting the knowledge. The organization and representation of the rules were automatically performed by the tool. The knowledge bases were not reviewed and edited with the subjects. An automated review and edit capability should be incorporated into the tool. This review and edit process would increase the accuracy and consistency of the knowledge bases generated. Also, the terms used at this primitive level are closely related, e.g., move cursor versus move mouse and pressing a key versus pressing and releasing a key. Therefore, there is a need for verification to clarify the semantics used at the most primitive level. A trained analyst will need to use and execute the developed model to find any inaccuracies in the model.

This is the only evaluation of knowledge bases in terms of their accuracy and consistency. This analysis needs to be done for other knowledge-acquisition methods including machine-aided tools to determine which methods can be used to generate the most accurate and consistent knowledge bases.

The subjects described the basic goals and subgoals to be achieved to accomplish the task of moving a piece of text using cut and paste. The basic steps are 1) select the text, 2) cut the text, 3) move cursor to the new location, and 4) paste the text. However, in defining the primitive actions to be performed, the subjects provided different levels of detail. The results show that the tool can be beneficial in eliciting an expert's knowledge. The results from using the tool are expected to increase by incorporating some changes into the tool. These changes were based on observations of subjects using the tool and suggestions from the subjects. These suggested changes are discussed next.

4.3 Suggested Improvements to the Tool

The following suggestions for improvements to the knowledge-acquisition tool, CAT, are based on the observation of and feedback from the subjects using the tool. There are four categories of suggested changes. If these suggestions are incorporated, the accuracy and consistency of the knowledge bases would be improved. All usability problems of the tool as observed by the experimenter and identified by subjects were addressed. The problems were not filtered by use of an impact analysis.

1. Graphical Representation of Knowledge Base

As the user is building the knowledge base, a graphical representation is being shown to the user. However, the current version of CAT displays only one goal-subgoal level. The graph should be expanded to show more of the knowledge base. For example, the graph could show the next one or two levels higher and have the current goal highlighted. By showing more of the graph, the subject would be able to see the context in which a step is defined.

A second issue with the graphical display is the matching of the graph with the question being asked in a yes-no dialog box. For example, when the user is asked "Do you wish to describe the steps necessary to accomplish the goal of x?", the graph does not show the context in which **x** is used or described. That is, the goal-subgoal level in which **x** is a step or subgoal is not shown. Again, the context in which the goal was defined would facilitate the user being able to answer the question more easily without having to remember "where did I define this step?" or "what did I mean by the step?" One subject said, "I had a feeling I typed this twice." The matching of the graph and a dialog box should occur right after the previous dialog box is exited and right before the next dialog box is brought up.

A third issue with the graph is the use of symbols and colors to denote what a node is: goal, method, defined step, undefined step, or primitive. The current color codes do not mean anything to the user. Also, the colors are too close in color to easily distinguish between them. One suggestion made by a subject was to use a little icon next to the node to distinguish the type of node.

2. Modes of Operation

Currently, there are two modes of operation to the tool: non-guided and guided. The non-guided mode does not provide any process for guiding the user in the elicitation phase. This mode is used for reviewing and editing an existing knowledge base. The non-guided mode allows the user to edit the knowledge base by adding or renaming a goal, method, or steps; or navigate through the knowledge base to check his or her model. One change to be made in the non-guided mode is in the process to edit a defined goal. Currently, depending on how the defined goal is edited, the results will be different. The goal can be edited as a step of another goal or as the current goal. If the goal is edited as a step, then after the goal is edited the steps to accomplish the goal will no longer be associated with the goal. In essence, a new goal is created by changing the name of a defined goal when it is edited as a step of another goal. In contrast, if the goal is edited as a current goal, then just the name of the goal changes wherever the goal is defined as a step and the goal's steps are still associated with the goal. This is implicit in the workings of the tool The results of an edit on a goal should be made explicit to the user. That is, if the name is changed, then the user should be explicitly asked if the substeps should be associated, if a new goal should be created, or if the name should be changed everywhere or just for this occurrence.

The guided mode, which was used in the experiment, provides the top-down, breadth-first expansion of the goals to perform the knowledge elicitation. As part of the guided mode, the user should have an option to stay in guided mode and traverse the knowledge base at the same time. The ability to navigate the knowledge base is related to the graphical representation issue discussed above. The guided mode should also include a refinement process. The refinement process should perform a process similar to the elicitation process, but the refinement would ask the user for any changes or extensions to the current model.

As part of the guided mode, the user should be forced to "consolidate" rules which contain more than a given number of steps into higher-level goals. Some of the subjects started defining the primitive actions without defining a goal hierarchy for accomplishing the top-level goal. That is, they input primitive actions for the method for the top-level goal The experimenter made an intervention (noted in 3.6.3 Machine-Aided Knowledge Acquisition Session) to help the subject develop the first level of the goal hierarchy. It was easy for the subject to tell the action but not necessarily to tell the groups or goals that describe the primitive steps.

For example, if a subject input the following method for a goal, then it should be consolidated:

STEP 1 "determine position of piece of text to be cut",
STEP 2 "move cursor to beginning of piece of text",
STEP 3 "press mouse button down",
STEP 4 "move cursor to end of piece of text to be cut",
STEP 5 "verify correct text has been selected",
STEP 6 "release mouse button",
STEP 7 "press apple key down",
STEP 8 "press X key down",
STEP 9 "release X key",
STEP 10 "release apple key".

The user should be asked to divide the method into higher level goals. In this case, steps 1-6 and steps 7-10 would be made into the higher-level goals "select text" and "cut text." The current version of CAT has the process to consolidate the steps, but consolidation is not part of the guided process. Consolidation needs to be made a mandatory part of the guided mode.

3. Method Editor Dialog Box

The suggestions for changes to the method editor dialog box are being made to make the tool easier for a person to use the tool.

Currently, in the method editor dialog box, which is used to define a method for accomplishing a goal, the "enter" key on the keyboard is used to end the definition of the method and "control-n" is used to define a new step. "Enter" should be used to define a new step and a new means should be incorporated to end the method editor dialog box. Users instinctively use the "enter" key for a new line. Subjects continually hit "enter" to define a new step causing the method editor dialog box being closed before the subject was done defining the method. Also, the use of control-n for a new step is hard for a "ten-finger typist" because he or she must remove their hand from the home position on the keyboard to press the control and n keys.

When the user creates a method for a goal, he or she is presented with the method editor dialog box. This dialog box has a field for the method name and fields for the steps of the method. When the method editor dialog box is first presented the cursor is in the field for the first step and not in the method name field which is the first field in the layout of the method editor dialog box. The cursor should be placed in the method name field when the method editor dialog box is initially shown. This could also eliminate some problems in defining alternative methods. Some subjects were confused on how to define alternative methods. They would define the alternative methods as steps to the goal, instead of each method having its own set of steps and method name.

While in non-guided mode, the user should have the option of editing the method for a goal with the method editor dialog box not just one step of a method at a time. Also, if a new step is inserted in the method editor dialog box, then the new step should be inserted right after the location of the cursor and not at the end of the list of steps. Cut and paste functions would also help the user in the method editor dialog box. When a user describes a step that is longer than the step-name field box and goes to the next step, the text description should be returned to the beginning of the step-name field box so that the user can more easily tell what steps have been defined.

A pull-up box of primitives from which the user can select from a library of primitives to use as a step in the current method could be added to the method editor dialog box. Having a library of primitives would help the user determine the level of detail needed in the knowledge base because it is difficult to determine the primitive level. As one subject said it is "hard to figure how far to breakdown" the steps. The primitives are usually defined by the analyst and are dependent on the system being analyzed (Kieras, 1988).

4. Yes-No Dialog Boxes

Yes-no dialog boxes are used to ask the user if: 1) a goal is a primitive or not, 2) there are alternative methods to accomplish a goal, and 3) there are alternative sets of conditions. These yes-no dialog boxes all look the same except for the prompt string. The user must read each prompt completely to determine the purpose of the dialog box. The yes-no dialog boxes should be distinguished in some way so the user can visually distinguish the purpose of the yes-no dialog box without reading the complete prompt. For example, the user is asked "Do you wish to describe the steps necessary to accomplish the goal of x?" and the available responses are yes or no. If the user chooses yes, then a method editor dialog box is displayed for the goal. If the user choose no, then the goal is made a primitive. The user chooses yes or no by clicking on a button in the yes-no dialog box with yes or no written on it. Another way to do this is by having "decompose" on the yes button and "primitive" on the no button. This way the user can easily determine the purpose of the yes-no dialog box and make a decision on what to do for the goal. Also, the goal that is being asked about by the yes-no dialog box should be highlighted and not in the same text format as the prompt string.

CHAPTER 5 - CONCLUSIONS AND RECOMMENDATIONS

5.1 Is the Problem Solved?

This section explicitly explains how the body of knowledge review and the evaluation methodology are connected to the research question and purpose. Figure 40 shows these connections. Each node represents a subgoal to be achieved to accomplish the goal of solving the problem. This research is attempting to solve a problem (to get knowledge from disparate sources). To solve this problem, there is a question (How do we extend and automate GOMS to elicit knowledge of a problem space from individuals?) and purpose (overcome the knowledge-acquisition bottleneck via an automated knowledge-acquisition tool). The solid lines show what is supported in the literature. The dotted lines show what the evaluation methodology is supporting.

The problem was solved. A process, GOMS, for performing the knowledge elicitation that corresponds to the components of the information processing model of problem solving was found, extended, and automated in the form of a machine-aided knowledge-acquisition tool, CAT. The tool was tested and evaluated. The knowledge bases generated by using the tool were found to be accurate and consistent. Further enhancements to the tool are expected to increase the accuracy and consistency of the knowledge bases generated with the tool.

