Rochester Institute of Technology
RIT Digital Institutional Repository

Theses

1990

Automated knowledge acquisition tool for identification of generic tasks

Arlene J. Buck

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation
Buck, Arlene J., "Automated knowledge acquisition tool for identification of generic tasks" (1990). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

Rochester Institute of Technology
School of Computer Science and Technology

Automated Knowledge Acquisition Tool for Identification of Generic Tasks

by

Arlene J. Buck

A Thesis, submitted to
The Faculty of the School of Computer Science and Technology
In partial fulfillment of the requirements for the degree of
Master of Science in Computer Science.

Approved By:

Professor John A. Biles

Professor Hugh Wood

Professor Peter Anderson

June, 1990

Short Title Identify Generic Tasks

I giv my permission for reproduction of this thesis for non - profit use.

by

Arlene J. Buck

Abstract

Recent research in the Knowledge Acquisition (KA) field, centers on defining a formal methodology for the KA process. This research includes the following goals: automating the KA process to decrease the KA time constraint; applying psychological techniques to extract the underlying structure of the expert's knowledge; and defining expertise in terms of "generic tasks" to yield possible knowledge organizations and strategies for the implementation of the expert system.

This thesis provides an overview of the benefits and concerns of an automated KA system, psychological scaling techniques as they apply to KA, and the relevance of generic tasks. A generic task defines a knowledge type and organization, and a control strategy that characterizes a component of an expert system.

This thesis also includes the design and implementation of a Knowledge Acquisition Tool for Identification of Generic Tasks. This tool provides an interface to the expert for the initial KA encounter. Using psychological techniques, the tool extracts a list of the main concepts of expertise, and elicits a rating from the expert comparing the similarity of each of these concepts to generic task concepts. The results become inputs to a clustering technique that organize the concepts into the generic tasks. The result of any concepts that do not cluster could identify a previously undefined generic task. The implementation is in the C language, accessing the FASTCLUS procedure of the SAS software package for VAX hardware.

Table Of Contents

Chapter 1
Introduction

Artificial Intelligence encompasses the study of Expert Systems. An Expert System is computer software that attempts to emulate an expert's knowledge, problem solving strategies, and decision making abilities. In the development of an Expert System, a knowledge engineer's main task is the extracting of the knowledge, strategies and abilities from the expert, termed Knowledge Acquisition (KA). Much of the research in knowledge acquisition is geared toward a formal methodology of this task.

This research has included: automation of the KA process, the application of psychological techniques to the KA process, and the definition of expertise in terms of generic tasks.

During the KA process, communication between the expert and the knowledge engineer is time consuming, and the extracted knowledge is open to subjective interpretation by the knowledge engineer. Research into an automated KA tool addresses these problems.

Psychological techniques applied to KA include personnel construct theory, interview, protocol analysis, clustering, and multidimensional scaling. All techniques strive to uncover the underlying structure of the expert's knowledge.

At the time of this study, generic tasks have been identified as possible building blocks for an expert system. Each generic task defines a knowledge structure and strategies for control of that structure. The expertise can be defined as any number of these generic tasks connected by some message passing scheme.

An automated KA tool for identification of Generic Tasks is defined in this study. The tool extracts from the expert a list of concepts representing his expertise. The tool prompts the expert to rate these concepts comparing their similarity to each generic task's concepts. These results are the inputs to a clustering technique that clusters the expert's concepts around generic tasks. The implementation is in the C language and uses the FASTCLUS procedure of a SAS software package for VAX hardware.

This study gives the background of the benefits and concerns of an automated knowledge acquisition tool, an overview of generic tasks, and an overview of psychological techniques applied to KA. An implementation of an automated tool to identify generic tasks using psychological techniques is presented.

Chapter 2 provides the theoretic basis of automated KA tools, psychological techniques apply to KA, and generic tasks.

Chapter 3 discusses the concepts of the psychological scaling techniques and generic tasks chosen for the automated KA tool for identifying generic tasks.

The details of the implementation of the automated KA Tool for identifying generic tasks are discussed in Chapter 4. Chapter 5 contains examples using the automated KA Tool for identifying generic tasks. Sample programs, results and a discussion accompany the examples. Chapter 6 is a discussion of the overall conclusions of this study. The appendix contains information that users of the automated KA Tool for Identifying Generic Tasks will find useful. These materials include Knowledge Engineer's and Expert's guides which describe the usage of the Automated KA Tool to Identify Generic Tasks.

Chapter 2
Background

2.1 The Benefits and Concerns of an Automated KA Tool

The primary benefit of an automated KA tool is the shortening of the time required for the KA process. ". . since manual methods for acquiring strategic knowledge push the limits of human cognitive abilities, automated methods for acquiring strategic knowledge will be particularly important to expert system development" [HAYE83, 158]. The KA process requires communication between the expert and the knowledge engineer; the contents of this communication is open to subjective interpretation by the knowledge engineer. On the other hand, KA communication between an automated KA tool and an expert is not open to subjective interpretation by the automated tool. For example: the automated tool does not become weary over time, is not affected by moods or esthetic changes in its environment, and can't have personality conflicts with the expert.

Each of the present automated KA tools such as ETS, MDIS, MORE, SALT is limited to a particular class of expert systems. For example, "ETS interviews experts for knowledge on classification problems," [BOOS86, 31]. If the expert system does not fit into one of the classes defined for an existing tool or if the expert system is a combination of these classes, it cannot be built using an automated tool. There also exists the possibility that use of an automated tool will yield a system that doesn't adequately match human expert knowledge. This is due to the use of structures and strategies mandated by the tool that do not adequately reflect the structures and strategies used by the expert.

An automated tool reduces the time a Knowledge Engineer would meet with an expert. During the time of the automated KA process, the Knowledge Engineer could prepare for the KA process by becoming familiar with the jargon and terminology of the domain. By predefining the domain in terms of generic tasks, the Knowledge Engineer can modify his other KA tools for the knowledge and strategies in those tasks.

The generic tasks encompass all ranges of knowledge representations and strategies. Therefore, an Automated Tool based on generic tasks would not be susceptible to a negative tool bias.

2.2 The Generic Task Approach

Research involving the knowledge of experts has yielded the observation that this knowledge consists of more than one specific kind of knowledge. ". Expertise in a technical domain comprises knowledge of more than one kind, not all of which can reasonably be represented in the form of empirical rules" [GAMM85, 105]. Therefore it is conceivable for the expert system representation of the knowledge to be a mixture of rules, frames, semantic networks, and other knowledge representations currently used by knowledge engineers. The difficulty is how to express this mixture formally.

One solution refers to "the character of the system's tasks as major factor in determining which language or system to use when formalizing the domain knowledge" [HAYE83, 133]. Chandrasekaran's group formally expressed this mixture as any combination of "generic tasks" connected by a message passing scheme. At this time, there are 6 identified generic tasks: Hierarchical classification, hypothesis matching, object synthesis by plan selection and refinement, state abstraction, knowledge directed information passing, and abductive assembly of hypothesis. Each generic task "can be characterized by providing information about (1) task specification in the form of generic types of input and output information; (2) specific forms, which include the basic pieces of domain knowledge needed for the task and specific organizations of this knowledge particular to the task; and (3) a family of control regimes appropriate for the task. [CHAN86, 24].

A description of each task follows:

1) Classification - to classify a situation as an element of a hierarchy.

The hierarchy is composed of elements. Each element contains a partial description of a situation that is possible in the hierarchy, along with the necessary evidence for the confirmation or lack of confirmation of a partial description of an input situation to match the element's partial description. The elements are arranged so that elements describing more general information are at the top of the hierarchy, while the lowest level contains elements describing specific knowledge. The goal of classification is identification.

The strategy for classification is top down, or establish refine, and the process begins at the highest level in the hierarchy. If a partial description of the situation matches an element, the process continues through the successive lower levels of the hierarchy trying to match a more specific

element. The process succeeds if the entire situation can be covered in the hierarchy.

An example of this type of task is: Classify a patient's medical observations as an element of a disease hierarchy. Each symptom would be treated as a partial input situation. The hierarchy would have general knowledge of a heart, liver, etc. at the top. The lowest level would contain knowledge of a particular diagnosis like hepatitis. The strategy would begin at the general knowledge level and move down the hierarchy matching symptoms to elements of the hierarchy. This continues until all symptoms are accounted for and a particular diagnosis is reached.

2) State Abstraction - provide an account of any changes to a system process as a result of a state change of any component of the system.

Each component of the real system is represented by a block of knowledge about that component. The entire system is represented by the hierarchical connections between the knowledge blocks. The hierarchy forms components into subsystems and subsystems into the final system. In this way, the KA representation mirrors the real system, achieving a qualitative simulation.

The strategy begins at the knowledge block representing the changed component. The knowledge is modified in that block to match the state change and the modification then radiates out through all the connections of that block to other blocks. The modifications move up the hierarchy until the top of the system is reached and stabilizes. The strategy is bottom up, and follows the architecture of the real - world subsystem.

An example is what would happen to the nuclear power plant process if valve A is closed? The knowledge block containing valve A would be modified to be closed. All knowledge blocks affected by valve A closing would be modified until the entire nuclear power plant process stabilizes.

In the classification hierarchy, the control moved from general to specific knowledge and the goal is identification. In this strategy, the control moves from specific to general knowledge and the goal is to simulate.

3) Knowledge Directed Information Passing - to identify knowledge of any concept related to a given concept.

The concept is represented by a frame which lists slots for all of its attributes. The frames are then arranged in a hierarchy representing IS_A or PART_OF relationships between two concepts. Each attribute slot contains the attribute or a method for finding that attribute. These methods are termed inheritance, demons, or default values. Inheritance is a path to a more generalized concept from which the given concept may "inherit" the attribute. Demon is a procedure which produces the attribute from other knowledge. Default values exist if there is no other way to determine the attribute.

The strategy for the search begins at the given concept's frame. If the information is not directly available, the inheritance slot (link) is traversed. If unsuccessful, any demons to "query other concepts in other parts of the hierarchy for values" [CHAN85, 297]. Lastly, the default value is used.

An example of Knowledge Directed Information Passing: If a patient has a liver ailment it is necessary to know if he had been exposed to anesthesia. The anesthesia frames can have attribute slots for "had anesthesia" and "had surgery" This relates surgery to anesthesia so that surgery frame is queried.

The goal of knowledge directed information passing is to identify relationships between the knowledge concepts.

4) Object Synthesis by Plan Selection and Refinement Design an object satisfying the given specifications

The object is represented by a hierarchy of elements. Each element contains plans for the design of a particular component of the object. The links of the hierarchy mirror the possible connections of the components to imply the real object. Knowledge of possible plans exists to aid in the initial choice of components.

The strategy is top down and recursive until a design is realized that satisfies all given specifications. A component is chosen using the precompiled knowledge of existing plans. A plan of design for that component is chosen based on the given specifications. This plan choice suggests other components to be chosen. One component is chosen and a plan using that component is chosen. The design is based on the given specifications. If a component is chosen whose plans for design cannot satisfy the given specifications, a failure occurs. This failure rolls back to the last successful component, and other component choices are explored. If those

components also fail, the chosen plan of the last successful component is changed and this process continues.

An example is a design of a furnace with specifications of a 95% efficiency rating of heat given off during combustion to the heat that is transferred to the building. Each "component" is a possible element of the furnace, and its possible settings. The strategy chooses an element and then a possible setting. These choices lead to another element and possible setting. Failures to meet the specifications roll back this process. This continues until the design includes the elements and settings necessary for the furnace to be built within specifications.

The goal of Plan Selection and Refinement is to link the knowledge in the "best" possible arrangement.

5) Hypothesis Matching to match a concept (plan) against relevant information and determine if it is close enough for success ("goodness of fit") ("degree of likelihood") ("degree of appropriateness").

The knowledge is organized into a hierarchy. The top level of the hierarchy defines a concept in terms of other more specific concepts. The "goodness of fit" for this concept then becomes some combination of the "goodness of fit" of each subconcept. The lowest levels of the hierarchy actually match the input data to the subconcepts of that level within its "goodness of fit"' specification.

The strategy is to begin at the top matching a concept to the top level concept. The control continues to the lowest level where an attribute of the input concept is matched to a subconcept of the hierarchy. The "goodness of fit" is calculated and passed back up through the hierarchy combining with the "goodness of fit" of other attributes of the input concept. The top layer then calculates the overall "goodness of fit" and returns the results.

No specific example of hypothesis matching is given because " the problem of matching hypothesis against data is a general subtype of reasoning useful in a number of different contents." [CHAN86,25]

6) Aductive Assembly of Explanatory Hypothesis construct the best composite explanatory hypothesis given a number of hypotheses each having an associated degree of belief. "The knowledge is causal or other relations

(such as incompatibility, suggestiveness, special case of) between the hypothesis and relative significance of data items." [CHAN86,26]

The knowledge has no organizational structure if the set of hypothesis is small. If the set is large, the hypothesis are organized at different levels of abstraction of the assembled hypothesis.

The strategy is a "means - end" regime driven by the goal of explaining all the significant findings." [CHAN85,298] A composite explanation is assembled by choosing the best hypothesis that explains the latest significant information adding it to the latest composite, and then removing any "explanatory superfluous parts." [CHAN86,26] This continues until all information is explained or all hypothesis are exhausted. The goal is a complete explanation of the hypothesis.

A Survey of the present well known expert systems can be presented in terms of generic tasks. Mycin uses classification, and Intermist uses abductive assembly of hypothesis. Prospector uses classification, while Dendral uses hypothesis matching and abductive assembly of hypothesis.