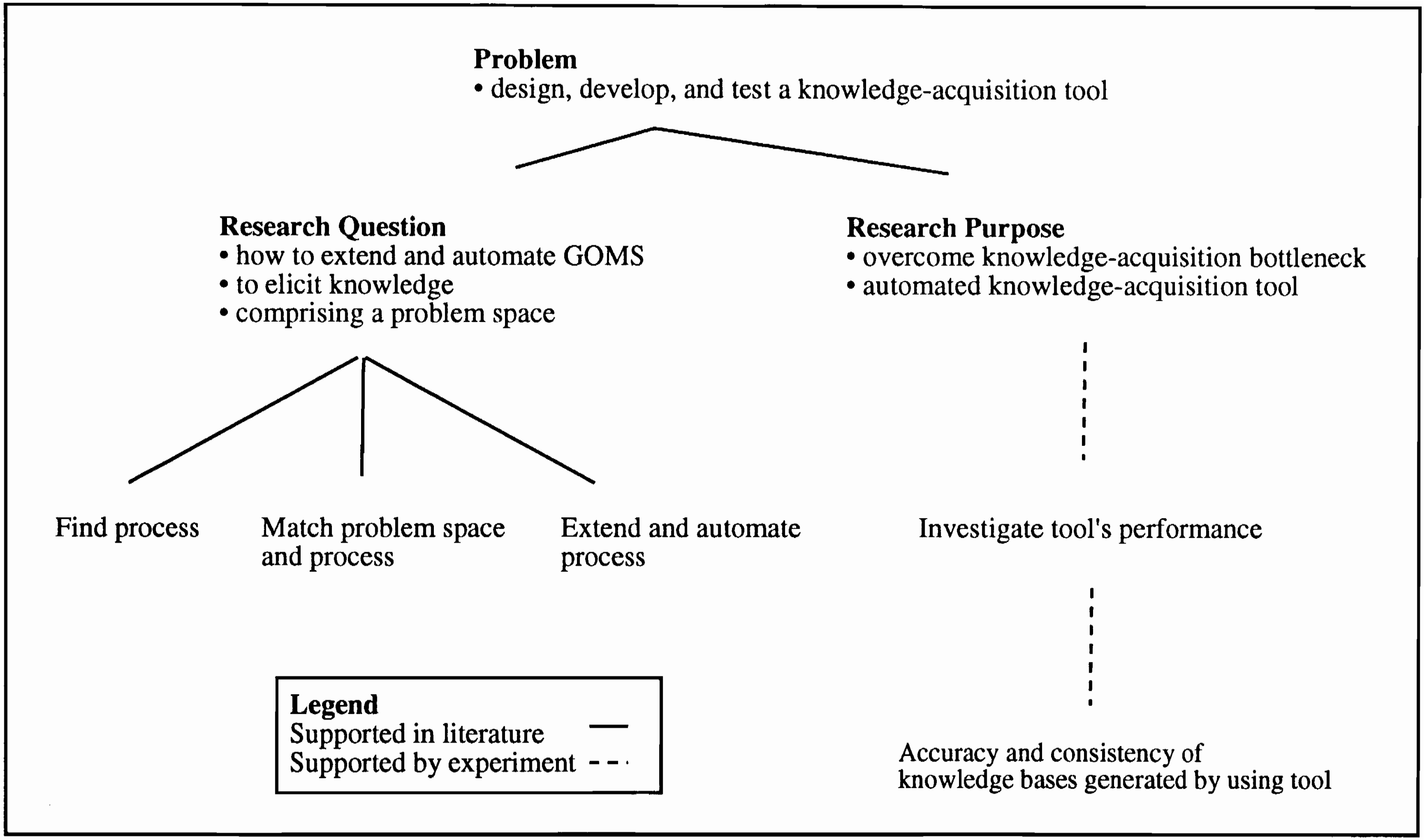

Problem
• design, develop, and test a knowledge-acquisition tool
Research Question
• how to extend and automate GOMS
• to elicit knowledge
• comprising a problem space
Research Purpose
• overcome knowledge-acquisition bottleneck
• automated knowledge-acquisition tool
Find process
Match problem space and process
Extend and automate process
Investigate tool's performance
Accuracy and consistency of knowledge bases generated by using tool
Legend
Supported in literature
Supported by experiment

Figure 40. Connections of the parts of research.

5.2 Recommendations for Further Research

The future research areas are divided into further testing of the tool, extension of the tool, and knowledge-acquisition tool evaluation criteria.

Further testing of the tool involves the incorporation of the various sources of variance as defined in 3.2.2 Possible Sources of Variance for this Experiment. An obvious and beneficial experiment would be to compare the tool's performance to that of a knowledge engineer (KE). Different versions of the tool could also be tested. For example, the sequence in which the prompts are asked could cause a difference in system performance. After the suggested changes are incorporated, the tool could be tested again to determine the effect of the changes. The tool should be tested with other types of tasks. The "best" way to use the tool is an important issue. What is the role of the KE with the tool? Should we use the tool to build an initial knowledge base and use a KE to refine the initial knowledge base?

Two major extensions to the tool need to be researched. The first extension is the capability of the tool to semantically "understand" what the user is inputting into the tool, i.e., incorporation of a natural language processor. This would probably allow the tool to better elicit the expert's knowledge by looking for gaps or missing steps, such as a mental primitive, or complementary physical primitives, such as pressing and releasing a key.

The second tool extension is the incorporation of a derivational analogy system (described in 2.4.3.3.2 Learning by Derivational Analogy). By having a derivational analogy system and understanding what the expert is inputting, the following could be achieved:

1) past plans or models could be matched and adapted to the current model being inputted,

2) past plans or models could be used to guide the expert in the elicitation and refinement processes, and

3) comparison of knowledge bases could be made to determine the correctness of a given knowledge base or rules within the knowledge base.

Further research is needed in the development of techniques and measures used to evaluate knowledge-acquisition tools. Formal techniques such as usability studies can be done to determine how well the user-tool interaction is, but the tool must also be evaluated against its purpose of generating knowledge bases. The methodology used for this experiment seemed adequate for an initial evaluation of the tool. The evaluation of the tool is based on the evaluation of the knowledge bases. The measures used also were adequate, but the accuracy measure could not be used if a baseline model was not available. Evaluation of the knowledge bases built and the metrics used to evaluate the knowledge base need to be researched. The research community will provide the evaluation on the appropriateness of the methodology and measures used for this experiment.

5.3 If I Did the Experiment over Again, What Would I Change?

I would make these changes if I were to run the experiment again:

1. Establish the common meaning of the terms used to describe the task, for example, to distinguish between press, hold, depress, release, and click a key or mouse button.

2. Do two separate studies: first a usability study and then a knowledge-base-development study. The results from this study could be improved if some of the usability issues are addressed.

3. When the subjects are interpreting their primitives, a printout of the complete knowledge base in the form of a graph would have facilitated the comparisons because the subject would have been able to see the context (i.e., goal-subgoal hierarchy) in which the primitives were used.

4. Videotape the training session to guarantee each subject receives the same exact training.

5. Take better notes while observing the subjects by explaining the subject's action, conditions under which the action occurred, results of the actions, and how the actions could be avoided or accounted for.

5.4 Summary of Major Outputs

The following is a list of the outputs of doing this research:

1) A complete review of thirty-one knowledge-acquisition methods from manual to machine learning,

2) An evaluation methodology and metrics to evaluate knowledge-acquisition methods,

3) An evaluation of a developed automated knowledge-acquisition tool called Cognitive Analysis Tool (CAT), in terms of the accuracy and consistency of the knowledge bases generated by using the tool, and

4) Suggested improvements to the current version of the tool.

REFERENCES

Abdul-gader, A. H., & Kozar, K. A. (1990). Discourse analysis for knowledge acquisition: The coherence method. Journal of Management Information Systems, 6(4), 61-82.

Abrett, H., & Burstein, M. H. (1988). The KREME knowledge editing environment. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 1-24). San Diego, California: Academic Press Inc.

Adeli, H., & Yeh, C. (1990). Explanation-based machine learning in engineering design. Engineering Applications of Artificial Intelligence, 3(2), 127-37.

Anderson, J. R. (1981). Acquisition of cognitive skills. ONR Technical Report 81-1.

Anderson, J. R. (1983). The architecture of cognition. Cambridge, Massachusetts: Harvard University Press.

Berube, D. S. (1990). Assessing differences in data and information makeup at two different organizational levels using two managerial jobs. Unpublished master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Bhansali, S., & Harandi, M. T. (1990). Program synthesis using derivational analogy (Tech. Rep. No. UIUCDCS-R-90-1591) Urbana-Champaign: University of Illinois at Urbana-Champaign, Department of Computer Science.

Booker, L. B., Goldberg, D. E., & Holland, J. H. (1990). Classifier systems and genetic algorithms. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 235-282). Cambridge, Massachusetts: The MIT Press.

Boose, J. H., & Bradshaw, J. M. (1988). Expertise transfer and complex problems: Using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 39-64). San Diego, California: Academic Press Inc.

Bradshaw, J. M. & Boose, J. H. (1990). Decision analysis techniques for knowledge acquisition: Combining information and preferences using Aquinas and Axotl. International Journal of Man-Machine Studies, 32(2), 121-86.

Broner, I., King, J. P., & Nevo, A. (1990). Structured induction for agricultural expert systems knowledge acquisition. Computers and Electronics in Agriculture, 5(2), 87-99.

Burton M. (1972). Semantic dimensions of occupation names. In A. K. Romney, R. N. Shepard, & S. B. Nerlove (Eds.), Multidimensional scaling: Theory and applications in the behavioral sciences: Volume II: Applications (pp. 55-71). New York: Seminar Press Inc.

Bylander, T., & Chandrasekaran, B. (1988). Generic tasks in knowledge-based reasoning: The "right" level of abstraction for knowledge acquisition. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 65-77). San Diego, California: Academic Press Inc.

Card, Moran, & Newell. (1983). The psychology of human-computer interaction. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

Carbonell, J. G. (1990). Introduction: Paradigms for machine learning. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 1-9). Cambridge, Massachusetts: The MIT Press.

Caudill, M. (1990). Using neural nets: Making an expert network - part 4. AI Expert, 5(7), 41-45.

Cendrowska, J. (1988). PRISM: An algorithm for inducing modular rules. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 255-276). San Diego, California: Academic Press Inc.

Cleaves, D. A. (1988). Cognitive biases and corrective techniques: Proposals for improving elicitation procedures for knowledge-based systems. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 23-34). San Diego, California: Academic Press Inc.

Cooke, N. M., & McDonald, J. E. (1988). The application of psychological scaling techniques to knowledge elicitation for knowledge-based systems. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 65-82). San Diego, California: Academic Press Inc.