Chandrasekaran and associates are currently developing representation languages for each generic task. CSRL exists for classification and DSPL exists for object synthesis by selection and refinement. HYPER for Hypothesis Matching is under development along with PEIRCE for abductive assembly. "a simple version of the state abstraction task language is available now." [CHAN86, 29]. Several systems have been developed using generic task philosophy:

RED An expert system for interpreting data for red blood cell identification and uses 4 generic tasks. [JOSE85]

Jesse an expert system which models some aspects of Japanese energy policy decision making and uses 2 generic tasks. [GOEL87]

Another system is a toolkit of generic tasks for use in the development of an aid for operators of nuclear power plants. [HAJE88] It uses 2 generic tasks.

Chandrasekaran states "If we can perform an epistemic analysis of the domain such that 1) the complex task can be decomposed in terms of generic tasks, (2) paths and conditions for information transfer from the agents that perform these generic tasks to the others which need the information can be established and (3) knowledge of the domain is available to encode into knowledge structures for the generic tasks, then the complex task can be

"knowledge engineered" clearly and successfully. [CHAN85, 28] In that same article, Chandrasekaran goes on to state ". . the reason we are not yet able to handle difficult design problem solving is that we are often unable to find an architecture of generic tasks in terms of which the complex task can be constructed." [CHAN86, 28]

Chandrasekaran states the benefits of generic tasks as follows:

- ease and clarity of expert system design and implementation
  KA at a more conceptual level than rules, frames, networks, etc
- clear explanations of problem solving strategies
  uncertainty handling is viewed as consisting of different types of each kind of problem solving rather than as a general method across all kinds of problem solving.
- the ability to create a representation language that implicitly contains the control regime and knowledge representation associated with a task.

Generic tasks have proven a successful methodology in several expert systems. Red, Jessie and MDX are examples of expert systems implemented with generic tasks that are on the market today. In each system, an analysis of the domain yielded the identification of several generic tasks. Using these identified generic tasks, the knowledge engineer extracted further domain knowledge and implemented these systems using CSRL or DSPL languages. These systems have proven the success of identifying a domain in terms of generic tasks. To determine which generic tasks comprise a given domain, comparison of concepts of the domain could be compared to the previously stated characteristics of generic tasks.

2.3 Psychological Techniques Applied to KA

2.3.1 Introduction

The task of constructing a model of the expert's knowledge is very difficult. Presently, knowledge engineers rely on informal interviews with feedback driven refinement. This methodology implies an extraction of knowledge from the expert, an interpretation of that knowledge by the knowledge engineer, and an editing of that interpretation by the expert for discrepancies which suggest modifications or additions to the interpretation. This process is time consuming, and is insufficient for uncovering any

knowledge that is compiled (unconscious processing) or procedural in nature (contains heuristics, rules, or strategies).

Another methodology presently used is protocol analysis. Protocol Analysis is the recording of the behavior exhibited by an expert as he solves a problem. This recording is then transcribed and analyzed by the knowledge engineer. His interpretation of the knowledge is refined by the expert as in interviews. Protocol Analysis goes beyond the knowledge that the expert can verbalize to a knowledge inferred by the actions of the expert. This method is also time consuming and open to subjective interpretation of the observations by the knowledge engineer. Current research is investigating other psychological techniques for extracting the knowledge from the expert.

2.3.2 Personal Construct Theory

KA research has looked to psychology to help define and extract knowledge.

In 1955, George Kelly stated:

"Man creates his own ways of seeing the world in which he lives; the world does not create them for him. He builds his constructs and tries them on for size. His constructs are sometimes organized into systems, groups of constructs which embody subordinate and superordinate relationships. The same events can often be viewed in the light of two or more systems yet the events do not belong to any system. Moreover, man's practical systems have particular foci and limited ranges of convenience." [KELL55,12]

Kelly's personal constructs led to the use of Repertory grids. The repertory grid is a representation of the expert's view of his domain. The grid is composed of an element and a characteristic on which the element can be rated on a subjective scale (ie. 1 to 5). This scale is used for all characteristics. For example an element is a person and the characteristics are friendly and heavy. Each have a rating of 1 to 5 where 1 is "not" and 5 is "very" The expert reevaluates this grid until it represents his view. Boose's Expertise Transfer System (ETS), an automated KA process, uses Repertory Grid Testing. The ETS system cannot elicit deep causal knowledge and "is best suited for analytic problems whose solutions may be based on production systems. The system can not readily handle synthesis class problems or problems which require a combination of analysis & synthesis." [BOOS83,32] This system has

not been abandoned; enhancements are being added, but the repertory grid is limited by "creating a static representation of the expert's knowledge and the information obtained is more of a basis for discussion than a knowledge base for an automated reasoning system" [CHIG86, 66]

2.3.3 Psychological Scaling & Clustering Methods

Psychological Techniques have been applied to Knowledge Acquisition for modeling and extracting the expert's knowledge. The results have exposed the expert's knowledge and the interrelationships of the characteristics of that knowledge.

Multidimensional scaling (MDS) identifies similarities among concepts of the expert's knowledge and groups them conceptually. MDS analyzes judgments of psychological distance between each concept in the domain and every other concept in the domain. "It represents these domain concepts in an n-dimensional space. Each dimension in the spatial model can be interpreted as a characteristic used by the expert to internally organize the domain concepts. Within the MDS space, concept clusters may be evident, revealing the expert's conceptual grouping of items. "We propose that techniques such as MDS can be used to uncover natural representations for actions on objects of task space." [MANH86, 937] MDS requires only a simple judgment of the similarity of pairs of objects. "In carrying out multidimensional scaling, however, there are a number of practical problems. First, how many dimensions are there in the underlying geometry. . Second, how do we know when we have a good solution (there are a number of measures of goodness of fit based on the general idea of whether the distances implied in the spatial configuration are consistent with the initial similarity or distance assessments made by the judge)? Third, how should you rotate the axes so that they best represent the salient attributes?" [CHIG86,67]

- Link Weighted Network Scaling Techniques produce a network with nodes representing the concepts of the domain, and links representing the relationship (IS_A, PART_OF) between the concepts. This technique uses estimates of psychological proximity. There exists similar techniques analyzing psychological distance judgments to produce a tree structure. The leaf nodes represent the concepts and the branches represent a set of

characteristics shared by all subordinate nodes. The length of the branch indicates the importance of those characteristics to the expert.

- Cluster Analysis is a numerical technique that groups concepts together if they are similar in their psychological distance judgments. The definition of the groups suggests different clustering techniques. The groups' definition can be disjoint, hierarchial, overlapping, or fuzzy. The assignment of a concept to a group changes for each definition.

The representation of the concept can change in cluster analysis. A matrix in which each row and column can correspond to a concept is one representation. Another representation is a matrix in which the rows are the concepts and the columns are variable attributes of that concept. The concepts or the variable attributes can be clustered. Clustering can create a hierarchial or hetarchial classification. "Cluster analysis has long been of interest to workers in pattern recognition and a variant of the method has been promoted as a machine learning tool." [CHIG86,70]

There also exists other techniques for unidimensional scaling: ranking, rating, sorting.

Ratings, although simplistic, are subjective. The expert is given a scale and then assigns a scale value to each concept. The scale must contain a verifiable criterion or the ratings become meaningless.

Ranking is fast and simplistic. In ranking, the expert judges each concept to all others in the domain. This method is unacceptable for large concept sets.

Sorting separates concepts by their different values of an attribute. This method is successful if an attribute is separable; for example, color is not a separable attribute, for the expert must consider its hue, brightness and saturation.

All psychological techniques depend on identification of the concepts in the domain. There exist several techniques for eliciting concepts: concept listing, interview, step listing and chapter listing. In concept listing, the expert lists all the concepts relevant to the domain. Interview consists of interaction with the expert and noting all the concepts mentioned. In Step Listing, the expert lists the steps he takes to solve the problems in the domain, regardless of order. The concepts are extracted from these steps. In Chapter Listing, the expert lists the chapter titles and subtitles for a book he is to write about his domain.

The Psychological Techniques mentioned aid in eliminating subjective interpretation by the Knowledge Engineer in determining the structure of a domain. The preliminary extraction of the domain concepts uses chapter listing, concept listing, interview and step listing. In a study by Cooke, Chapter Listing generated a large number of concepts [COOK86, 1428]. Interview also generated a large number of concepts but required direct communication between expert and knowledge engineer. The chapter listing technique could easily be automated.

For the analysis of the concepts, multidimensional scaling is very complicated, and requires software difficult to acquire and use. Cluster Analysis is more readily available, and easy to use. Cluster Analysis places objects into groups such that these objects are similar to each other. The definition of that similarity suggests different clustering techniques. For the study, the similarity will be between the domain concepts and the generic task characteristics.

Chapter 3
Implementation

3.1 Introduction

The automated KA tool for identification of generic tasks consists of three components: the generation of the domain concepts, the similarity rating of the domain components to the concepts of each generic task, and the use of a clustering algorithm to analyze the ratings by clustering the domain concepts around generic tasks. The generation of the domain concepts such that all possible domain concepts are included would be ideal. Rather than implement exhaustive methodologies for this ideal, effort will be directed to the rating and analysis components. This decision was supported by the desire that this tool be a preliminary tool of the knowledge engineer in acquiring information about the domain instead of the only tool used by the knowledge engineer.

3.2 Implementation

In this study, an attempt was to provide an adequate generation of domain concepts, a thorough similarity rating scheme, and an acceptable clustering algorithm to yield a reasonable breakdown of the domain in terms of generic tasks. This implementation extracted the concepts of the domain from the expert using the chapter listing technique. These concepts were rated by the expert as to their similarity to the generic task concepts. A similarity rating helped to decrease the bias and vagueness of asking "Do you think that you have a classification generic task." These ratings become the inputs into a Clustering Algorithm that finds the domain generated concepts which match a specific generic task. The use of a clustering algorithm reduces the possibly subjective misinterpretation of the ratings by the knowledge engineer. The results of the clustering algorithm provide the knowledge engineer with the information to begin his interviews with the expert. This entire process, termed the Automated KA Tool for Identification of Generic Tasks will decrease the KA process time and reduce bias in the tools chosen to implement the expert system.

3.2.1 Generation of Domain Concepts

Chapter listing asks the expert to imagine he / she has been asked to write a book. The expert is prompted to list the chapter titles and subtitles for such a book. This technique was automated with a C program. The program presented the book concept to the expert and the expert was prompted to enter each chapter title and associated subchapter titles. This process continued and was flexible enough to allow the expert to make revisions at any level. Satisfaction of the expert defines the termination of this process. The titles must be limited to one string of no more than 80 characters because testing proved 80 characters to be an acceptable limit. The titles are stored as a linked list during the entry phase, and later moved into a file. It is not necessary for all titles to be unique (no redundancy), as they will cluster to the same task and not distort the result.

3.2.2 Rating the Similarity

All titles as verified by the expert will be presented one at a time against the following generic task concepts. The generic task concepts are arranged into three groups that address the knowledge structure, strategy, and goal of a generic task. Within each group, each generic task concept addresses one of the 6 generic tasks. The expert rates the similarity of each title to each of these 18 concepts. The generic task concepts are:

Knowledge Structure Concepts:

1) This object can be labeled as an element of some hierarchy of elements.
2) This object can be labeled as a component in some subsystem of objects.
3) Characteristics of this object are shared by other objects.
4) This object can be labeled as an element of some overall plan.
5) This object provides partial evidence for a theory.
6) This object provides a partial explanation of a truth.

Strategy Concepts:

7) If this object is established or rejected, other objects can be also.
8) This object can change its state.
9) This object can inherit characteristics.
10) This object can be chosen as part of a plan.
11) This object can be matched.
12) This object can provide justification.

Goal Concepts:

13) If this object is true, the problem can be classified further.
14) If this object is true, the problem can be traced further.
15) If this object is true, the problem can be described further.
16) If this object is true, the problem can be designed further.
17) If this object is true, the problem can be evaluated further
18) If this object is true, the problem can be understood further

The similarity ratings range from 0 2 where:

No similarity 0
Little similarity 1
Very similar 2

For each title, the rating procedure yields 3 observations. Each observation is a row of 6 ratings; each rating represents the similarity of this title to a generic task. The 3 observations correspond to a generic task's relationships, strategy, and goal. These observations are stored in a file, and the expert can not change any of the ratings. The assumption is that his first instincts are best; just as in standardized testing, your first answer is usually the right one. Also during testing, the experts showed no desire to review their answers. If the knowledge engineer wants the expert to do the process twice, the expert can answer yes to the desire to use an already defined set of concepts. This saves the expert from having to define the concepts twice. Instructions pertaining to this rating scheme will present the expert with examples of the rating process. It is important that the expert be comfortable with the ratings prior to starting the concept / generic task ratings.

3.2.3 Rating the Similarity

The observations are the inputs into the FASTCLUS procedure, which is available by the SAS Institute. The FASTCLUS procedure separates the observations into groups such that each observation is a member of only one group. An observation is assigned to a group on the basis of the Euclidean distance between its quantitative variables and mean of the quantitative variables of all observations within a group. This procedure needs a minimum of 100 observations or the results may be affected by the order of the observations in the input set.

Inputs will look like:

Task #1 Rating | Task #2 Rating | Task #3 Rating | Task #4 Rating | Task #5 Rating | Task #6 Rating | Observation #

In the FASTCLUS procedure some observations are chosen and called cluster seeds. Preliminary clusters are formed by assigning each observation to the cluster with the closest seed match. For each observation assigned, the cluster seed is recalculated by computing the current mean of the cluster. The number of initial cluster seeds can be specified by setting the MAXCLUSTERS option.

In this study MAXCLUSTERS will be set to the number of generic tasks.

FASTCLUS prints the initial cluster seeds and their changes with each iteration. It also prints a cluster number, the number of observations in the cluster, the root mean square distance between observations in the cluster, the maximum distance from the seed to any observation. The output also includes the number of the cluster with the mean closest to the mean of the current cluster, and the distance between the "means" of the current cluster and its next closest cluster.

The statistics on each task are also printed.

The initial seeds will be:

	Task #					
Cluster #	1	2	3	4	5	6
1	2	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	2	0	0	0
4	0	0	0	2	0	0
5	0	0	0	0	2	0
6	0	0	0	0	0	2

These seeds were chosen so that each cluster had only one strong task
These seeds will be recalculated with each observation, and will yield a chart similar to this:

Cluster Means Task Number

Cluster #	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
1	1.9	15	.3	4	1.5	.7
2	.6	2	1.3	.7	.6	.4
3	.7	.6	1.7	.5	1.3	.2
4	.8	.7	1.6	1.6	.9	.4
5	1.3	1.8	1.2	.7	1	.8
6	.3	.7	.7	.9	.6	1.3

This chart will represent the strength of each task across all clusters. From these results, the highest means will yield the tasks upon which this domain is built.

3.2.4 Implementation Extension

The tool was refined during implementation and testing to create a friendly, and concise user interface and yield accurate results. Several design decisions were made during this time will be presented. These include the user interface and SAS procedure implementation details.

3.2.4.1 Separation of Titles and Ratings

The program has the ability to separate the entering of the titles from the entering of the ratings. This was implemented so the ratings would not be biased from weariness of the expert from a long computer session. The program stores the titles entered in the first session in a filename entered by the expert. During the second computer session, the expert inputs the filename and continues with

the ratings. The choice of one or two computer sessions is left to the expert.

3.2.4.2 Edit Menu

The editing of entered titles had to provide for: listing of titles, adding subchapter titles, deleting titles, and changing a title. It was also desirable to pattern the screen arrangement after familiar search screens such as those in a library. The editing procedure in this implementation lists the chapter titles, five at a time. The number of titles per screen is a # define that can be easily changed. The expert has the choice of moving to the previous list of titles, or to next screen of titles, choosing a specific chapter title or ending the editing session. The implementation handles the first and last screen of titles so that the appropriate choices of next screen or previous screen are presented. After the chapter title is chosen, the choices are: edit that title, edit one of that title's subchapter titles, or add a subchapter title. If the chapter title is chosen, the choices are: delete it or change it. If subchapters are to be edited, the titles are displayed like the chapter titles. If a subchapter is to be added, the appropriate prompt appears. After the editing of that title is complete, the expert can choose to edit another title. The editing menu can be selected by the expert after the completion of entering a chapter title and all of its subchapter titles.

3.2.4.3 Entering Titles

The appropriate prompt of "Enter a Chapter Title" is followed by "Enter 'none' when finished" This allows the expert to end without having to respond to a "Do you wish to end?" prompt after each title.

The number of characters per title, including spaces, is currently limited to 80. The limit is set by a # define that can easily be changed. Eighty was chosen because it is one line of characters on the screen, and the average length of a title during testing never exceeded 80.

Following each chapter title is a prompt to enter a subchapter title to further detail the chapter title. The expert continues to enter subchapter titles until 'none' is entered.

The titles are stored in a file named by the expert. The program also adds the extension ' .dat' to the filename because VAXA demands a file extension.

3.2.4.4 Entering Ratings

The introductory screen for ratings has to explain the ratings and how the ratings are chosen. This was difficult since the titles actually represent the thought process of the expert. This thought process is compared to the thought process represented by each of the stored concepts. The stored concepts are 18 statements that describe the three characteristics of each generic task and are detailed in section 3.2.2. The introduction to the ratings is implemented as follows:

The next part of the Automated KA Tool asks you to rate
your titles against other concepts.
Each title will be displayed on the screen next to a concept.
If they are NOT similar, type 0
If they are somewhat similar, type 1
If they are VERY similar, type 2
The concept will not be directly related to your title but
will be a description of a thought process. If your title is
involved in the thought process described by the concept then
the concept and your title are similar.
PRESS RETURN WHEN READY

For example, my book is How To Drive
TITLE:Driving safely in snow
CONCEPT: This object can be labeled as an element of some hierarchy

How I would rate the title to the concept:
When I entered driving in snow, my thought process was that
driving in snow is part of driving. It is similar to, yet different

from driving in rain and driving on dry roads which
are other chapters in my book. Thus, driving in snow can be
considered part of the driving hierarchy - and I would type
2 because the title and concept are similar.
WHEN READY TO CONTINUE PRESS RETURN

At each rating the title and the concept are presented as follows

TITLE
CONCEPT
Enter 0 -if similar
Enter 1 - if somewhat similar
Enter 2 if very similar
Enter H - for help

Help presents 3 lines that further define the concept. The expert is then asked to enter the rating. The ratings are stored in "rates.dat".

3.2.4.5 Implementation of User Input Protection

All instances of user responses to queries are implemented to only accept inputs within the range of acceptable responses. The responses are not case sensitive.

3.2.4.6 The Stored Generic Task Concepts

The 18 statements that are rated in similarity to each title are contained in task.dat. Each statement is one line (80 characters long) and reflects one aspect of a generic task. That aspect can be the knowledge representation, the control strategy or the goal of that generic task. Each statement is followed by three lines (80 characters long) that provide a help facility. The 3 lines further explain the statement to the expert, if the expert enters 'H' for help. The task.dat file is accessed by the thesis.exe program during the rating of the concepts.

3.2.4.7 The Stored Seeds

The clustering algorithm has an optional initial seed definition. The initial seeds were chosen to be:

```
2 0 0 0 0 0
0 2 0 0 0 0
0 0 2 0 0 0
0 0 0 2 0 0
0 0 0 0 2 0
0 0 0 0 0 2
```

This was to establish the initial clusters to have only one strong task per cluster.

3.2.4.8 The Statistical Procedure

The SAS FASTCLUS procedure was chosen to cluster the ratings to a task. It required a program using SAS statements that is implemented in thesis.sas. Thesis.SAS defines the rates.dat file and seeds.dat as inputs. Rates.dat is the file of ratings; seeds.dat contain the ideal ratings, and both are listed in Appendix C.

Thesis.sas is implemented as follows:

```
1) options nodate linesize = 80;
   /*
   ** Are the SAS options. It was decided that no DATE would be printed
   ** and that the columns per line is 80.
   */
2) data tasks;
   /*
   ** Signals the beginning of the Data Step and gives a name 'Tasks' to
   ** the SAS data set.
   */
3) filename rates 'rates.dat';
   /*
   ** assigns the rates.dat filename to rates
   */
```

```
4) infile rates;
        /*
        ** opens rates.dat
        */
5) input task1, task2, task3, task4, task5, task6, con_no@;
        /*
        ** describes each line of rates.dat
        */
6) run;
        /*
        ** executes the DATA step to open rates.dat
        */
7) data seeds;
        /*
        **Signals the beginning of the Data Step and gives a name 'Seeds'
        ** to the SAS data set.
        */
8)  filename inits 'seeds.dat'
        /*
        ** assigns the seeds.dat filename to inits
        */
9)  infile inits;
        /*
        ** opens seeds.dat
        */
10) input task1 task2 task3 task4 task5 task6
        /*
        ** describes each line of seeds.dat
        */
```

```
11) run;
      /*
      ** executes the DATA step to open seeds.dat
      */
12) proc print
      /*
      ** signals the beginning of a printing procedure
      */
13) title 'Generic Task Identification'
      /*
      ** the title to be printed
      */
14) run;
      /*
      ** executes the print procedure to print the title
      */
15) proc fastclus data = tasks seed = seeds maxc = 6 maxiter = 10 out = cluslist;
      /*
      ** signals the beginning of a FASTCLUS procedure and sets the
      ** following:
      **  DATA = tasks - tasks is the input file to be read
      **  SEED = seeds - seeds is the input file for seeds
      **  MAXC = 6 - maximum clusters = 6 for the 6 tasks
      **  MAXITER = 10 - maximum iterations for recomputing cluster seeds
      **  OUT = clustlist  names an output data set.
      */
16) var task1 task2 task3 task4 task5 task6;
      /*
      ** lists the variables that are considered in the analysis
      */
17) id con - no;
      /*
      ** used to identify each observation on the printout
      */
```

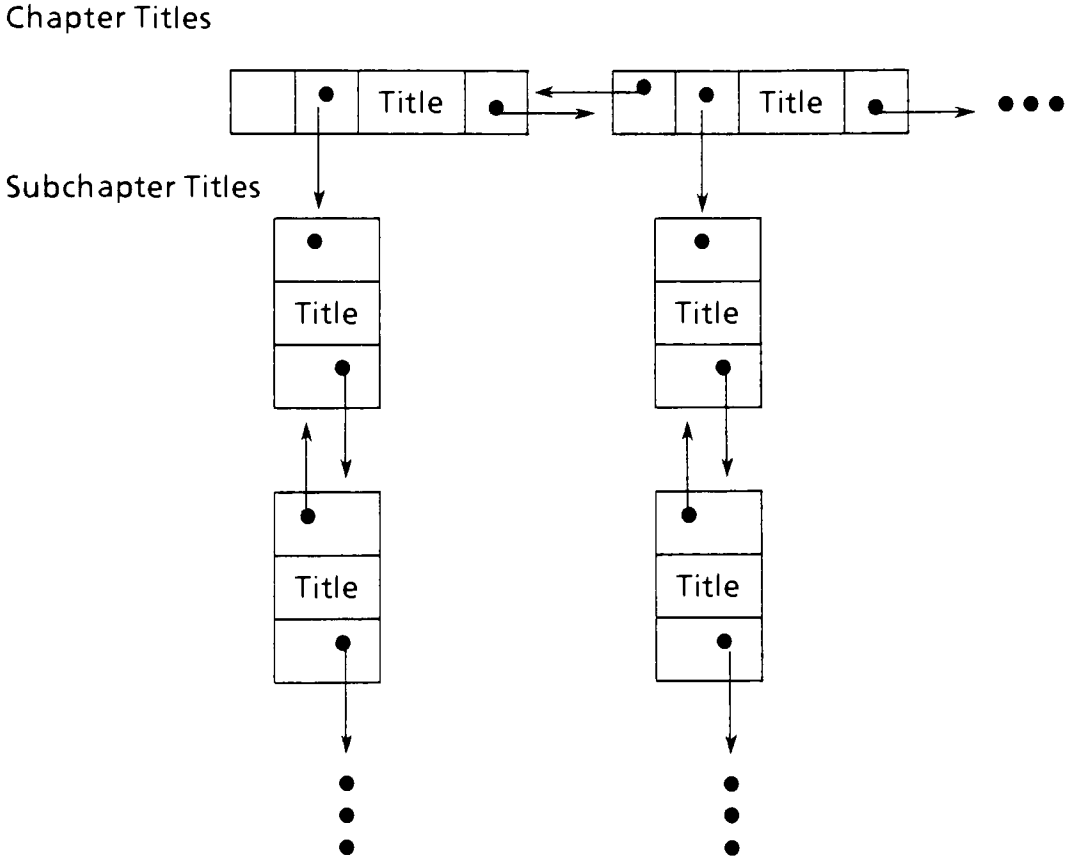
```
18) run;
      /*
      ** executes the FASTCLUS procedure
      */
19) proc candisc out = can;
      /*
      ** signals the beginning of the CANDISC procedure, OUT = CAN
      ** names an output SAS data set
      */
20) class cluster;
      /*
      ** defines the class for analysis
      */
21) var task1 task2 task3 task4 task5 task6;
      /*
      ** lists the variables to be analyzed
      */
22) title2 'canonical discriminant analysis of task clusters';
      /*
      ** gives a title to the resulting printout
      */
23) run;
      /*
      ** executes the CANDISC procedure
      */
24) proc plot;
      /*
      ** signals the beginning of the Plot procedure
      */
25) plot can1 & can2 = cluster
      /*
      ** plots the results from the CANDISC procedure
      */
```

```
26) title2 'Plot of Canonical Variables identified by clusters'
      /*
      **defines a title for the output
      */
27) run;
      /*
      ** executes the plot statement
      */
28) endsas;
      /*
      ** ends the SAS program
      */
```

It was decided to include the CANDISC procedure to summarize between cluster variations. This combines observations and yields the strongest clusters.

3.2.4.9 Implementation of Data Structures During Execution

The program consists of two functions accessing data structures: entering of the titles, and entering of the ratings. The titles are stored in a doubly linked list to allow easy deletion and insertion during editing. Each chapter title node contains a pointer to a linked list of subchapter nodes.
Each chapter title node (chap) is defined as follows



The ratings are written to a file immediately because only the initial rating was desired. The ratings are stored 6 per row, followed by a unique observation number

Chapter 4
Results

4.1 Introduction

The Automated Knowledge Acquisition Tool for Identification of Generic Tasks was tested to insure correct functionality and a friendly, correct user interface by three instances.

The correct functionality was based on the implementor's perception as to the usage of this program. It was assumed the program would be operated within the realm of friendly usage. This assumption was based on the intended user being a possibly computer ignorant yet not malicious expert. The implementation, therefore, ignores all responses from the expert not within a specified range of responses and all response comparisons are case insensitive. This allows for the expert's inadvertent inputs, but it does not provide total protection against malicious misuse.

The user interface was constrained to be concise, unintimidating and true to the characteristics of generic tasks detailed in Section 2.3. During the implementation the wording in tasks.dat underwent numerous changes to meet these constraints.

Another issue during implementation was the environment mandated by the usage of FASTCLUS. SAS programs can be executed only on VAXA cluster of the RIT undergraduate computer. This operating system was time consuming to master and thus constrained the implementation. The environment of the ROSS lab and modem facilities also constrained the implementation. The lab technicians were not knowledgeable, the facility was overcrowded, and the modem facilities would frequently be down. The modem facilities would also cause problems when executing the SAS procedures, yielding results of undecipherable control characters. This environment also limited the experimentation with disabling keys and screen controls for the user interface for protection against misuse.