Dauer, J. A. (1990). Successful knowledge elicitation. Information Executive, 3(1), 54-6.

Diederich, J., Ruhmann, I., & May. (1988). KRITON: A knowledge-acquisition tool for expert systems. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 83-94). San Diego, California: Academic Press Inc.

Di Piazza, J. S. (1990). Interweaving knowledge extracting, organizing, and evaluating: A concrete design for preventing logic and structure bugs while interviewing experts. Journal of Automated Reasoning, 6(3), 299-317.

Driskell, J. E., & Olmstead, B. (1989). Psychology and the military: Research applications and trends. American Psychologist, 44(1), 43-54.

Dym, C. L., & Levitt, R. E. (1991). Knowledge-based systems in engineering. McGraw Hill, Inc.

Ericson, K. A., & Simon, H. A. (1984). Protocol analysis: Verbal reports as data. Cambridge, Massachusetts: The MIT Press.

Eshelman, L. (1988). MOLE: A knowledge-acquisition tool for cover-and-differentiate systems. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 37-80). Boston, Massachusetts: Kluwer Academic Publishers.

Eshelman, L., Ehret, D., McDermott, J., & Tan, M. (1988). MOLE: A tenacious knowledge acquisition tool. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 95-108). San Diego, California: Academic Press Inc.

Feigenbaum, E., McCorduck, P., and Nii, H.P. (1989). The rise of the expert company. New York: Vintage Books.

Gaines, B. R. (1988). An overview of knowledge-acquisition and transfer. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 3-22). San Diego, California: Academic Press Inc.

Garg-Janardan, C., & Salvendy, G. (1988). A conceptual framework for knowledge elicitation. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 119-129). San Diego, California: Academic Press Inc.

Gennari, J. H., Langley, P., & Fisher, D. (1990). Model of incremental concept formation. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 11-61). Cambridge, Massachusetts: The MIT Press.

Greenwood, J. A., & Sandomire, M. N. (1950). Sample size required for estimating the standard deviation. Journal of American Statistical Association, 45, 258.

Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. Machine Learning, 5(4), 355-81.

Gruber, T. R., & Cohen, P. R. (1988). Design for acquisition: Principles of knowledge-system design to facilitate knowledge acquisition. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 131-147). San Diego, California: Academic Press Inc.

Hayward, S. A., Woelinga, B. J., & Breuker, J. A. (1988). Structured analysis of knowledge. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 149-160). San Diego, California: Academic Press Inc.

Hinton, G. E. (1990). Connectionist learning procedures. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 185-234). Cambridge, Massachusetts: The MIT Press.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1987). Classifier systems, q-morphisms, and induction. In L. Davis (Ed.), Genetic algorithms and simulated annealing (pp. 116-128). London: Pitman Publishing.

Kahn, G. (1988). MORE: From observing knowledge engineers to automating knowledge acquisition. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 7-35). Boston, Massachusetts: Kluwer Academic Publishers.

Kass, A., Leake, D., & Owens, C. (1986). Appendix: SWALE, a program that explains. In Schank, R. C., Explanation patterns: Understanding mechanically and creatively (pp. 232-54). Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc.

Kelly, G. A. (1955). The psychology of personal constructs. New York: Norton.

Kieras, D. E. (1988). Towards a practical GOMS model methodology for user interface design. In M. Helander (Ed.), Handbook of human-computer interaction (pp. 135-157). North-Holland: Elsevier Science Publishers B.V.

Kieras, D. E., & Polson, P. G. (1985). An approach to the formal analysis of user complexity. International Journal of Man-Machine Studies, 22, 365-394.

Klahr, D., Langley, P., & Neches, R. (Eds.) (1987). Production system models of learning development. Cambridge, Massachusetts: The MIT Press.

Klinker, G. (1988). KNACK: Sample-driven knowledge acquisition for reporting systems. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems, (pp. 125-174). Boston, Massachusetts: Kluwer Academic Publishers.

Klinker, G., Bentolila, J., Genetet, S., Grimes, M., & McDermot. (1988). KNACK: Report-driven knowledge acquisition. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 195-209). San Diego, California: Academic Press Inc.

Kodratoff, Y., Manago, M., & Blythe, J. (1988). Generalization and noise. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 301-324). San Diego, California: Academic Press Inc.

LaFrance, M. (1988). The knowledge acquisition grid: A method for training knowledge engineers. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 81-91). San Diego, California: Academic Press Inc.

Leedy, P. D. (1989). Practical research (4th edition). New York: Holt, Rinehart & Winston.

Littman, D. C. (1988). Modeling human expertise in knowledge engineering: Some preliminary observations. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 93-104). San Diego, California: Academic Press Inc.

Management Systems Laboratories & Virginia Productivity Center. (1992). NPR/VPI (MSL/VPC) grand strategy system academic and research plan (v 3.2). Blacksburg, VA: Author.

Marcus, S. (1988a). SALT: A knowledge-acquisition tool for propose-and-revise systems. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 81-123). Boston, Massachusetts: Kluwer Academic Publishers.

Marcus, S. (1988b). Taking backtracking with a grain of SALT. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 211-226). San Diego, California: Academic Press Inc.

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E.(1986). The appeal of parallel distributed processing. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), Parallel distributed processing (pp. 45-76). Cambridge, Massachusetts: The MIT Press.

McDermot, J. (1988). Preliminary steps toward a taxonomy of problem-solving methods. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 225-256). Boston, Massachusetts: Kluwer Academic Publishers.

Michalski, R. S. (1991). Inferential learning theory: A conceptual framework for characterizing learning processes (Report: P91-13 MLI 91-6). Washington, D.C.: George Mason University, Center for Artificial Intelligence, School of Information Science and Technology.

Minton, S., Carbonell, J. G., Etzioni, O., Knoblock, C. A., & Kuokka, D. R.. (1987). Acquiring effective search control rules: Explanation-based learning in the PRODIGY system. Proceedings Fourth International Workshop on Machine Learning (pp. 122-133).

Mitta, D. A. (1989). Knowledge acquisition: human factors issues. Proceedings of the Human Factors Society: 33rd Annual Meeting, 1, 351-355.

Miyake, N. (1986). Constructive interaction and the iterative process of understanding. Cognitive Science, 10, 151-177.

Mockler, R. J. (1990). Non-technical manager's modelling of management decision situations. Journal of Systems Management, 41(5), 7-12.

Moore, E. A., & Agogino, A. M. (1988). INFORM: An architecture for expert-directed knowledge acquisition. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 227-244). San Diego, California: Academic Press Inc.

Mostow, J. (1990). Design by derivational analogy: Issues in the automated replay of design plans. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 119-184). Cambridge, Massachusetts: The MIT Press.

Newell A., & Simon, H. A. (1972). Human problem solving. Englewood Cliffs, N.J.: Prentice-Hall.

Offutt, D. (1988). SIZZLE: A knowledge-acquisition tool specialized for the sizing task. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 175-200). Boston, Massachusetts: Kluwer Academic Publishers.

O'Leary, D. E., & Watkins, P. R. (1990). A portfolio of knowledge acquisition approaches for a knowledge-based system. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 3, 264-8. Los Alamitos, CA: IEEE Computer Society Press.

Olson Reitman, J., & Rueter, H. H. (1987). Extracting expertise from experts: Methods for knowledge acquisition. Experts Systems, 4(3), 152-168.

Pazzani, M. J. (1988). Explanation-based learning for knowledge-based systems. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 217-237). San Diego, California: Academic Press Inc.

Pettit, E., & Pettit, M. (1988). Analysis of the performance of a genetic algorithm-based system for message classification in noisy environments. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 335-350). San Diego, California: Academic Press Inc.

Prietula, M. J., Beauclair, R. A., & Lerch, F. J. (1990). A computational view of group problem solving. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 3, 101-9. Los Alamitos, CA: IEEE Computer Society Press.

Pugh, D. R., & Price, C. J. (1990). Automating knowledge acquisition for generic tasks. Engineering Applications of Artificial Intelligence, 3(3), 171-9.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), Machine learning: An artificial intelligence approach (pp. 463-482). Los Altos, California: Morgan Kaufmann Publishers, Inc.

Quinlan, J. R. (1988). Simplifying decision trees. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 241-254). San Diego, California: Academic Press Inc.

Regoczei, S. & Plantinga, E. P. O. (1988). Creating the domain of discourse: Ontology and inventory. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 293-308). San Diego, California: Academic Press Inc.

Rowley, D. T. (1990). PC/BEAGLE (software package review). Expert Systems, 7(1), 58-60.

Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), Parallel distributed processing (pp. 45-76). Cambridge, Massachusetts: The MIT Press.

Saunders, B. W. (1982). The industrial engineering profession. In G. Salvendy (Ed.), Handbook of industrial engineering (pp. 1.1.1-1.1.16). New York: John Wiley & Sons.

Schank, R. C. (1986). Explanation patterns: Understanding mechanically and creatively. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Schank, R. C., & Leake, D. B. (1990). Creativity and learning in a cased-based explainer. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 353-385). Cambridge, Massachusetts: The MIT Press.

Shaw, M. L. G., & Gaines, B. R. (1988). KITTEN: Knowledge initiation and transfer tools for experts and novices. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 309-338). San Diego, California: Academic Press Inc.

Shepard, R. N. (1972). Introduction to volume I. In R. N. Shepard, A. K. Romney, & S. B. Nerlove (Eds.), Multidimensional scaling: Theory and applications in the behavioral sciences: Volume I: Theory (pp. 1-19). New York: Seminar Press Inc.

Simon, H. A. (1960). The new science of management decisions. New York: Harper and Brothers.

Simon, H. A. (1975). The functional equivalence of problem solving skills. Cognitive Psychology, 7, 268-288.

Teknowledge, Inc. (1984). T-1 user's guide. California: Author.

Trippi, R. R., & Turban, E. (1990). Auto-learning approaches for building expert systems. Computers & Operations Research, 17(6), 553-60.

U. S. Department of Energy: Office of New Production Reactors. (1990). New production reactors program plan (DOE Report No. DOE/NP-0007P). Washington, D.C.: Author.