During the analysis of the examples, SAS was run using seeds and without using seeds. The results were the same, but the CANSISC procedure plot, although yielding the same results did not have the distinct visual clustering in the unseeded results. Therefore the implementor chose to display the seeded results in the following results.

4.2 Examples of the Usage of the KA Tool

In an attempt to demonstrate the capabilities of the Automated Knowledge Acquisition Tool For Identification of Generic Tasks, three examples were chosen. The examples were an attempt to identify the generic tasks of a diagnostic domain, a plan domain and an unknown domain.

4.2.1 The Diagnostic Domain

An expert was chosen in this domain and asked to use the program. The expert was given the expert's user manual and the program was explained in accordance with the knowledge engineer's user manual. The expert was an electrical engineer responsible for debugging circuit boards.

4.2.1.1 The Titles and Ratings

The following was generated by the expert. Each title is followed by the ratings that title generated. The ratings are grouped into observations as described in section 3.2.2.

Title	Observation 1	Observation 2	Observation 3
Backplane	222201	001200	121102
Pin Assignments	222200	000100	111001
Master Communication Channel	222000	001000	111101
Set Initialization Mode	221200	020200	200111
Read Initialization Mode	222200	020200	111001
Hardware Diagnostics	221200	000201	222012
Slave Communication Channel	222200	021101	111001
Communication Clock	222200	000000	100011
Read / Write Memory	222200	001202	001000
Power Up Diagnostics	222200	000101	121000
Read Only memory	222200	000211	000001
Power Up Checksum Test	222210	000201	010000
Input Ports	222220	000202	000000
Input Loop Back Test	222211	000202	211010
TTL Inputs	222200	000202	111000
Sensor Inputs	222200	000212	221101
Output Ports	222201	220202	202100
Output Loop Back Test	222210	201202	120000
AC Driver Circuit	222200	020202	110001

Darlington Driver	222200	100101	111001
LED Driver	222200	110101	010011
Real Time Clock	222200	010201	000000
Power Up Diagnostics	222201	000202	011010
Timer Bypass Circuit	222200	020102	000000
Voltage Diagnostics	222202	220201	220010
Logic Voltage Test	222211	120212	202000
Driver Voltage Test	222222	200202	220000
Diagnostics Voltage Test	222200	220202	222010
Power Normal / Reset	222222	220222	222010
Watchdog Timer Circuit	222220	120212	210001
Keeping Watchdog Alive	222200	000201	000100
Recovery from Timeout	222220	100102	000000
Hardware indicator Circuit	222222	220222	222022
Softload	222200	000202	000000
Boot Prom	222200	101202	000100
Softload Directory	222200	200202	001000
NVM	222200	100202	000001

5.2.1.2 The SAS Results.

The Cluster Means is as follows

Cluster	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
1	2	2	1.94444	2	44444	.36111
2	1	2	15385	1.76923	46154	1.53846
3	0	.14286	.28571	.21429	.07143	0
4	47368	.05263	.21053	1.94737	10526	1.63158
5	1.6	1.4	1.150	15	.35	.55
6	.66667	.33333	0	4444	.33333	1

Horizontal Analysis by Cluster:

A strong task has a cluster means close to two. A weak task has a cluster means close to zero.

Cluster 1 has tasks 1, 2, 4, 3 are strong, and 5, 6 are weak.

Cluster 2 has tasks 2, 4, 6, 1 and 3, 5 are weak.

Cluster 3 has no tasks strong.

Cluster 4 has tasks 4, 6 and 1, 2, 3, 5 are weak.

Cluster 5 has tasks 1, 2, 3, and 4, 5, 6 are weak.

Cluster 6 has no tasks strong.

Vertical Analysis by Task:

Task 1 is in Cluster 1

Task 2 is in Cluster 1, 2

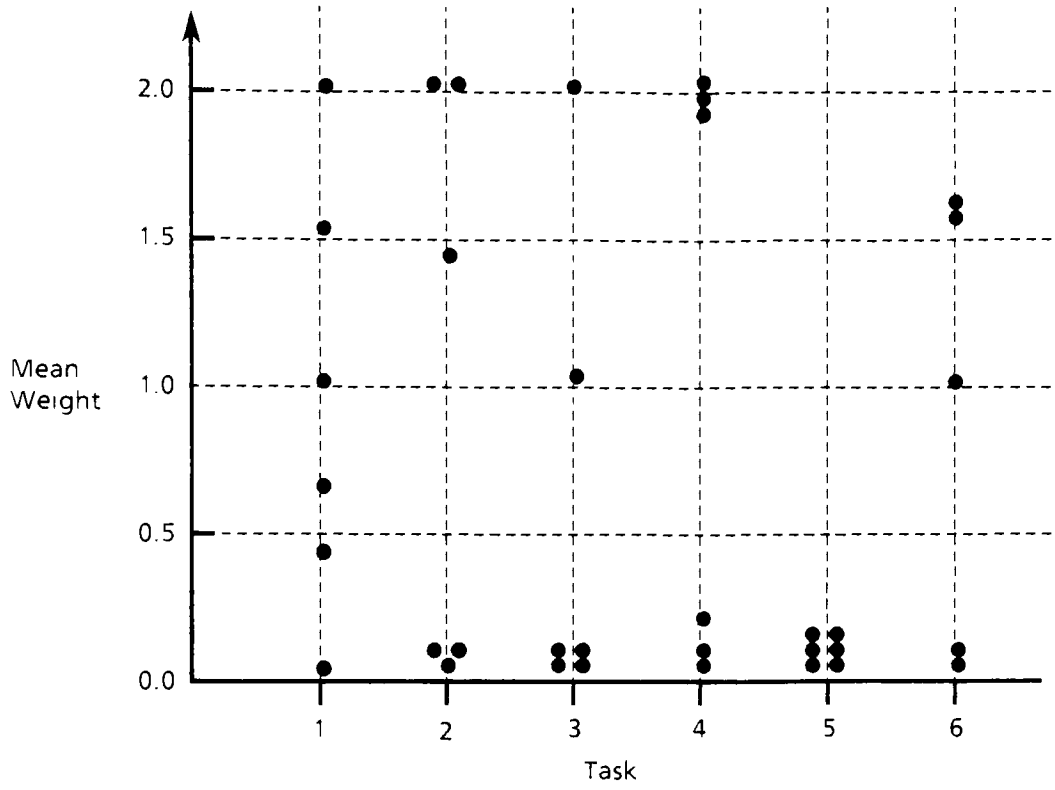
Task 3 is in Cluster 1

Task 4 is in Cluster 1

Task 5 is in no Cluster

Task 6 is in Cluster 4

A graph of the cluster means to visualize the tasks yields:



The dot is the cluster means and the graph depicts the strength of the task across all clusters.

Task 1 is fair in the majority of the clusters.

Task 2 is strong in half of the clusters.

Task 3 is fair in two of the clusters.

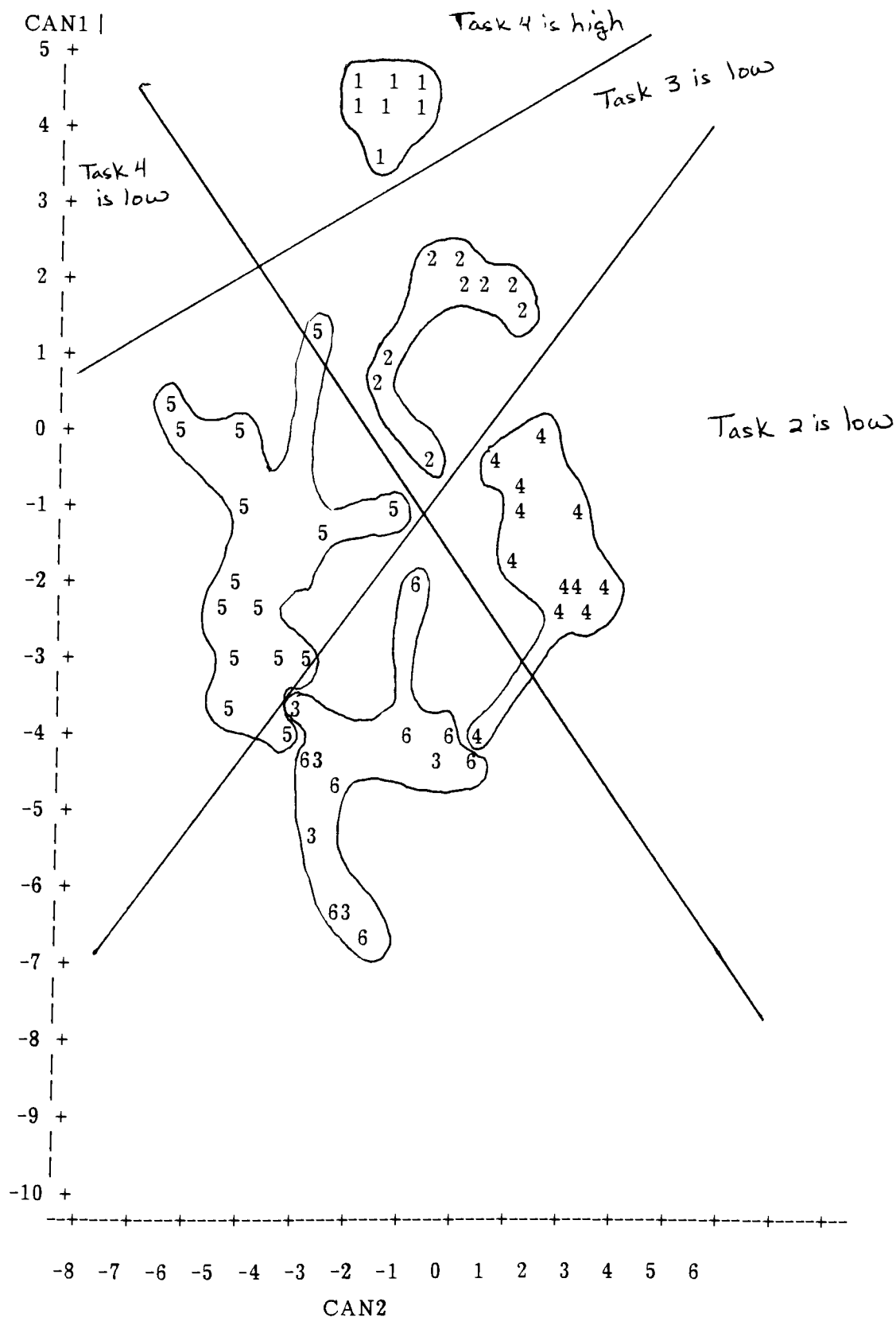
Task 4 is the strongest in half of the clusters.

Task 5 is nonexistent in all the clusters.

Task 6 is fair in two of the clusters.

This implies this domain consists of Task 4, Task 2, Task 1, Task 6 and Task 3 somewhat.

Looking at the plot generated by the CANDISC procedure:



The plot divides into 5 sections. As you move down from the top section, Task 3 cluster means decreases then the Task 2 cluster means decreases. This implies that Task 2 is stronger than Task 3. Going from right to left Task 4 cluster means decreases. The absence of Task 6, 1, and 5 in the tasks that differentiate sectors implies that a weak Task 1 and 6 appear in all and Task 5 appears in none of the clusters. Clusters 3 and 6 merge into 1 cluster because their only difference is Task 6 which is not a strong enough task to differentiate on.

Therefore Task 4, 2 are the main tasks and aspects of 1 and 6 are present in the domain. Task 3 is weak. Task 5 is not part of this domain.

Translating the task number yields this domain consisting of:

Plan selection and Refinement
State Abstraction
Classification
Explanation
Some Knowledge Directed Information Passing

and no Hypothesis Matching.

4.2.1.3 The Observations.

The expert chose to use the program over two sessions. During the first session, the entering of the titles went smoothly. The expert could edit any typographical errors and understood the prompts.

During the second session, the ratings were entered. The expert appeared to have difficulty when the response should have been zero for not similar. This implied the expert would rather struggle to find some similarity than enter a zero.

During the presentation of the results, the expert was astonished at the accuracy of the results.

4.2.1.4 The Conclusions.

The analysis of the SAS results was done prior to a scanning of the titles or the ratings by the implementor. The implementor had no knowledge of the expertise beyond it being a diagnostic problem. The implementor presented the results to the expert with trepidation for the presence of plan selection, classification, and explanation, and knowledge directed information seemed unusual for a diagnostic problem. The implementor expected a strong state abstraction domain.

The expert agreed with the results of the analysis and clarified the usage of the unusual tasks with examples. The state abstraction is the simulation of circuit boards and power sources that comprise a subsystem of a Xerox product. The expert formulates a plan of action and refines that plan as the diagnosis proceeds. This is the plan selection & refinement task.

The test parameters are layered and all parameters must succeed to proceed to the next layer. The expert classifies test parameters into these layers, and this is the classification task.

The explanation becomes important when writing the manual for the technical representative describing the solution to this problem. This task only becomes important at the end of the expert's process and is therefore not a strong task in the domain. It is important to note that an existence of a task in the analysis implies the task is in the expert's domain.

Information can be surmised from a description of the watchdog timer circuit and how it is kept alive to explain the indicator and recovery from a fault. This is an example of a knowledge directed information passing task used by the expert.

The expert could not give an example of hypothesis matching. There is no 'goodness of fit' decision making in his thought process. This agreed with the analysis.

In conclusion, the analysis was an accurate representation of the expert's thought process.

4.2.2 The Planning Domain

An expert was chosen in this domain and asked to use the program. The expert was given the expert's user manual and the program was explained in accordance with the knowledge engineer's user manual. The expert was a district manager of a financial investment company.

4.2.2.1 The Titles and Ratings

The following was generated by the expert. Each title is followed by the ratings that title generated. The ratings are grouped into observations as described in section 3.2.2.

Title	Observation 1	Observation 2	Observation 3
Economic Environment	120221	022222	101000
The Federal Reserve System	012200	022222	000011
Economic Indicators	222121	222222	222022
The Business Cycle	222122	122222	002222
The Time Value of Money	202222	210222	020222
Personal Financial Planning	122220	122222	022222
The Process	000022	211122	222212
Financial Statements	222201	000022	000100
regulation of Planners	122000	201022	222222
Risk Management	222222	222222	222222
Universal Life	021112	220212	220222
Variable Life	022210	222222	000222
Health, Property, and Liability Insurance	022222	222222	211112
Medical Expense Insurance	122211	212211	100202
Homeowners Insurance	022210	222222	120202
Nursing Home Insurance	022200	222222	200202
Principles of Life Insurance	222201	122222	201222
Investment Planning	202221	222222	222222
Risk Tolerance	022220	012202	220022
Interpreting Statistical Data	022020	000222	002202
Fixed Income Investment	122200	100222	000000
Equity Investments	012220	222222	022222
Tax Planning	122200	112222	111222
Personal Income Tax Planning	222222	022221	202220
Forms of Business Ownership	022200	002200	000000
Business Taxation	202211	222111	122222
Retirement Planning	022200	222212	222222
Retirement Planning Personal	122202	222222	222200
Qualified Pension and Profit Sharing Plans	211101	122222	211222

Individual Retirement Accounts	222200	022212	211202
Tax Sheltered Accounts	222222	202102	000000
Estate Planning	202220	222212	221222
Estate Planning Principles	222222	222211	221111
Life Insurance Ownership and Beneficiaries	102222	020222	002211
United Transfer Tax System	020101	201112	022200
Gross Estate Calculations	202222	220111	000220
Wills, Intestacy	202200	122212	112222
Trusts	112220	200200	022000
Ownership of Property	202200	221222	000000
Charitable Transfers	112201	100212	000220
Property Transfers	022200	102212	200000

4.2.2.2 The SAS Results.

The Cluster Summary is as follows:

The Cluster Means is as follows

Cluster	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
1	2.00000	1.91429	1.60000	1.68571	1.74286	1.85714
2	0.59259	1.74074	1.92593	1.74074	0.48148	0.33333
3	1.68421	0.15789	1.73684	1.73684	1.10526	1.31579
4	0.90909	0.27273	0.00000	2.00000	1.18182	1.45455
5	0.30000	0.00000	0.10000	0.10000	0.50000	0.50000
6	0.47619	1.76190	1.61905	1.95238	1.85714	1.90476

Horizontal Analysis by Cluster:

A strong task has a cluster means close to two. A weak task has a cluster means close to zero.

Cluster 1 has tasks 1, 2, 6, 5, 4, 3 all tasks were represented.

Cluster 2 has tasks 3, 2, 4.

Cluster 3 has tasks 3, 4, 1, 6, 5.

Cluster 4 has tasks 4, 6, 5.

Cluster 5 has no tasks strong.

Cluster 6 has tasks 4, 6, 5, 2, 3.

Vertical Analysis by Task:

Task 1 is in Cluster 1

Task 2 is in Cluster 1

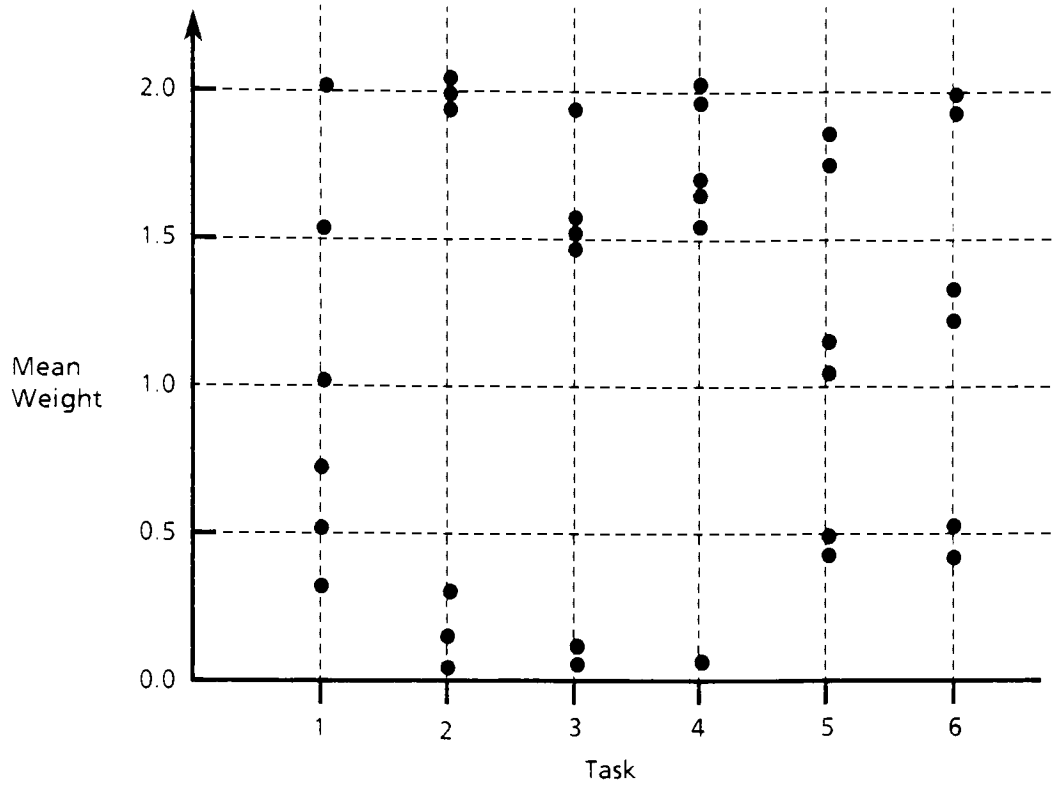
Task 3 is in Cluster 2

Task 4 is in Cluster 4

Task 5 is in Cluster 6

Task 6 is in Cluster 6

A graph of the cluster means to visualize the tasks yields:



The dot is the cluster means and the graph depicts the strength of the task across all clusters.

Task 4 is the strongest task

Task 3 is the next strongest task

Task 2 is the next strongest task

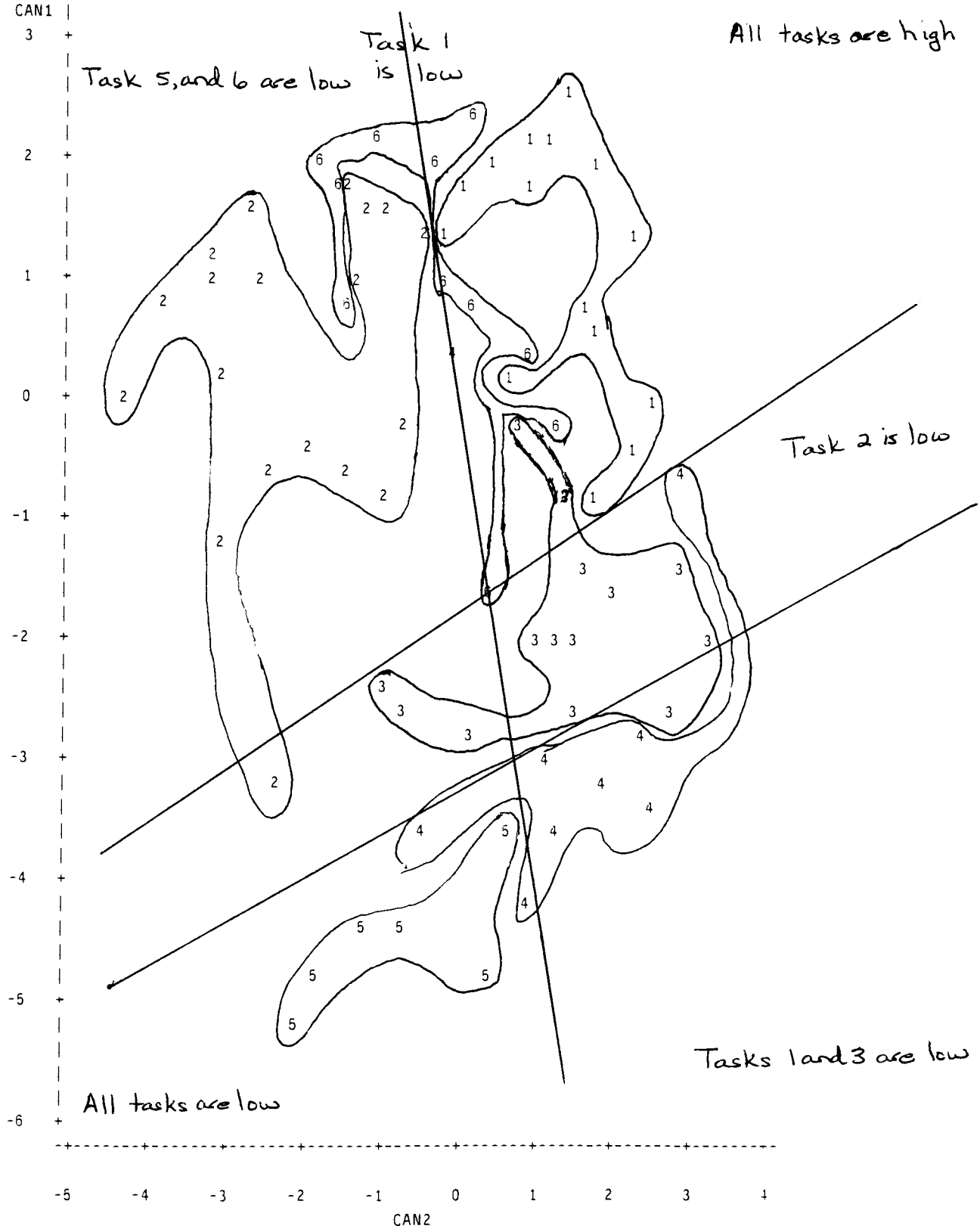
Task 6 is strong

Task 5 is strong

Task 1 occurs but only in a small aspect of the domain

Looking at the plot generated by the CANDISC procedure:

Plot of Canonical variables identified by cluster
PLOT OF CAN1*CAN2 SYMBOL IS VALUE OF CLUSTER



NOTE: 49 OBS HIDDEN

This plot divides into 6 cluster means sectors. Moving from top to bottom decreases Task 2 cluster means and then Tasks 1 and 3. This implies that Task 3 and Task 1 is stronger than Task 2. But Task 1 is also represented in the left / right sections of the graph. Moving from right to left decreases Task 1 cluster means and then Tasks 5 and 6 cluster means. The diagonal from upper right to bottom left decreases ALL tasks. This implies the following task strengths.
Task 4 is in the majority of the clusters
Task 5,6 are in the majority of the clusters
Task 3 is in the majority of the clusters
Task 2 is present
Task 1 is present only slightly. This is from the separate cluster eliminating Task 1 in the upper sectors.

Translating the generic task numbers yields a domain consisting of

plan - selection and refinement
Knowledge - directed information passing
State Abstraction
Assembly of Explanatory Hypothesis
Hypothesis Matching
and some classification

4.2.2.3 The Observations.

The expert chose to use the program over two sessions. The titles were entered during the first session. The expert had no trouble entering and editing the titles.

During the rating session, the expert seemed comfortable with the ratings and used the help option infrequently.

As the results were presented to the expert, he gave examples to verify that the results were correct. He was not surprised in the accuracy of the results. He was familiar with psychological testing techniques through his employees being tested to determine their style.

4.2.2.4 The Conclusions.

The implementor had chosen this expert expecting a planning domain. Assuming the expert creates a financial plan for a client, it should have been a predominantly planning task. The implementor was surprised by the results and not confident presenting them to the expert.

The expert explained that the classification of a client was the first action he takes. The expert changes his style to match the client's classification and gave examples. This explains the weak presence of the classification task.

The State Abstraction was also accurate because all aspects of a financial service can change state; ie. stock market, interest rates, tax issues. The expert is always monitoring for changes and adjusting the financial investments to create the best possible combination for that client.

This plan is then refined by the client's inputs, not the expert's. This explains the lack of total domination of the planning task in the analysis, and the presence of the 'best fit' hypothesis matching.

The information used by the expert is knowledge directed in that information such as higher interest rates can be retrieved from knowledge of a depressed stock market. Since the expert must explain his recommendations to his client, the Assembly of Explanation hypothesis task is required.

In conclusion, the results were an accurate reflection of the expert's thought process.

This example displays the importance of using an automated tool that is not influenced toward a particular task. The implementor's bias toward plan selection and refinement did not influence the results of this example.

4.2.3 The Unknown Domain

An expert was chosen and asked to use the program. The expert was given the expert's user manual. The expert was a teacher and considered an expert in parent / teacher communications. The implementor had no prior knowledge of the tasks to be generated by this example.

4.2.3.1 The Titles and Ratings

The following was generated by the expert. Each title is followed by the ratings that title generated. The ratings are grouped into observations as described in section 3.2.2.