Voss, J. F., Greene, T. R., Post, T. A., & Penner, B. C. (1983). Problem solving skill in the social sciences. In G. H. Bower (Ed.), The psychology of learning and motivation: Advances in research theory (Vol. 17) (pp. 165-212). New York: Academic Press.

Voss, J. F., Tyler, S. W., & Yengo, L. A. (1983). Individual differences in the solving of social sciences problems. In R. F. Dillon & R. R. Schmeck (Eds.), Individual differences in cognition (pp. 205-232). New York: Academic Press.

Weber, E. S., Liou, Y. I., Chen, M., & Nunamaker, J. F., Jr. (1990). Toward more intelligent organizations. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 4, 290-9. Los Alamitos, CA: IEEE Computer Society Press.

Welbank, M. (1990). An overview of knowledge acquisition methods. Interacting with Computers, 2(1), 83-91.

Williams, K. E., Hamel, C. J., & Shrestha, L. B. (1987). Handbook for evaluating user-computer interfaces for computer-aided instruction. Naval Training Systems Center Technical Report: NTSC TR87-033.

Williams, K. E. (1991). Strategic information management center: Functional description. Blacksburg, Virginia: Virginia Polytechnic Institute and State University, Management Systems Laboratories.

Williges, B. H. & Williges, R. C. (1984). Dialogue design considerations for interactive computer systems. In Muckler, F. A. (Ed.), Human Factors Review: 1984 (pp. 167-208). Santa Monica, California: The Human Factors Society.

BIBLIOGRAPHY

Alexander, J. H., Freiling, M. J., Shulamn, S. J., Rehfuss, S., & Messick, S. L. (1988). Ontological analysis: An ongoing experiment. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 25-37). San Diego, California: Academic Press Inc.

Anderson, J. R. (1990). A theory of the origins of human knowledge. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 313-351). Cambridge, Massachusetts: The MIT Press.

Atkinson, G. L. (1990). Technology transfer utilizing automated knowledge acquisition tools. In B. Silverman, V. Hwang, & S. Post. (Eds.), Proceedings of the Fifth Annual AI Systems in Government Conference (Cat. No.90CH2875-3), (pp. 88-93). Los Alamitos, CA: IEEE Computer Society Press.

Bachant, J. (1988). RIME: Preliminary work toward a knowledge-acquisition tool. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 201-224). Boston, Massachusetts: Kluwer Academic Publishers.

Belkin, N. J., Brooks, H. M., & Daniels, P. J. (1988). Knowledge elicitation using discourse analysis. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 107-124). San Diego, California: Academic Press Inc.

Boose, J. H., Shema, D. B., & Bradshaw, J. M. (1990). Capturing design knowledge for engineering trade studies. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 78-89). Amsterdam, Netherlands: IOS.

Buntine, W. (1988). Induction of horn clauses: Methods and the plausible generalization algorithm. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 277-297). San Diego, California: Academic Press Inc.

Cleary, J. G. (1988). Acquisition of uncertain rules in a probabilistic logic. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 325-334). San Diego, California: Academic Press Inc.

Davis, J. P. & Bonnell, R. D. (1990). Producing visually-based knowledge specifications for acquiring organizational knowledge. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 105-22). Amsterdam, Netherlands: IOS.

Delgrande, J. P. (1988). A formal approach to learning from examples. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 163-181). San Diego, California: Academic Press Inc.

Elofson, G. S., & Konsynski, B. R. (1990). Supporting knowledge sharing in environmental scanning. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 4, 281-8. Los Alamitos, CA: IEEE Computer Society Press.

Gaines, B., & Linster, M. (1990). Development of second generation knowledge acquisition systems. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 143-60). Amsterdam, Netherlands: IOS.

Gale, W. A. (1988). Knowledge-based knowledge acquisition for a statistical consulting system. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 109-118). San Diego, California: Academic Press Inc.

Gascuel, O. (1990). A conceptual regression method. In Morik, K. (Ed.), EWSL89 Proceedings of the Fourth European Working Session on Learning, (pp. 81-90). London, UK: Pitman.

Goul, M., Philippakis, A., Richards, S., Sandman, T., & Schamp, A. (1990). Project CoEx: A distributed artificial intelligence orientation to the design of a cooperating experts' electronic meeting system. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 3, 89-100. Los Alamitos, CA: IEEE Computer Society Press.

Green, J., & d'Oliveira, M. (1982). Learning to use statistical tests in psychology: A student's guide. Milton Keynes, England: The Open University Press.

Irani, K. B., Cheng, J., Fayyad, U. M., & Qian, Z. (1990). Application of machine learning techniques to semiconductor manufacturing. Proceedings of the SPIE - The International Society for Optical Engineering, 1293 (Pt. 2), 956-65.

Jae Kyu Lee, In Koo Lee, Hyung Rim Choi, & Sung Mahn Ahn. (1990). Automatic rule generation by the transformation of Expert's Diagram: LIFT. International Journal of Man-Machine Studies, 32(3), 275-92.

Johnson, P. E., Zaulkernan, I., & Garber, S. (1988). Specification of expertise. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 125-145). San Diego, California: Academic Press Inc.

Kahn, G. S., Breaux, E. H., De Klerk, P., & Joseph, R. L. (1988). A mixed-initiative workbench for knowledge acquisition. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 161-173). San Diego, California: Academic Press Inc.

Kearney, M. (1990). Making knowledge engineering productive. AI Expert, 5(7), 46-51.

Kitto, C., & Boose, J. H. (1988). Heuristic for expertise transfer: An implementation of a dialog manager for knowledge acquisition. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 175-194). San Diego, California: Academic Press Inc.

Klein, M., Lu, S.C.-Y., & Baskin, A. B. (1990). Towards a theory of conflict resolution in cooperative design. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 4, 41-50. Los Alamitos, CA: IEEE Computer Society Press.

Kornell, J. (1988). Formal thought and narrative thought in knowledge acquisition. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 35-44). San Diego, California: Academic Press Inc.

Kowalczyk, W., & Treur, J. (1990). On the use of a formalized generic task model in knowledge acquisition. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 198-221). Amsterdam, Netherlands: IOS.

Lafferty, L., Koller, A. M., Jr., Taylor, G., Schumann, R., & Evans, R. (1990). Techniques for capturing expert knowledge: An expert systems/hypertext approach. Proceedings of the SPIE - The International Society for Optical Engineering, 1293 (Pt. 1), 181-91.

Langley, P., & Zytkow, J. M. (1990). Data-driven approaches to empirical discovery. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 283-312). Cambridge, Massachusetts: The MIT Press.

Larichev, O. I., & Moshkovich, H. M. (1990). Decision support system class for R&D planning. First International Conference on Expert Planning Systems (Conf. Publ. No. 322 pp. 227-32). London, England: IEE.

Liou, Y. I., & Nunamaker, J. F., Jr. (1990).
Using a group decision support system environment for knowledge acquisition: a field study. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 3, 40-9. Los Alamitos, CA: IEEE Computer Society Press.

Marcus, S. (1988). Introduction. In S. Marcus (Ed.), Automating knowledge acquisition for expert systems (pp. 1-6). Boston, Massachusetts: Kluwer Academic Publishers.

Major, N., & Reichgelt, H. (1990). ALTO: An automated laddering tool. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 222-36). Amsterdam, Netherlands: IOS.

McIntyre, S. C., Higgins, L. F., & Couger, J. D. (1990). Knowledge base enrichment via object oriented creativity techniques. Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences, 3, 373-81. Los Alamitos, CA: IEEE Computer Society Press.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. Artificial Intelligence, 42(2-3), 363-91.

Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1990). Explanation-based learning: A problem solving perspective. In J. Carbonell (Ed.), Machine learning: Paradigms and methods (pp. 63-118). Cambridge, Massachusetts: The MIT Press.

Mitchell, A. A. (1988). The use of alternative knowledge-acquisition procedures in the development of a knowledge-based media planning system. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 147-159). San Diego, California: Academic Press Inc.

Morik, K. (1988). Acquiring domain models. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 245-256). San Diego, California: Academic Press Inc.

Musen, M. A., Fagan, L. M., Combs, D. M., & Shortliffe, E. H. (1988). Use of a domain model to drive an interactive knowledge editing tool. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 257-273). San Diego, California: Academic Press Inc.

Neale, I. M. (1990). Modelling expertise for KBS development. Journal of the Operational Research Society, 41(5), 447-458.

Rappaport, A. (1988). Multiple-problem subspaces in the knowledge design process. In J. H. Boose & B. R. Gaines (Eds.), Knowledge acquisition tools for expert systems (pp. 275-292). San Diego, California: Academic Press Inc.

Reynolds, R. G., Maletic, J. I., & Porvin, S. E. (1990). PM: A system to support the automatic acquisition of programming knowledge. IEEE Transactions on Knowledge and Data Engineering, 2(3), 273-82.

Shadbolt, N., & Wielinga, B. (1990). Knowledge based knowledge acquisition: The next generation of support tools. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 313-38). Amsterdam, Netherlands: IOS.

Stone, E. F. (1981). Research methods in organizational behavior. Glenview, Illinois: Scott Foresman.

van Aken, E. M. (1991). A multiple case study on the information system to support self-managing teams. Unpublished master's thesis, Virginia Polytechnic Institute and State University.

van Someren M. W., Li Ling Zheng, & Post, W. (1990). Cases, models or compiled knowledge? - A comparative analysis and proposed integration. In B. Wielinga, J. Boose, B. Gaines, G. Schreiber, & van M. Someren (Eds.), Current trends in knowledge acquisition (pp. 339-55). Amsterdam, Netherlands: IOS.

Whitson, G., Wu, C., & Taylor, P. (1990). Using an artificial neural system to determine the knowledge base of an expert system. In Berghel, H., Unger, E., & Rankin, R. (Eds.), Proceedings of the 1990 ACM SIGSMALL/PC Symposium on Small Systems, (pp. 268-70). New York, New York: ACM.

Wilkins, D. C., Clancey, W. J., & Buchanan, B. G. (1988). Knowledge base refinement by monitoring abstract control knowledge. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 183-195). San Diego, California: Academic Press Inc.

Wisniewski, E., Winston, H., Smith, R., & Kleyn, M. (1988). A conceptual clustering program for rule generation. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 197-215). San Diego, California: Academic Press Inc.

Woods, D. D., & Hollnagel, E. (1988). Mapping cognitive demands in complex problem-solving worlds. In B. R. Gaines & J. H. Boose (Eds.), Knowledge acquisition for knowledge-based systems (pp. 45-63). San Diego, California: Academic Press Inc.

Wu, Z., Yang, T., Ni, F., He, Z., & Yu, Z. R. (1990) C-oriented tool for building second generation expert systems. Future Generations Computer Systems, 6(1), 77-83.

APPENDIX 1 - RULE BASE FOR EXPERIMENTAL TASK

The following is the baseline knowledge base generated with the knowledge-acquisition tool. The primitive method list is also given. The model is adapted from Kieras (1988) by adding the rules for using the cut and paste key commands and removing the rules for selecting a word by double-clicking on the word.

Baseline Knowledge Base

```
MAINGOAL "move text with cut and paste";

GOAL COMPLETE "move text with cut and paste";
  METHOD "move text using cut and paste";
    STEP "cut text",
    STEP "paste text";
    END;
  END;

GOAL COMPLETE "cut text";
  METHOD "steps to cut text";
    STEP "select text",
    STEP "issue Cut command";
    END;
  END;

GOAL COMPLETE "paste text";
  METHOD "steps to paste text";
    STEP "select insertion point",
    STEP "issue Paste command";
    END;
  END;

GOAL COMPLETE "select text";
  METHOD COMPLETE "select arbitrary text";
    STEP "determine position of beginning of text",
    STEP "move cursor to beginning of text",
    STEP "press mouse button down",
    STEP "determine position of end of text",
    STEP "move cursor to end of text",
    STEP "verify correct text is selected",
    STEP "release mouse button";
    END;
  END;

GOAL COMPLETE "issue Cut command";
  METHOD COMPLETE "cut command keys";
    STEP "press command",
    STEP "press x",
    STEP "release command",
    STEP "release x";
    IF "know command keys"
      TRY METHOD "cut command keys";
    END;
  METHOD COMPLETE "use Cut menu command";
    STEP "move cursor to Edit on menu bar",
    STEP "press mouse button down",
    STEP "move cursor to Cut",
    STEP "verify Cut is selected",
    STEP "release mouse button";
    IF "don’t know cut command keys"
      TRY METHOD "use Cut menu command";
    END;
  END;

GOAL COMPLETE "select insertion point";
  METHOD "steps to select the insertion point";
    STEP "determine position of insertion point",
    STEP "move cursor to insertion point",
    STEP "click mouse button";
    END;
  END;

GOAL COMPLETE "issue Paste command";
  METHOD COMPLETE "paste command keys";
    STEP "press command",
    STEP "press v",
    STEP "release command",
    STEP "release v";
    IF "know paste command keys"
      TRY METHOD "paste command keys";
    END;
  METHOD COMPLETE "paste menu command";
    STEP "move cursor to Edit on menu bar",
    STEP "press mouse button down",
    STEP "move cursor to Paste",
    STEP "verify Paste is selected",
    STEP "release mouse button";
    IF "don’t know paste command keys"
      TRY METHOD "paste menu command";
    END;
  END;
```

Primitive Method List for Baseline Knowledge Base

```
GOAL COMPLETE "select text";
METHOD COMPLETE "select arbitrary text";
    STEP "determine position of beginning of text",
    STEP "move cursor to beginning of text",
    STEP "press mouse button down",
    STEP "determine position of end of text",
    STEP "move cursor to end of text",
    STEP "verify correct text is selected",
    STEP "release mouse button";


GOAL COMPLETE "issue Cut command";
METHOD COMPLETE "cut command keys";
    STEP "press command",
    STEP "press x",
    STEP "release command",
    STEP "release x";

METHOD COMPLETE "use Cut menu command";
    STEP "move cursor to Edit on menu bar",
    STEP "press mouse button down",
    STEP "move cursor to Cut",
    STEP "verify Cut is selected",
    STEP "release mouse button";


GOAL COMPLETE "select insertion point";
METHOD "steps to select the insertion point";
    STEP "determine position of insertion point",
    STEP "move cursor to insertion point",
    STEP "click mouse button";

GOAL COMPLETE "issue Paste command";
METHOD COMPLETE "paste command keys";
    STEP "press command",
    STEP "press v",
    STEP "release command",
    STEP "release v";

METHOD COMPLETE "paste menu command";
    STEP "move cursor to Edit on menu bar",
    STEP "press mouse button down",
    STEP "move cursor to Paste",
    STEP "verify Paste is selected",
    STEP "release mouse button";
```

APPENDIX 2 - PROTOCOL AND TASK SHEETS FOR SCREENING SESSION

Protocol for Screening Task Session

__0. Prepare apparatus for the screening task.
   a. Turn computer on.
   b. Load Microsoft Word.
   c. Load screening task file.

__1. Give subject the informed consent form to sign.

Please read and sign the Informed Consent Form. If you have any questions, please ask me and I will answer them for you.

__2. Have subject sign consent form and fill out the questionnaire.

__3. Explain nature of task.

The purpose of this session is to have you perform a task that you will later be asked on how to perform. This is done to ensure that you can do the task.

You will use Microsoft Word version 5.0 on the Macintosh to perform the task defined on the piece of paper I will give to you. I will observe you performing the task. If you do not complete the task as described, then you will not be used for further parts of the experiment and you will be paid $1.

__4. Give subject written explanation of task.

__5. Have subject perform task.

__6. Observe subject perform task.

__7. Check against criteria to see if subject performed task adequately, then do next session. If subject didn't perform task adequately, then terminate process with subject.

Extraordinary Observations of Session

Subject Information Questionnaire

Subject #___

1. How long have you used a Macintosh?

   3 months   6 months   9 months   longer than a year

2. How long have you used Microsoft Word for the Macintosh?

   3 months   6 months   9 months   longer than a year

3. How often do you use Microsoft Word?

   everyday   once a week   once a month

Task Explanation for Screening Session

During this session you are to perform the following operations on the text displayed on the computer screen.

FOLLOW THE STEPS EXACTLY AS DEFINED.

Operations to be done on the text.

1. Select the word "out" in line 6.

2. Cut the text with the **cut menu command.**

3. Move the cursor to the bottom of the paragraph and insert the text by using the **paste shortcut/command/keyboard equivalent/quick keys.**

4. Select the text beginning in line 9 with "degrees" and ending in line 10 with "Virginia."

5. Cut the text with the **cut shortcut/command/keyboard equivalent/quick keys.**

6. Move the cursor to line 1.

7. Insert the text with the **paste menu command.**

Text to Perform Operations on:

```
1    Text Example
2
3    The College of Engineering has a reputation for offering an excellent
4    education for the student who desires to obtain a baccalaureate degree.
5    Today's engineering freshman class has an average scholastic achievement
6    score of 1200 out of a possible 1600.  This is significantly higher than the
7    mean national score of 906.  By graduating more than 1,000 students each
8    year, the college consistently ranks among the top ten in the number of
9    baccalaureate degrees granted.  The College of Engineering is the second
10   largest college at Virginia Tech.  Graduate and undergraduate enrollment
11   has increased 44% from 1976 to 1988.  The greatest increase has been in
12   graduate enrollment.
13
```

Score Sheet for Subject Screening Session

Subject # ___

__1. Select the word "out" in line 6.

__2. Cut the text with the **cut menu command.**

__3. Move the cursor to the bottom of the paragraph and insert the text by using the **paste shortcut/command keys.**

__4. Select the text beginning in line 9 with "degrees" and ending in line 10 with "Virginia."

__5. Cut the text with the **cut shortcut/command keys.**

__6. Move the cursor to line 1.

__7. Insert the text with the **paste menu command.**

APPENDIX 3 - PROTOCOL AND TASK SHEETS FOR FAMILIARIZATION SESSION

Protocol for Familiarization Session

__0. Prepare apparatus.
- __a. Turn computer on.
- __b. Set CAT window to full screen.

__1. Explain purpose of this session.
The purpose of this session is to familiarize you with a tool called CAT. We will do three things in this session: 1) I will define the terms and process used by CAT, 2) I will run through an example, and 3) you will do an example on your own.

__2. Define keys terms used in CAT.
- a. **top-level goal** - the main task we are eliciting knowledge about, an example is "prepare to drive a car"
- b. **goal** - an action or task which is described by steps, an example is "prepare to drive a car"
- c. **steps** - action or task taken to accomplish a goal, examples are "sit in seat" and "put seat belt on"
- d. **method** - a set of steps to accomplish a goal; used to describe or group the steps
- e. **selection rule** - specifies the conditions under which a given method is used to accomplish a goal when have alternative methods; an example is "the door is locked"
- f. **primitive** - a step which is not decomposed into further substeps, there are mental and physical primitives, examples of mental primitives are "determine position of door handle" and "verify seat belt is fastened", examples of physical primitives are "move handle to door" and "remove the key"

Show picture of the example of mailing a letter and explain what each these key words mean with respect to the example.