Title	Observation 1	Observation 2	Observation 3
before conferences	2 2 1 2 2 2	2 1 2 2 2 2	2 2 2 2 2 2
set up schedule 15 minutes each and 5 minutes between	2 1 2 2 2 2	2 0 2 2 2 2	2 2 2 2 2 2
purchase index cards	2 2 0 2 1 0	0 0 2 2 1 1	2 2 2 2 2 2
gather books to be displayed the night of conferences	2 1 1 1 1 1	1 2 1 1 1 1	1 1 1 1 0 1
organize reports and index cards in order of conferences	2 2 2 2 2 2	2 0 2 2 2 2	2 2 2 2 2 2
xerox handouts for parents	2 2 1 2 1 1	2 2 2 2 2 2	2 2 2 2 2 2
staple handouts for parents	2 2 0 2 2 2	2 2 2 2 0 0	2 0 2 1 1 1
put last name of child on handouts	1 2 0 1 2 2	2 0 0 2 2 2	2 2 2 2 2 2
put name of child on index card	2 2 1 2 2 1	2 0 0 2 2 0	2 2 2 2 2 2
put reading level on index card	2 2 2 2 2 2	2 0 0 2 2 2	2 2 2 2 2 2
list child's strong points	2 2 2 2 2 2	2 2 2 2 2 1	2 1 1 1 2 1
list child's weaknesses	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
get conference form for each child	2 2 2 2 1 2	2 2 2 2 2 2	2 2 2 2 2 2
organize conference forms in order of conferences	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
night of conference	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
place conference schedule on door and on table near you	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
place handouts on table	2 2 2 2 2 2	2 2 2 2 2 1	1 1 1 1 1 1
place report cards and index cards on table	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2

place pens on table	220200	000200	111111
set up display books on small table outside the door	222211	122221	211111
greet parents with smile and walk to the table	220200	011220	000000
look at report card first	222222	222222	222222
ask parents if they have any questions about the report	221211	200211	111111
if concerns, discuss these first and make notes on index cards	222222	222222	222222
go to index cards	222222	222222	222222
explain reading level have series book to look at	222222	222222	222222
discuss child's strong points	222222	222222	222222
discuss weaknesses - suggest ways 'WE' can help child improve	222222	222222	222222
conclude conference quickly by summarizing any concerns	222222	222222	222222
hand out parenting handouts - these were helpful to me as parent/teacher	222222	222222	111111
thank parents for coming and walk them to the door	221211	111221	000000
if time - write down any important notes for conference form	222222	222222	222222
write up conference forms - date, who attended, what was discussed	222222	222222	222222
if parents become antagonistic	220222	200020	111111
do not argue - remain calm	222222	222222	222222
take deep breath - try to get them to open up about problem	222222	222222	222222
if they say something negative repeat back to keep them talking	222222	222222	222222

make phone call to parents who didn't show up	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
get information to parents who requested it	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
psychological testing, counselor	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2
discuss concerns with principal	2 2 2 2 2 2	2 2 2 2 2 2	2 2 2 2 2 2

4.2.3.2 The SAS Results.

The Cluster Means is as follows

Cluster	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
1	2.00000	1.97619	1.98810	2	1.98810	2
2	1.30	1.0	1.10000	1	0.900	1
3	0	0.5	1.5	2	1.5	0.5
4	1.3333	0.66667	0.00000	1.3333	0.66667	0.11111
5	1.77778	0.88889	0.66667	1.77778	2	1.77778
6	1.88889	2.0000	1.55556	2	1.3333	0.88889

Horizontal Analysis by Cluster:

A strong task has a cluster means close to two. A weak task has a cluster means close to zero.

Cluster 1 has tasks 1, 2, 3, 4, 5, 6, all strong

Cluster 2 has all tasks fairly weak.

Cluster 3 has tasks 4, 5, 3 are strong.

Cluster 4 has tasks 1, 4 fairly weak, 2, 5, 6, 3 are very weak.

Cluster 5 has tasks 1, 4 , 5, 6 strong and 2, 3 weak.

Cluster 6 has tasks 1, 2, 3, 4 strong, 5 weaker & 6 is weak.

Vertical Analysis by Task:

The cluster in which the task is the strongest.

Task 1 is in Cluster 1

Task 2 is in Cluster 6

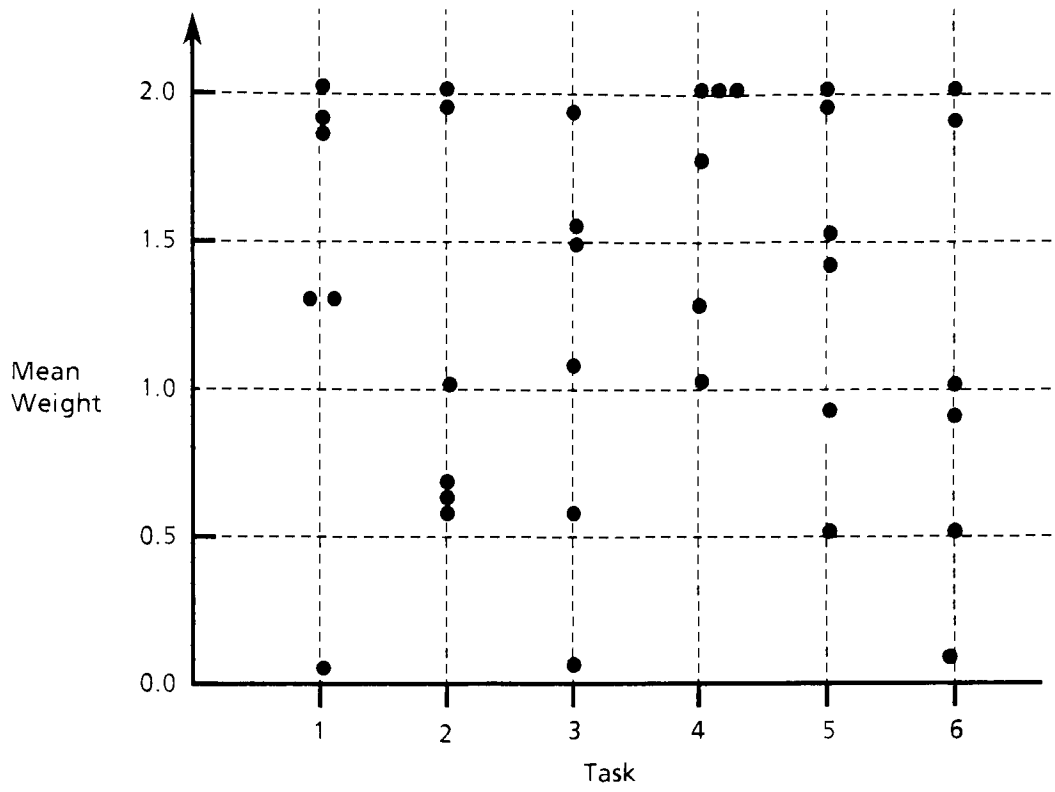
Task 3 is in Cluster 1

Task 4 is in Cluster 1, 3, 6

Task 5 is in Cluster 5

Task 6 is in Cluster 1

A graph of the cluster means to visualize the tasks yields:

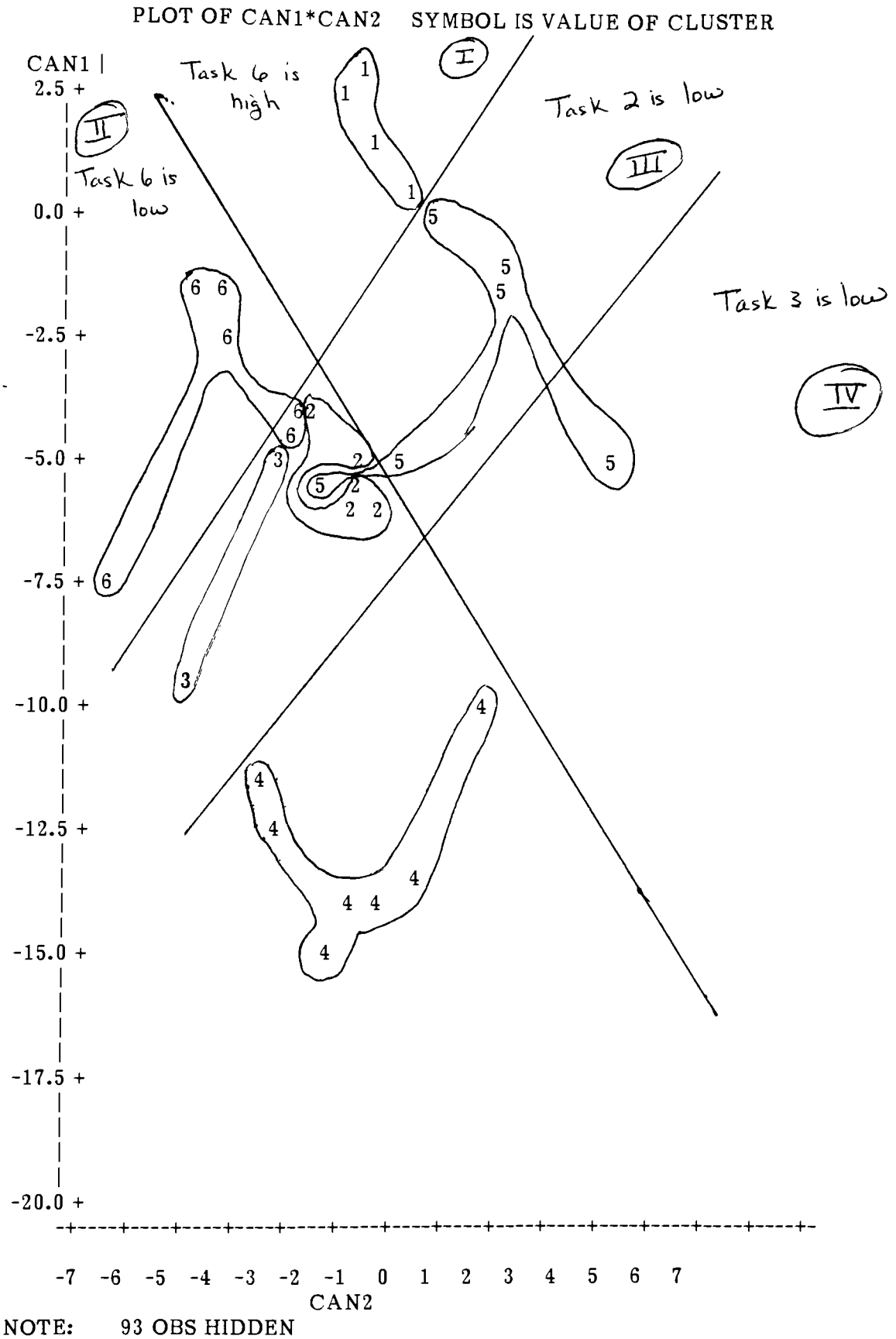


The dot is the cluster means and the graph depicts the strength of the task across all clusters.

Task 4 is strong in all clusters
Task 1 is strong in all but 1 cluster
Task 5 is strong in the majority
Task 3 is strong in the majority
Task 2 and 6 are weak

Looking at plot generated by the CANDISC procedure
CAN1 represents Task 6, 3, 2
CAN2 represents Task 6

Candisc Plot:



NOTE: 93 OBS HIDDEN

Moving from Sector I to Sector II, the Task 6 cluster means becomes weaker.
Moving from Sector I level to the Sector III level the Task 2 cluster means weakens.
Moving from Sector III level to Sector IV level the Task 3 cluster means weaken.
The outliers imply that some occurrences within a cluster differ on a task, but not enough to become a different cluster. The blending of clusters 2, 3, 5, 6 implies that the tasks they share are stronger than the tasks they differ on. They share Task 4 and differ on Task 1, Task 6 and somewhat differ on the rest.

Therefore from the plot:
Task 4 is strongest
Task 1 is strong
Task 5 is strong
Task 3 is strong
Task 6 is weak
Task 2 is weak

Converting the Task numbers to generic tasks yields the domain is a combination of

Plan Selection and refinement
Classification
Hypothesis Matching
Knowledge directed information

Explanation & State Abstraction are weak

4.2.3.3 The Observations.

The expert chose to use the program over two sessions. During both sessions, the expert required the presence of the knowledge engineer. This was due to the inexperienced of the expert at the computer. This expert had a shorter attention span and liked to talk about every 20 minutes during the sessions. The knowledge engineer tried to leave the expert alone as much as possible. One incident occurred that displayed the title / thought process connection. The expert was on the first title "Before Conferences" during the rating session. The expert stated to the knowledge engineer "When I'm formulating the truth about a child, it is my perception not necessarily the parent's". The truth concept triggered a complex thought process from a

simple title. It was interesting to observe the ability of a simple title in combination with the generated concepts triggering complex thought processes that can be transposed to a rating and clustered.

4.2.3.4 The Conclusions.

The analysis of the SAS results was done prior to a scanning of the titles or ratings by the implementor. The analysis was presented to the expert.

The plan selection and refinement task describes the preparation the expert does prior to a conference. Any notes from the parents, conversations with other teachers, and conversations with the principal are used as material to plan the conference techniques used by the expert. Plans are chosen from among those in the expert's repertoire. The refinement goes on during the conference.

Classification was a task used to classify a child's problem areas, ie: a bully, a shy child, a speech problem. These classifications would then imply certain solutions, ie: counseling, books, testing. Parents could also be classified as 'controlling', 'scared', etc. which imply different styles the expert would use to approach them.

Hypothesis matching refers to the expert and parents together, creating the solution or "best fit" to a child's problem.

Knowledge directed information was important to the expert because learning disabilities are usually inherited through the father. Thus when a parent is approached with the disability of a child, their own childhood experiences bias their reaction. The implementor used the phraseology "information that can be surmised from other information" not the word inheritance to describe this task, and the expert responded in this manner. The implementor had not expected this response for this task.

Explanation is not a strong task because the expert strives for a "we" solution, where the teacher and parents and student are a team. The expert does not state opinions and explain them. However, procedures such as testing may need to be explained.

The State Abstraction task is involved during the conference. The body language of the parents change and the teacher needs to be aware and compensate to keep the conference moving in a forward direction.

In conclusion, the tasks did describe the domain and the strength or weakness of the task, as stated in the analysis, was accurate.