__3. Do a walk-through example
- a. top-level goal screen
  1. explain dialog box help - move over each area of the screen
  2. pressing the help button
  3. purpose of this screen - name the top-level goal
  4. input the goal from the example - mail a letter
- b. method editor
  1. purpose - define the set of steps to accomplish a goal
  2. parts of the dialog box
     step name - describes the actions to be done to accomplish the goal
     method name - used as a description or name of the steps; important especially when define alternative methods
  3. moving between the parts - tab key or mouse button
  4. defining a new step - New/Cntrl-N
  5. do example - enter the steps of mail a letter
  6. **explain only one action per step**

c. order the steps
   1. purpose - define the order the steps should be performed
   2. parts of the dialog box
   3. results of choosing each type
   4. do example - both "As entered" and "Other"

d. define alternative methods
   1. purpose - define another set of steps
   2. parts of the dialog box
   3. results of pressing Yes - takes to the method editor to define another set of steps for the current goal; do this if there alternative ways to accomplish the goal
   4. results of pressing No
   5. do example
   6. **show if hold left mouse down then can "hide" current dialog box**

e. definition of selection rule
   1. purpose - get if defined alternative methods
   2. parts of the dialog box
   3. one condition per line
   4. do example

f. alternative selection rules
   1. purpose - allow you to specify another set of conditions
   2. parts of the dialog box
   3. results of pressing Yes
   4. results of pressing No

g. define a method for the goal of
   1. purpose
   2. parts of the dialog box
   3. results of pressing Yes - define steps to accomplish the goal
   4. results of pressing No - makes the step a primitive
   5. do example

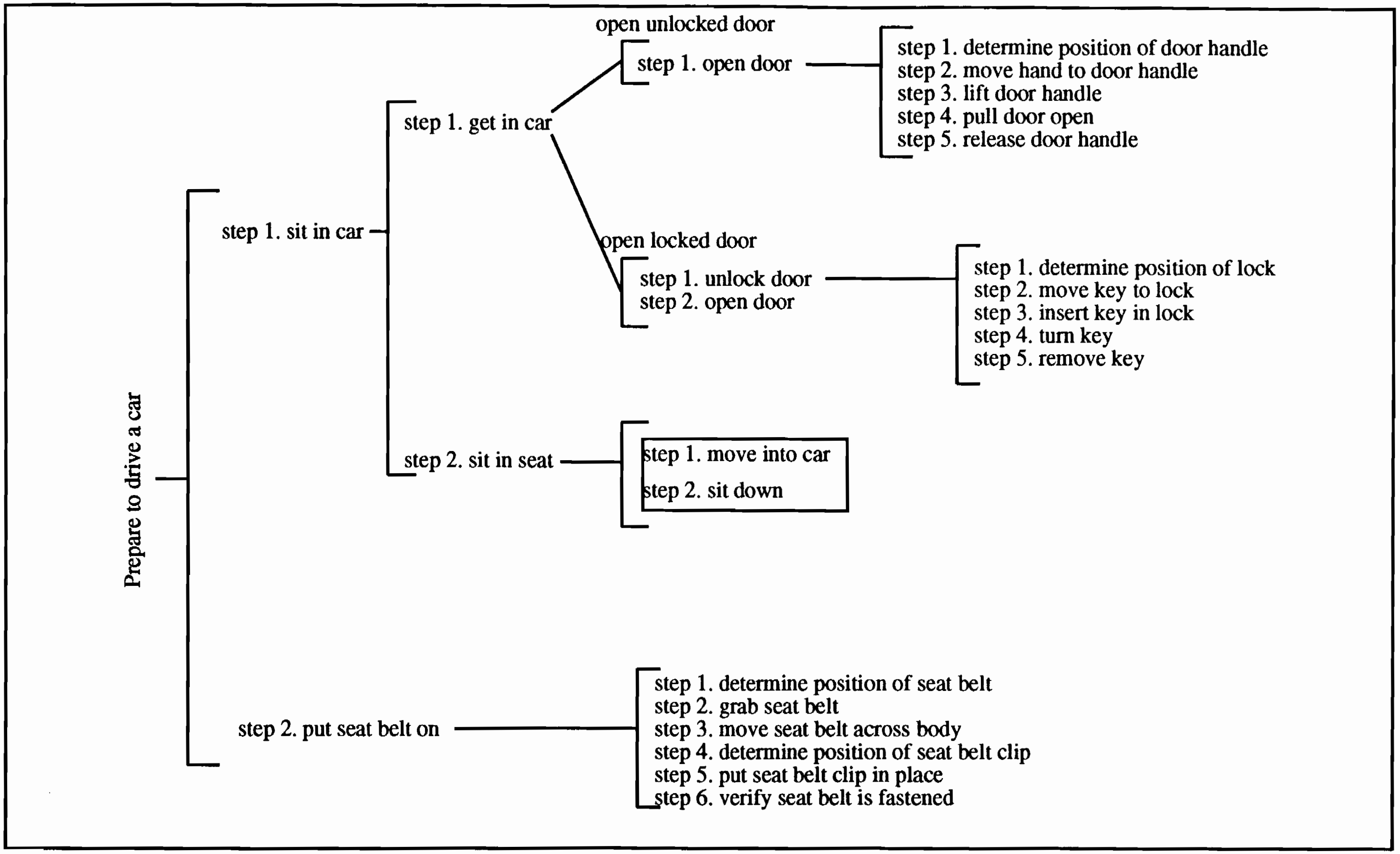
h. Complete the example problem

   Show how to move through the graph with the clicking of a goal/step and navigate commands.

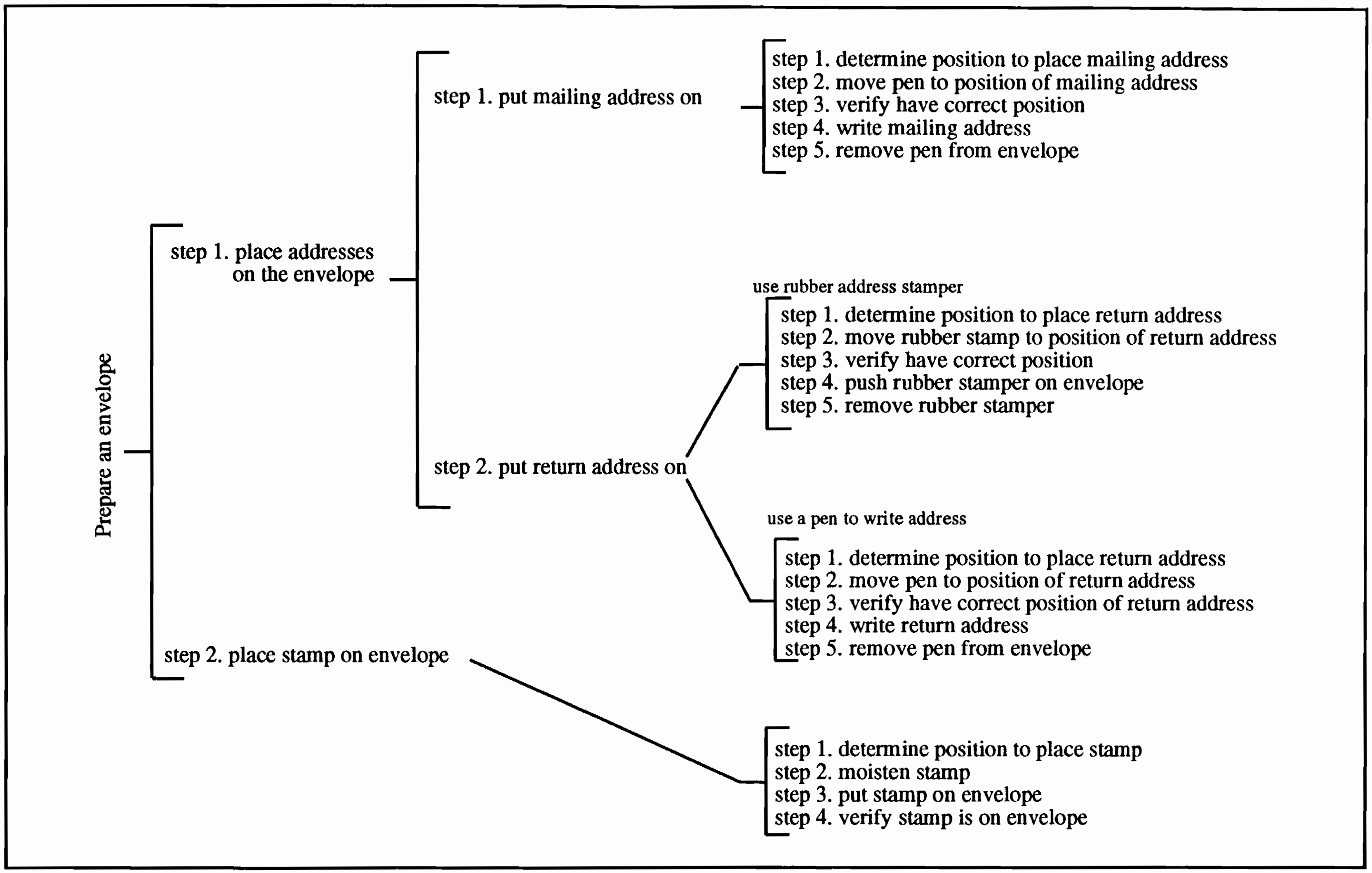
__4. Have subject do an example on his or her own. I will provide feedback on what the subject is doing right and what subject is doing wrong with specific advice on how to perform an activity the right way.

Give subject the "prepare an envelope" example task explanation sheet.

I now want you to use the tool with the following example. This example contains both physical primitives such as "put stamp on envelope" and mental primitives such as "determine position of..." and "verify..."
Extraordinary Observations of Session



Prepare to drive a car demonstration model. (Note: the steps "move into car" and "sit down" are highlighted by the box to show them as concurrent actions.)


Prepare an envelope
step 1. place addresses on the envelope
step 1. put mailing address on
step 1. determine position to place mailing address
step 2. move pen to position of mailing address
step 3. verify have correct position
step 4. write mailing address
step 5. remove pen from envelope
step 2. put return address on
use rubber address stamper
step 1. determine position to place return address
step 2. move rubber stamp to position of return address
step 3. verify have correct position
step 4. push rubber stamper on envelope
step 5. remove rubber stamper
use a pen to write address
step 1. determine position to place return address
step 2. move pen to position of return address
step 3. verify have correct position of return address
step 4. write return address
step 5. remove pen from envelope
step 2. place stamp on envelope
step 1. determine position to place stamp
step 2. moisten stamp
step 3. put stamp on envelope
step 4. verify stamp is on envelope

Mail a letter demonstration model.

APPENDIX 4 - PROTOCOL AND TASK SHEET FOR MACHINE-AIDED SESSION

Protocol for Machine-Aided Session

__0. Prepare apparatus before subject arrives.
   a. Turn computer on.
   b. Start CAT.

__1. Explain nature of session.

The purpose of this session is to elicit your knowledge and develop a model on how to perform the task I will describe. You will interact with a tool called CAT which will guide you through a process of defining the steps to accomplish a goal.

__2. Give subject explanation of the task, explain the task, and explain the example primitives for this experiment. Read and explain the task sheet. Demonstrate the primitives and "move a file to trash."