Chapter 5
Conclusions

The current implementation of the Automated Knowledge Acquisition Tool for Identification of Generic Tasks provides a reasonable user interface for the expert, and credible results for the knowledge engineer Contemplating the results and observations of the examples, several areas of future enhancement arise.

The observation of the expert searching for some similarity instead of immediately entering a zero is one area. The belief that the rating of similarity as stated in section 4.2 provides an unbiased comparison is still valid, however, the following questions arise. Would a scale of yes, no, maybe or 1, 2, 3 instead of 0, 1, 2 reduce the experts reluctance? Would a direction such as "Please do not over analyze your rating. If there is no immediate similarity to you, enter zero" work. The analysis of the psychological issues in this area could prove interesting and yield an improved implementation. This issue of nuances of language influencing the actions of the expert further supports the claim that a "interview only" knowledge acquisition methodology would bias the expert's responses implementation.

Another area of experimentation would be the task.dat file. The wording of these sentences as implemented has been successful, but there is always the question "Could it be improved"?

Analysis of the ratings generated by the experts yielded the following:

One title generated 3 lines of ratings. The first line rated the structure, the second line the strategy, and the third line the goal. The natural assumption was that the three lines would be similar for a title. The observation of the examples did not imply this. The first reaction is to question the wording of the generic task concepts. Further analysis, however, presents the following conclusions. A title is actually the trigger to a thought process of the expert. That thought process can contain one aspect of the domain that is involved in many different relationships with other aspects. This explains 'a row of twos' in the structure row. However, this same aspect would only have 1 or 2 strategies applied to itself. This is observed in the second row being mainly zeros. The last row represents a goal. Similarly, this aspect would only be involved in one of a few of the expert's goals. Continued use of this tool and

observing the results could verify these conclusions and imply that the wording of the generic task concepts is accurate.

The success of multidimensional scaling documented in this paper yields another area an implementation extension. Using this technique would require a greater statistical background than this implementor, but could produce a stronger analysis than FASTCLUS.

A windows environment for the user interface would greatly enhance this implementation. The windows display is user friendly, popular and flexible. The porting of this thesis to other environments is possible. The 'C' language program is portable to all operating systems having a 'C' compiler. The SAS software packages needed are available for UNIX on a SUN or Hewitt Packard workstation for approximately $950 initially and a $432 yearly renewal fee. It is also available for the PC environment for $760 initially and $200 renewal fee. These prices are approximate and vary depending on the SAS package purchased and the university discounts.

Automation of the entire KA process can be achieved by automating two additional processes. Refinement of each title into the associated generic task's knowledge representation is the first process to automate. This process must allow addition and deletion of titles, flexible movement of the titles within a knowledge representation, and provide an adequate explanation of the knowledge representations to the expert. The final process for complete automation would refine the appropriate strategy for each knowledge representation and add the message passing scheme between strategies.

This study of a partial automation of the KA process eliminated knowledge engineer bias, reduced the time required to identify the generic tasks and exposed implicit knowledge. These successes imply that the automation of the entire KA process would reap even greater benefits. However, this study also exposed the need of some experts to have human contact. This occurred during the third test example where the expert was not computer literate and whose area of expertise involved human interaction. The results from this expert were successful, but it was not proven that the automated process was a more comfortable environment for this expert than the interview format, or that the automated process produced superior results. In conclusion it is the need for human contact that could thwart the automation of the entire KA process.

Bibliography

[ASAI88] Asai, N., Onishi, I., Mori, S., Otsuka, Y., and Makino, S.. "Development of an AI Supporting System For Knowledge Acquisition and Refinement" In Proceedings of the IEEE International Workshop on Artificial Intelligence for Industrial Applications. New York: IEEE, 1988. Pp 47-51.

[BOOS83] Boose, J.. "Personal Construct Theory and the Transfer of Human Expertise" In Proceedings of the National Conference on Artificial Intelligence. Los Altos: American Association of Artificial Intelligence, 1983. Pp 27-33.

[BOOS86] Boose, J.. Expertise Transfer for Expert System Design. New York: Elsevier Science Publishers B. V., 1986.

[BROW86] Brown, D. C.. "Expert System Problem Solving Applications". In ACM Fourteenth Annual Computer Science Conference: CSC '86 Proceedings. New York: ACM, 1986, Pp 420.

[CHAN84] Chandrasekaran, B.. "Expert Systems: Matching Techniques to Tasks" In Artificial Intelligence Applications for Business. Reitman, W. Norwood: Albex Publishing Corp, 1984, Pp 41-64.

[CHAN85] Chandrasekaran, B.. "Generic Tasks in Knowledge -Based Reasoning : Characterizing and Designing Expert Systems at the "Right" Level of Abstraction". In IEEE Expert, vol1, no. 3 (Fall 1985), 294-299.

[CHAN86] Chandrasekaran, B.. "Generic Tasks in Knowledge -Based Reasoning : High Level Building Blocks for Expert System Design" In IEEE Second Conference on Artificial Intelligence Applications : The Engineering of Knowledge Based Systems. Washington, D. C.: IEEE, 1986. Pp 23-30.

[CHIG86] Chignell, M.. "The Use of Ranking and Scaling Techniques in Knowledge Acquisition". In Expert Systems in Government Symposium. Washington, D. C.: IEEE, 1986. Pp 64-72.

[COOK86] Cooke, N., and McDonald, J.. "A Formal Methodology for Acquiring and Representing Expert Knowledge" Proceedings of the IEEE vol 74 no 10. (October 1986), 1422-1430.

[DAVI87] Davis, J.. "A Task Oriented Framework for Diagnostic and Design Expert Systems" In Foundations of Computer Aided Process Operations: Proceedings of the First International Conference. Amsterdam: Elsevier Science Publishers, B. V., 1987 Pp 695-700.

[GAMM85] Gammack, J., and Young, R.. "Psychological Techniques for Eliciting Expert Knowledge" In Research and Development in Expert Systems. Ed, Bramer, M., New York: The Press Syndicate of the University of Cambridge, 1985. Pp 105 -112.

[GOEL87] Goel, A., and Chandrasekaran, B., Sylvan, D. "Jesse: An Information Processing Model of Policy Decision Making". In Third Annual Expert Systems in Government Conference Proceedings. Washington, D. C.: IEEE, 1987. Pp 178-187.

[HAJE88] Hajek, B., Miller, D., Bhatnagar, R., Stasenko, J., Punch III, W., Yamada, M.. "A Generic Task Approach to A Real Time Nuclear Power Plant Fault Diagnosis and Advisory System" In Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications. New York: IEEE, 1988. Pp 154-160.

[HAYE83] Hayes Roth, F., Waterman, D., Lenant, D., Eds,. Building Expert Systems. Reading, Mass: Addison Wesley Company, 1983.

[JOSE85] Josephson, J., Tanner, M., Smith, J., Svirbely, J., Strohm, P.. "Red: Integrating Generic Tasks to Identify Red Cell Antibodies" In Expert Systems in Government Symposium. New York: IEEE, 1985. Pp 524-531.

[KELL55] Kelly, G.. The Psychology of personal Constructs. New York: W. W. Norton and Company Inc., 1955.

[KRUS78] Kruskal, J., and Wish, M., Multidimensional Scaling. Beverly Hills, California: Sage Publications, 1978.

[MANH86] Manheimer, J., and Kanarski, T "The Application of Psychological Scaling Techniques in Modeling Expert Knowledge" In Proceedings of the IEEE 1986 National Aerospace and Electronics Conference. New York: IEEE, 1986. Pp 935-940.

[McDon82] McDonald, J., Stone, J., Liebelt, L., Karat, J.. "Evaluating a Method for Structuring the User System Interface". In Proceedings for the Human Factors Society 26th Annual Meeting. Ed. Edwards, R.. Santa Monica, California: The Human Factors Society, 1982. Pp 551-554.

[SAS85] SAS Institute. SAS User's Guide: Statistics Version 5 Edition. : SAS Institute, 1985.

[SHEP72] Shepard, R., Romney, A., Nerlove, S., Eds.. Multidimensional Scaling, Theory and Applications in The Behavioral Sciences, Vol 1 and Vol 2. New York: Seminar Press, 1972.

Appendix A

User Manual for the Expert

Table of Contents

1.0 General Description of the Program

The Automated Knowledge Acquisition Tool for Identification of Generic Tasks is a program that defines an expert's knowledge in terms of an Artificial Intelligence concept, termed Generic Tasks. Each identified Generic Task suggests tools to aid the Knowledge Engineer in writing the expert system.

The expert's role in this program is to generate concepts from his area of expertise. To accomplish this task, the expert will enter into the program all the chapter and subchapter titles for a book the expert could write.

After entering all possible titles, the expert will determine the similarity of each title to each generic task characteristic. These ratings are used by a statistical software package that clusters the ratings to identify the Generic Tasks.

The program is very flexible and allows the expert to rate the titles at a later time.

2.0 Running the program.

It is assumed the expert can log on to the RIT undergraduate computer on VAXA into the account containing this program.

At the prompt type:

RUN THESIS and press the RETURN key

The introductory screen below should appear:

Welcome - please read the instructions carefully and respond
to the questions as completely as possible
You will be asked to create a book on your area of expertise
by listing all the chapter and subchapter titles for this book.
You will then be asked to compare your titles to stored concepts
to determine if they are similar.
This program is extremely flexible and you may change any title
you have entered at any time.

PRESS RETURN WHEN READY TO CONTINUE

3.0 Rate Stored Titles

After the introductory screen appears and the expert presses return, the following appears:

At this time you may rate previously stored titles
or enter your titles for your book.
Do you wish to enter titles (Y or N)?

type n and press the RETURN key.

The following appears so that the expert may enter the filename for the titles that will be rated. The filename consists of 8 characters and was created by the expert during the entering of the titles (see 8.2.4 Entering titles).

Enter the filename for the titles

type the filename and press the RETURN key.

If the filename is incorrect the program will ask for the filename again.

At this time the rating introductory screen appears go to Section 8.26 Enter Rating.

4.0 Enter titles

After the introductory screen appears and the expert presses return the following appears

> At this time you may rate previously stored titles
> or enter your titles for your book.
> Do you wish to enter titles (Y or N)?

type y and press the RETURN key.

The following appears

> Please enter a Chapter Title for your book

At this time, type in a title for your chapter. It can be up to 80 characters long (a space is a character). The title is complete when you press the RETURN key. If you have no more titles:

type none and press the RETURN key

If you typed in a title the following will appear

> Please enter a subchapter that would further describe
> this chapter. The subchapters can be in any order and
> there can be any number of them.

At this time type in a subchapter title. It can be up to 80 characters long (a space is a character). The title is complete when you press the RETURN key. If you have no more subchapter titles

type none and press the RETURN key.

At this time the following appears

> Do you wish to edit any titles (Y or N)?

if you wish to edit
type Y and press the RETURN key.

go to section 8.2.5 Editing titles for further instructions
else type N and press the RETURN key.

The sequence of enter a chapter title, enter a subchapter title, editing titles continues until you have entered 'none' for the chapter title.

At this time the following appears

Enter a filename for the titles
It cannot be more than 8 characters long
For example: mytitles
The titles will be stored in a file with the name you just entered and an extension of ".dat"

Type in filename of up to 8 characters. Press the RETURN key when finished. The program will automatically add the .dat extension. The extension is required by the VAXA system. The 8 character filename will be asked for if you choose to rate the titles at some later time. The following appears

At this time you may quit and rate the titles at a
different time or continue with the ratings
of titles.
Do you wish to quit (Y or N)?

If you wish to quit
type Y and press the RETURN key
else
type N and press the RETURN key
go to Section 8.2.6 rating the titles for further details.

5.0 Editing the titles

If you responded yes to the editing the titles, a screen of chapter titles appear, such as:

1. Title 1
2. Title 2
3. Title 3
4. Title 4

Enter (E) to end the editing
Enter (F) for the next screen of titles
Enter (B) for the last screen of titles
Enter the number of the title you need to edit

If the chapter title you wish to edit is on the screen, type the number in front of the title and press the RETURN key. For any of the other responses:
type E, B, or F and press the RETURN key.
Once the chapter title is chosen the following appears:

At this point you can edit this chapter title, add a
new subchapter or edit one of its subchapters.
Do you wish to edit this title (Y or N)?

If you type Y and press the RETURN key, the following appears:

Do you wish to delete the title (Y or N)?

If you wish to remove the title, type Y and press the RETURN key.

If you type N and press the RETURN key, the chapter title is displayed and the following appears

Enter the new title

At this time, enter the new title of up to 80 characters (space is a character) and press the RETURN key.
When this time is complete, you can continue editing or you return to entering titles, See Section 4.0 - Entering titles for further details.

It you type N and press the RETURN key the following appears

Do you wish to add a subchapter (Y or N)?

If you type Y; and press the RETURN key, the following appears

Enter the new subchapter title

At this time, enter the new subchapter title of up to 80 characters (space is a character) and press the RETURN key.
You return to entering titles. See Section 4.0 Entering titles.

If you type N and press the RETURN key a list of that chapter's subchapter titles appears with the same set of possible responses as the chapter titles.

Type the number of the subchapter title you wish to edit and press the return key. The following appears:

Do you wish to delete the title (Y or N)?

If you type Y and press the RETURN key, the title will be deleted.
If you type N and press the RETURN key, the subchapter title appears followed by

Enter the new title.

At this time enter the subchapter title of up to 80 characters (space is a character) and press the RETURN key.
You can continue to edit or return to entering titles See Section 4.0 Entering Titles.

6.0 Rating The Titles

The following screen appears

The next part of the Automated KA tool asks you to rate
your titles against other concepts.