__3. Record beginning time: __________.

__4. Record ending time: __________.

__5. Make sure analysis is complete by using Guidance command.

__6. Save the file.

__7. Have subject interpret their primitive list. Print subject's primitive list. Give subject both the baseline and his or her primitive list.

I now want you to match the primitives you defined with the primitives I will give you. For each primitive you defined, pick ONE primitive from the comparison list I provide you which means the same as the primitive you defined. Place the comparison primitive number next to the primitive on your list.

__8. Exit process.
Thank you for participating in this experiment. Please do not talk to anyone about this experiment, because they may be a subject. We can talk about this experiment and the results at a later time.

__9. Pay subject and have subject sign receipt.

Extraordinary Observations of Session

Task Explanation for Machine-Aided Session

During this session you will use a tool called CAT. The tool will prompt you for information about how to accomplish a goal.

Remember to describe both the physical and mental actions necessary to perform the task.

The goal is:

Move a piece of text using cut and paste while using Microsoft Word for the Macintosh.

You can begin this session by double-clicking the CAT icon and selecting a new model with the top-level goal of "move a piece of text using cut and paste"

Assumptions

When defining the task you can assume the following:

1) the Macintosh computer is on,
2) Microsoft Word is loaded,
3) the text module is loaded, and
4) the following are examples of primitives for this session:

Physical primitives

moving the cursor
clicking the mouse button
double-clicking the mouse button
pressing a key
releasing a key

Mental Primitives

determining position of a word or text
verifying an action.

Example use of the primitives:

Goal: move a file to the trash

step 1. determine position of file
step 2. move cursor to file
step 3. press mouse button down
step 4. verify correct file is selected
step 5. move cursor to trash
step 6. release mouse button

Comparison List of Primitives

1 determine position of beginning of word
2 determine position of beginning of text
3 determine position of end of text
4 determine position of insertion point

5 move cursor to beginning of word
6 move cursor to beginning of text
7 move cursor to end of text
8 move cursor to insertion point
9 move cursor to Edit on menu bar
10 move cursor to Cut
11 move cursor to Paste

12 press mouse button down
13 double-click mouse button
14 click mouse button
15 release mouse button

16 press command/apple
17 press x
18 press v

19 release command/apple
20 release x
21 release v

22 verify correct text is selected
23 verify Cut is selected
24 verify Paste is selected.

APPENDIX 5 - INFORMED CONSENT FORM

Informed Consent Form

This form constitutes informed consent by you to participate in this study. Please read it in its entirety and then sign on the next sheet.

Project Title: Evaluation of an Automated Knowledge-Acquisition Tool

Purpose of Experiment:
The purpose of this research is to investigate the knowledge-acquisition process which is the process by which knowledge is elicited, organized, represented, refined and verified.

Procedure to be Followed in the Study:
Three sessions will be needed to conduct this experiment: 1) subject performance of the screening task, 2) machine-aided tool familiarization, and 3) machine-aided tool knowledge acquisition. Session one will be performed to verify your ability to perform the task being modeled. If you perform this task adequately then you will be used for the rest of the experiment. If you can't perform the task, then you will not be used for the rest of the experiment and you will be paid $1. Session two, the familiarization session, is needed to allow you to become familiarized with the tool before using it for the machine-aided knowledge-acquisition session. Session three is the interaction of you with the tool.

Discomforts and Risks for Participants in the Study:
The only discomforts will be from sitting and interacting with a computer for a total time of approximately one(1) hour.

Expected Benefits:
You will gain a better understanding of how you store your knowledge.

Confidentiality of the Results:
All information gathered from your responses is intended for research purposes only. Therefore, it will remain confidential and will have all indentifiers removed as soon as all responses are combined. Once you have participated in the experiment, you will be known only by a code number.

Freedom to Withdraw:
You are free to withdraw your consent to participate and discontinue participation in the experiment at any time for any reason without forfeiting pay for time spent up until withdrawal.

Use of Research Material:
The information accumulated by this research may be used for scientific or educational purposes and information relating to your responses may be presented at scientific meetings and/or published and republished in professional journals or books, or used for any other purpose which is in the interest of education, knowledge, or research.

Approval of Research:
This research has been approved by the Institutional Review Board of Virginia Polytechnic Institute and State University.

Stated Permission From Subject:

1. I have read and understand the above description of the experiment, had an opportunity to ask questions, and had them all answered, and hereby acknowledge the above and give my voluntary consent for participation in this study.

2. I understand that I am participating freely in full understanding that I need not participate if I do not wish to, and if I participate I may withdraw at any time without penalty.

3. I understand that should I have any questions about this research and its conduct, I should contact any of the following.

Researcher: Tim Kotnour (231-2939)

Faculty Advisor: Dr. K. E. Williams (231-2955)

______________________________ ______________________________
Name (Please Print) Date

______________________________ ______________________________
Signature Student ID Number

APPENDIX 6 - RESULTS FROM THE PILOT STUDY

This appendix describes the results of the pilot study. The pilot study was used to test and change the experimental procedures as necessary. The subject data from the pilot studies were not integrated into the overall data. Five subjects were run.

There were several changes made to the procedures as a result of running the pilot study. Changes included: a procedure for familiarization session, a procedure for observation of subject-tool interaction, the wording of machine-aided session task, and a procedure to interpret a subject's primitives.

1. Familiarization session.

The first change was in the procedure to familiarize the subjects with the tool. The original procedure was to have the subject interact with the tool based upon a written script he or she was given. The procedure of using a script was inadequate. The new familiarization session had three steps. The first step was for the experimenter to explain the terms used in the tool with an example model. In the second step, the experimenter used the tool with an example task, explaining each step and dialog box as the model was keyed in. In the third step, the subject was given an example task to use with the tool. The experimenter provided feedback as the subject used the tool. This new procedure was used for the last four subjects of the pilot study. The effect of this change was the subject was able to more easily and directly understand how to use the tool.

2. Observation of subject-tool interaction.

The observation of the subject-tool interaction was used to note critical incidents. For example, a critical incident would be when the subject asks the experimenter for help or when the experimenter sees the subject is having difficulties using the tool. The experimenter would aid the subject and note these difficulties. The list of difficulties was used to suggest revisions to the tool. The collection of critical incidents was done during both the familiarization and the machine-aided sessions. This change effected the experiment by having a more systematic approach to collecting incidents from which suggestions for changes to the tool could be made.

3. Wording of machine-aided session task.

Wording of tasks given to the subject was changed to emphasize parts of the experiment. For example, the machine-aided goal was changed from "move a piece of text using the cut and paste commands" to "move a piece of text using cut and paste." The effect of this change was the subject was less biased to only describe the cut and paste menu commands and not the short-cut key equivalents.

4. Procedure to interpret a subject's primitives.

There was a need to interpret what a subject meant when he or she defined a primitive because, as part of the data analysis, each subject's primitive steps were compared to the other subjects' and the baseline's primitive steps. The interpretations were necessary because the subjects do not use the exact same phrases and may even attach different meanings to words or phrases. For example, the following two primitive steps were developed by two different subjects:

Subject #4
STEP "let go of the mouse"

Subject #5
STEP "release the mouse button".

Both of these steps accomplish the same action of releasing the mouse button, but an experienced user such as the experimenter is interpreting what these rules mean. The way to overcome this interpretation was to have the subject match the primitives he or she defined in their model with primitives from the baseline model. The effect of this change was an accurate interpretation of what the subject meant when he or she defined the step.

APPENDIX 7 - DATA REDUCTION SHEETS

This appendix contains the data reduction sheets used for the data analysis. Each table corresponds to one of the six primitive methods (i.e., select text, cut text with menu, cut text with keys, new position, paste text with menu, and paste text with keys) and contains the primitive steps of the given primitive method. Each primitive step is described by its source (i.e., baseline or subject) and type (i.e., mental or physical). The values in the table are used to denote if the subject defined the primitive in the given primitive method; 0 denotes the subject did not define the primitive step for the given primitive method and 1 denotes the subject did define the primitive step for the given primitive method.

Method: Select Text

Primitive			Subjects																			
Source	Type	Name	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21
S	M	determine text to cut	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
S	M	scan paragraph	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
S	M	verify found right text	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
S	P	move mouse to find I bar	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
S	M	detect mouse	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
B	M	determine begin of text	1	1	0	1	1	1	0	1	0	1	1	0	1	0	1	1	1	1	1	1
B	P	move cursor to begin of text	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1
B	P	press mouse button	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1
B	M	determine end of text	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
B	P	move cursor to end of text	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	P	release mouse button	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0	0	0	1	1	1
S	M	verify text was highlighted	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
B	M	verify correct text selected	0	0	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0	0	0	1

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Select Text

Primitive			Subjects																			
			2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4
Source	Type	Name	2	3	4	5	6	7	8	9	0	1	2	3	4	6	7	8	9	0	1	2
S	M	determine text to cut	0	0	0	0	1	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0
S	M	scan paragraph	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	verify found right text	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	P	move mouse to find I bar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	detect mouse	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	M	determine begin of text	1	1	1	0	0	0	1	1	1	1	0	1	1	1	0	1	1	1	0	1
B	P	move cursor to begin of text	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
B	P	press mouse button	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	M	determine end of text	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
B	P	move cursor to end of text	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
B	P	release mouse button	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S	M	verify text was highlighted	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	M	verify correct text selected	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Cut Text with Menu

Primitive			Subjects																			
Source	Type	Name	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21
S	M	locate cursor	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S	M	look at menu bar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine menu to use	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine 'edit' position	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
S	P	move cursor to menu	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
B	P	move cursor to 'edit'	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
B	P	press mouse button	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1
S	M	determine menu item to use	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine position of cut	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
B	P	move cursor to cut	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
B	M	verify cut is selected	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1
B	P	release mouse button	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
S	M	verify text is cut	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Cut Text with Menu

Primitive			Subjects																			
			2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4
Source	Type	Name	2	3	4	5	6	7	8	9	0	1	2	3	4	6	7	8	9	0	1	2
S	M	locate cursor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	look at menu bar	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S	M	determine menu to use	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine 'edit' position	1	0	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0
S	P	move cursor to menu	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
B	P	move cursor to 'edit'	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
B	P	press mouse button	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1
S	M	determine menu item to use	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine position of cut	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	move cursor to cut	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
B	M	verify cut is selected	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	release mouse button	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1
S	M	verify text is cut	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Cut Text with Keys

Primitive			Subjects																			
Source	Type	Name	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21
S	M	determine keys to use	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	locate command key	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S	P	move finger to command key	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	Press Command	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S	M	locate x key	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S	P	move finger to x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	Press X	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
B	P	Release X	0	0	1	0	1	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1
B	P	Release Command	0	1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
S	M	verify text is cut	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Cut Text with Keys

Primitive			Subjects																			
			2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4
Source	Type	Name	2	3	4	5	6	7	8	9	0	1	2	3	4	6	7	8	9	0	1	2
S	M	determine keys to use	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	locate command key	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S	P	move finger to command key	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
B	P	Press Command	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
S	M	locate x key	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S	P	move finger to x	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
B	P	Press X	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
B	P	Release X	0	0	1	1	1	1	1	0	0	1	0	1	1	1	1	0	1	1	0	0
B	P	Release Command	0	0	1	1	1	1	1	0	0	1	0	1	1	1	1	0	1	1	0	0
S	M	verify text is cut	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: New Position

Primitive			Subjects																			
Source	Type	Name	1	2	3	4	5	6	7	8	9	1 0	1 1	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1
S	P	move mouse to determine position	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
B	M	determine new position	1	1	1	1	1	0	1	0	0	0	1	0	1	0	1	0	1	1	1	1
B	P	move cursor to new position	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
B	P	click mouse	1	1	1	0	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	1
S	M	verify correct position	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: New Position

Primitive			Subjects																			
			2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4
Source	Type	Name	2	3	4	5	6	7	8	9	0	1	2	3	4	6	7	8	9	0	1	2
S	P	move mouse to determine position	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	M	determine new position	1	0	1	1	1	0	0	0	1	1	1	1	1	1	0	1	0	1	0	1
B	P	move cursor to new position	0	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	0	1	1
B	P	click mouse	0	0	1	1	1	0	1	0	0	1	1	1	1	1	0	0	1	0	0	0
S	M	verify correct position	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Paste Text with Menu

Primitive			Subjects																			
Source	Type	Name	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21
S	M	locate position of cursor	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S	M	look at menu bar	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine menu to use	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	P	move cursor to menu	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
S	M	determine position of edit	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
B	P	move cursor to edit	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
B	P	press mouse button	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1
S	M	determine menu item to use	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine position of paste	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
B	P	move cursor to paste	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
B	M	verify paste is selected	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
B	P	release mouse button	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
S	M	verify text is inserted	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Paste Text with Menu

Primitive			Subjects																			
			2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4
Source	Type	Name	2	3	4	5	6	7	8	9	0	1	2	3	4	6	7	8	9	0	1	2
S	M	locate position of cursor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	look at menu bar	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
S	M	determine menu to use	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	P	move cursor to menu	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
S	M	determine position of edit	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
B	P	move cursor to edit	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
B	P	press mouse button	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1
S	M	determine menu item to use	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	M	determine position of paste	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	move cursor to paste	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
B	M	verify paste is selected	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	release mouse button	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1
S	M	verify text is inserted	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Paste Text with Keys

Primitive			Subjects																			
Source	Type	Name	1	2	3	4	5	6	7	8	9	1 0	1 1	1 3	1 4	1 5	1 6	1 7	1 8	1 9	2 0	2 1
S	M	locate command	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S	P	move finger to command	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	press command	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
S	M	locate V	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
S	P	move finger to V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	P	press V	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
B	P	release V	0	0	1	0	1	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1
B	P	release command	0	1	1	0	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
S	M	verify text is inserted	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

Method: Paste Text with Keys

Primitive			Subjects																			
			2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	4	4	4
Source	Type	Name	2	3	4	5	6	7	8	9	0	1	2	3	4	6	7	8	9	0	1	2
S	M	locate command	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S	P	move finger to command	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
B	P	press command	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
S	M	locate V	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
S	P	move finger to V	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
B	P	press V	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
B	P	release V	0	0	1	1	1	1	1	0	0	1	0	1	1	1	1	0	1	1	0	0
B	P	release command	0	0	1	1	1	1	1	0	0	1	0	1	1	1	1	0	1	1	0	0
S	M	verify text is inserted	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0

NOTE: For Source: "S" = primitive defined by subjects, "B" = primitive defined by baseline model.
For Type: "M" = mental primitive, "P" = physical primtive.
In the table: "0" = subject didn't define the primitive, "1" = subject did define the primitive.

APPENDIX 8 - DESCRIPTION OF CAT

Cognitive Analysis Tool (CAT) is the automated knowledge-acquisition tool designed, developed, tested, and evaluated in this research. CAT is an application which runs in the Microsoft Windows 3.0 environment on an IBM PC or compatible. CAT is based on the GOMS model of describing cognitive task-based (i.e., procedural) knowledge as proposed by Card, Moran, and Newell (1983).

A practical implementation of the GOMS analysis technique begins in a top-down breadth-first manner. That is, first a top-level goal is specified by the user as something the user wishes to accomplish. Next, the user is requested to specify a method consisting of a series of steps to be executed to accomplish the top-level goal. At this point, the series of steps are of a high level and consequently this top-level method must be decomposed further. Having specified a top-level method, the user is requested to specify any alternative methods that would accomplish the same goal. If an alternative method is specified, then the user identifies the method's steps as before. Following the specification of all alternative methods for accomplishing the top-level goal, the user must then develop a selection rule or a set of selection rules. Each selection rule identifies a set of preconditions that discriminate between methods triggering which method is to be selected under differing contextual conditions to accomplish the goal.

Having specified a top-level goal, a top-level method, its alternatives, and a selection rule(s), the system then converts each step in the top-level method into a subgoal. A set of steps comprising a method is then requested of the user for each subgoal. If alternative methods for accomplishing a subgoal exist, then a selection rule must be specified as before. This process creates a goal-subgoal hierarchy with appropriate selection rules. Method steps are continuously transitioned to subgoals requiring lower-level methods and selection rules. The process terminates when all steps for each method have been designated as primitives. A primitive step can be identified by the user at any time indicating that the step does not require any further decomposition as a subgoal. Upon completion of this analysis a set of procedures (i.e. rules) have been defined that can be executed by a production system.

CAT employs the structured interview method to guide the user through the process of defining the procedural knowledge of a task. This structured-interview technique is implemented through a series of dialog boxes that prompt the user for specific information. The dialog boxes employed by CAT are: top-level dialog box, used to define the top-level goal; method editor dialog box, used to define the set of steps to accomplish a goal; selection rule editor dialog box, used to define the selection rule for alternative methods; and yes-no dialog boxes, used to ask the user if alternative methods or selection rules exist and if a goal is a primitive or not.

The top-level goal, method editor, and selection rule editor dialog boxes each have field boxes that allow the user to type text describing the required information. These dialog boxes also have buttons that the user can select with the mouse to activate a command such as "help." The yes-no dialog boxes use buttons to capture the user's response (i.e., yes, no, cancel, help) to the question being asked by the yes-no dialog box. All of the dialog boxes cover part of the screen with the remaining screen section displaying a graphical representation of the goal-subgoal hierarchy.

VITA

Timothy G. Kotnour

Education

Master of Science in Industrial and Systems Engineering, Management Systems Engineering, Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, Va. 9/92
G.P.A. in major 3.72/4.0, overall 3.64/4.0

Bachelor of Science in Industrial Engineering, Bradley University, Peoria, Ill. 5/90
Minors: Business Administration and Computer Science
G.P.A. in major 3.95/4.0, overall 3.84/4.0

Professional Registration

Registered Professional Engineer-In-Training, State of Illinois (1990)

Professional Experience

Graduate Research Assistant, *Management Systems Laboratories and Systems Engineering Design Laboratory*, Virginia Tech, 8/90 to Present

Engineering Intern, Performance Improvement Engineering, *Eli Lilly & Company*, Indianapolis, Ind., 5/89 to 8/89

Quality Control Inspector (3rd Shift), Quality Control Department, *Andrew Corporation*, Orland Park, Ill., 5/88 to 8/88

Management Systems Consultant, *Silver Cross Family Care Center*, Lockport, Ill., 5/88 to 8/88

Management Systems Consultant, *Marian Realty, Inc.*, Chicago, Ill., 5/87 to 8/87

Special Projects

Development of Transition Services with Floyd County School District
Designed and developed Transition Services with Floyd County Special Education Coordinator

Chairman, 1990 IIE Student Regional Conference
Directed and organized conference hosted at Bradley University involving 13 universities and over 170 participants, industry sponsored 70 percent of $10,000 budget

Senior Project-Bradely University
Analyzed and recommended scheduling procedures and manpower requirements at the Peoria Police Department, resulting in annual savings of at least $100,000

Paper and Presentations

Williams, K. E., and Kotnour, T. G., "Knowledge Acquisition Methodologies: A Review and an Analysis of Techniques for Intelligent Tutoring Applications," Office of Naval Research Technical Report, in preparation.

Williams, K. E., and Kotnour, T. G., "A Knowledge Center for Group Problem Solving and Formulating Planned Strategy," SE TIMS, October, 1992.

Williams, K. E., Deighan, J., and Kotnour, T. G., "Knowledge Acquisition for Group Problem Solving Systems," 14th Conference for Computers and Industrial Engineering, March, 1992.

Kotnour, T. G., "Intelligent Management Tool to Aid Problem Solving, Decision Making, and Planning," unpublished research grant working paper, January, 1992.

Kotnour, T. G., and Williams, K. E., "Automated Knowledge Acquisition for Second Generation Knowledge Based Systems: A Conceptual Analysis and Taxonomy," SE TIMS, October, 1991.

Kotnour, T. G., "Evaluation of an Overtime Management System for the Peoria Police Department," IIE Student Conference, March, 1990.

Kotnour, T. G., Schmelzer, B., and Lizam, "Evaluation of an Overtime Management System for the Peoria Police Department," City of Peoria Police Department Executive Committee, December, 1990.

Kotnour, T. G., "Facility Layout and Location at Marian Realty, Inc.," IIE Student Conference, 1985.

Timothy G. Kotnour