Each title will be displayed on the screen next to a concept
If they are NOT similar - type 0
If they are somewhat similar - type 1
If they are very similar - type 2
The concept will not be directly related to your title but
will be a description of a thought process. If your title can be involved in the thought process described by the concept then
the concept and your title are similar
For example my book is Learning To Drive
TITLE: Driving safely in snow
CONCEPT: this object can be labeled as some element of a hierarchy

How I would answer-

When I entered driving in snow, my thought process was that driving in snow is a part of driving; similar to , yet different from driving in rain and driving on dry roads. So driving in snow is part of a driving hierarchy and, therefore, the concept and title are similar. I would type 2.
Help is available for every rating.

The title appears

TITLE:

and the concept appears

CONCEPT:

And the choices appear

Are they
not similar - 0
somewhat similar - 1
very similar - 2
Enter 0, 1, 2 or (H) for Help

Type H and press the RETURN key and a further explanation of the concept appears followed by the rating choices.

Type the rating number 0, 1, or 2 and press the RETURN key.
This continues until all titles have been compared. The program then terminates with

The program is ending - Thank you.

Appendix B

User Manual for the Knowledge Engineer

Table of Contents

1.0 General Description

Automated Knowledge Acquisition Tool for Identification of Generic Tasks is a program that defines an expert's knowledge in terms of generic tasks. Each identified generic task suggests knowledge representations and strategies for the expert system. See section 2.3 for further definition of generic tasks.

The expert will interact with the user interface to generate a list of concepts. These concepts will then be compared by the expert to determine their similarity to concepts representing generic tasks. These ratings are stored in a file called Ratings.dat. This is the file that SAS uses to cluster the ratings to generic tasks. The knowledge engineer can use the results of the SAS program to identify the generic tasks.

2.0 Preparing the Expert

It is assumed that the knowledge engineer is designing an expert system with the aid of one or more experts and is familiar with the concepts of generic tasks.

The knowledge engineer must explain the following to the expert

1. That the expert will input chapter titles to the computer which the expert would choose if a book could be written about his expertise.
2. The expert must know about the basic keys on a computer key board, and logging on to the RIT undergraduate computer VAXA cluster.

3. The expert should be comfortable with this computerized procedure and realize that it will shorten the time needed for knowledge acquisition.

The user interface section of the program is flexible and contains understandable directions. The knowledge engineer should be familiar with the expert's user manual (section 8.2 of this document). Depending on the degree of computer literacy of the expert, the knowledge engineer may wish to go over the Expert's User Manual with the expert.

The primary concern is that the expert be comfortable with a computer and see the benefits of using this program.

3.0 Running the SAS

It is assumed that the knowledge engineer can log onto the RIT undergraduate computer VAXA cluster into the account containing this program. It is assumed the expert has previously entered his titles and has rated the titles.

The ratings from the expert are in a file called ratings.dat. This file is accessed by a program, thesis.sas. The SAS program executes a FASTCLUS procedure that results in two files. Thesis.lis contains the statistical results of the program and thesis.log contains the actions and errors, if any, from the program. To run the SAS program, at the prompt, type

thesis.sas and press the return key

the prompt will appear again at the conclusion of the program. To see the results on the screen type

thesis.lis | more and press the return key

The results can be redirected to the printer. See section 8.3.4 of this document for analysis of the results. Each time the thesis.sas program is executed, thesis.lis and thesis.log are overwritten.

4.0 Analysis of the Results

The results from the thesis.sas program is in thesis.lis. The analysis of the results will be done by table.

TABLE 1

The first table lists the seeds used for the FASTCLUS procedure. They list the ideal conditions to identify the generic tasks. The ideal would be that the expert stated the highest similarity rating of a title to only one task. This table is the same for any execution of thesis.sas.

TABLE 2

Table 2 lists the initial seeds again but this time in greater precision. This table is the same for any execution of thesis.sas.

TABLE 3

Table 3 lists the changes to the initial seeds. Each observation is assigned to the nearest seed and the seeds are recomputed as the means of the cluster. This iteration is set to 10. The Knowledge Engineer can enter thesis.sas and change the line containing MAXITER = 10, to create another analysis.

TABLE 4

Table 4 lists each observation, cluster it was assigned to, and its distance from the seed. Each title generated by the expert resulted in 3 observations. For example the first title listed in the expert defined title file resulted in observations 1, 2, and 3.

TABLE 5

Table 5 is a cluster summary. For each cluster it states

FREQUENCY number of observations in the cluster

RMS STD DEVIATION root mean square distance between the observation & the cluster.

MAXIMUM DISTANCE FROM SEED TO OBSERVATION

NEAREST CLUSTER CENTROID DISTANCE distance between the means of the current cluster and the nearest other cluster.

The frequency can be distorted by more observations so it is not used in the analysis.

TABLE 6

Table 6 is the statistics for variables. For each task it states:

TOTAL STD - standard deviation

WITHIN CLUSTER pooled within cluster standard deviation

R -SQUARED predicting the variable from cluster

RSQ / (1-RSQ) ratio of between cluster variance to within cluster variance.

These statistics and PSEUDOF STATISTIC, OVERALLR, SQUARED AND CUBIC CLUSTERING CRITERION are not necessary to identify the tasks

TABLE 7

Table 7 is the cluster means. The highest column values in a row are used to determine which tasks are to be associated with that cluster. This table will be used in the analysis.

TABLE 8

Table 8 is the cluster standard deviations and are not used for the identification of generic tasks.

TABLE 9

Table 9 is the beginning of the results from the CANDISC procedure that is used to further cluster the identified tasks. The tables are not used in the analysis, except the raw canonical coefficients that are to be used to determine which tasks are in CAN1 and CAN2 for the plot. The plot at the end is important in the analysis.

The analysis

For example:

Consider this table of cluster means from a thesis.lis.

Cluster	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
1	2.0	1.0	1.0	2.0	1.727	1.1818
2	.625	1.87	1.0	1.75	125	2.0
3	0.0	1.33	1.33	2.0	1.667	2.0
4	.1875	.43	.0625	1.9375	0.0	2.0
5	0.0	.375	0.0	2.0	2.0	2.0
6	1.5	2.0	0.0	0.0	0.0	2.0

Looking at the cluster means table horizontally:

Cluster 4 has task 6 and task 4 (largest column numbers)

Cluster 1 has task 1 and task 4 followed by task 5, 6, 2, 3

Cluster 2 has task 6, 2, 4, 3

Cluster 5 has task 4 and task 5 and task 6

Cluster 3 has 6, 4, 5, 2, 3

Cluster 6 has 6, 2. 1

Looking at the cluster means table vertically:

Task 1 is highest in cluster 1

Task 2 is highest in cluster 6

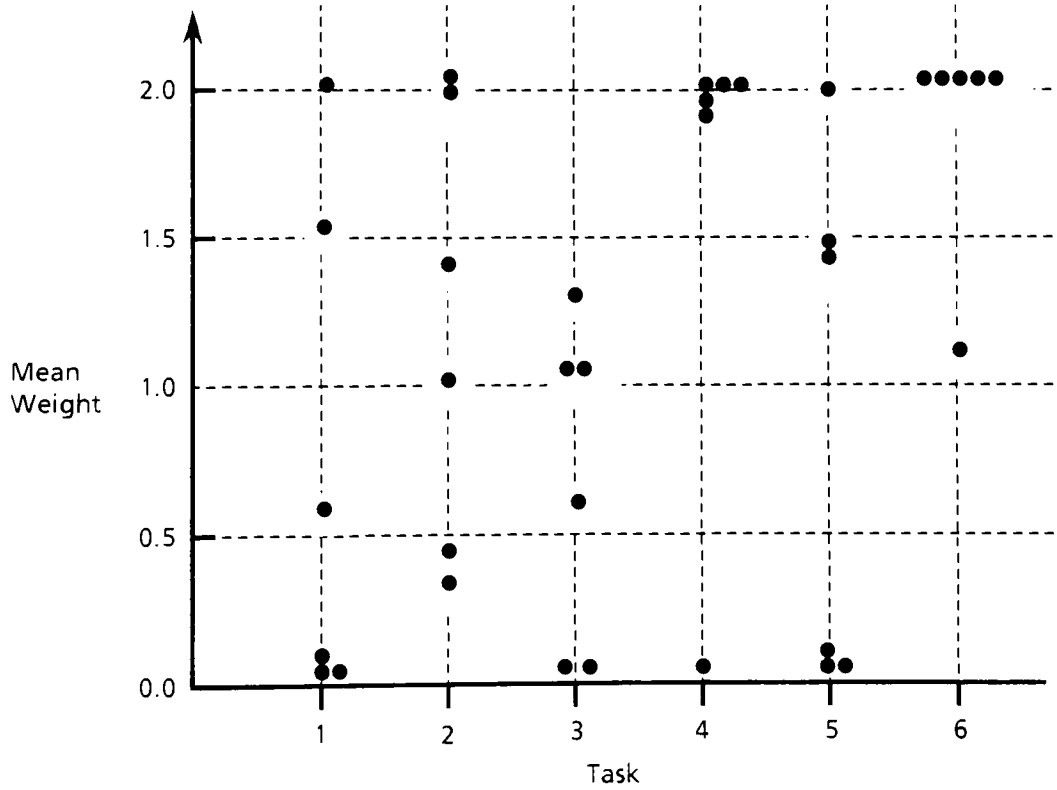
Task 3 is highest in cluster 3

Task 4 is highest in cluster 1, 3, 5

Task 5 is highest in cluster 5

Task 6 is highest in cluster 1, 2, 3, 4, 5, 6

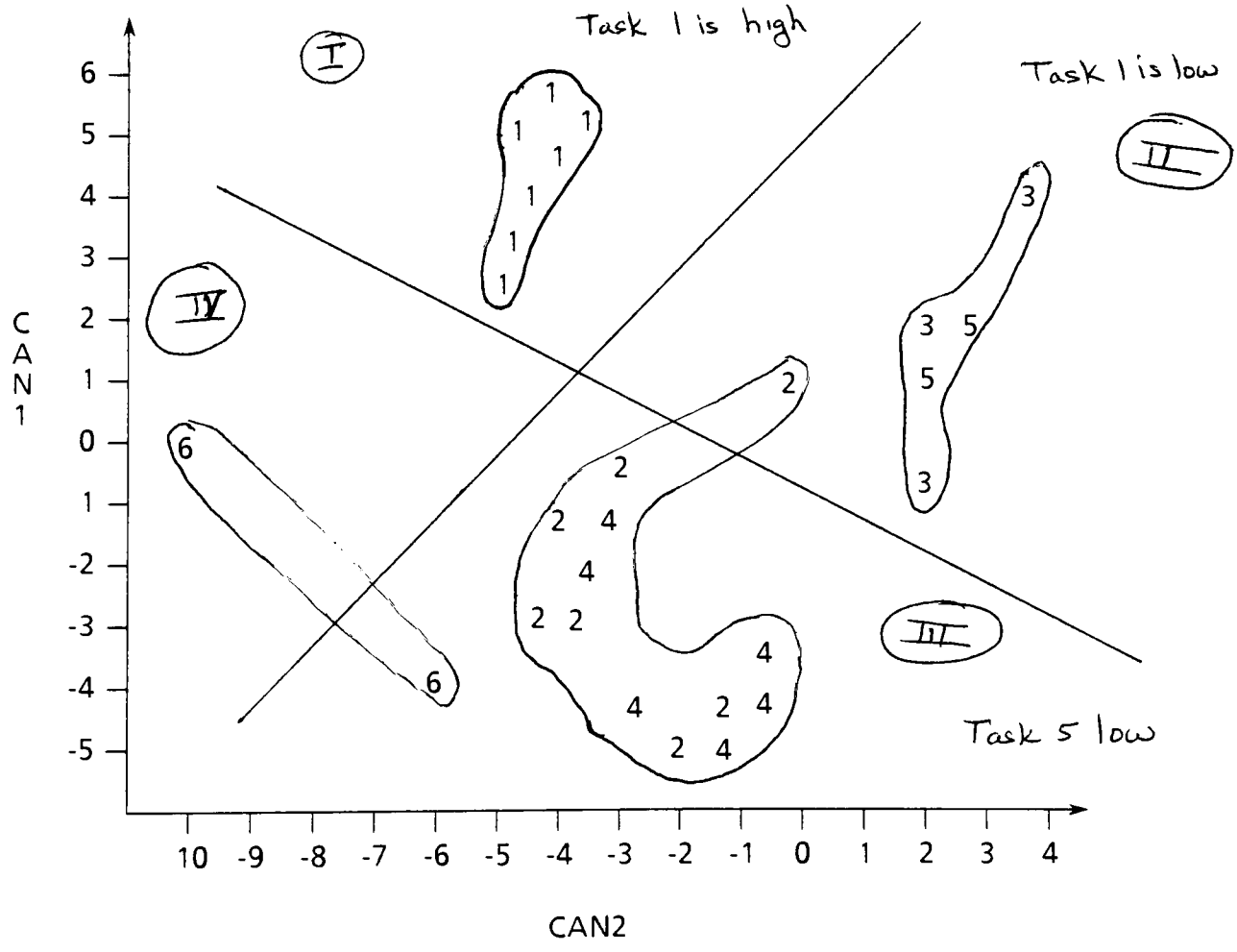
To visualize the results of the cluster means analysis, make a graph and plot the means as below.



Task 6 has all of its marks above 1, implying a strong presence of task 6 across all clusters.
Task 4 also is strong across the majority of clusters.
Task 2 is strong in the majority of clusters.
Task 5 is strong in half of the clusters.
Task 1 and task 3 are present in the domain, but are not strong.

Analyze the plot from the CANDISC procedure. From the raw canonical coefficients, under CAN1 find the tasks with high positive numbers. CAN1 represents these tasks. Do the same for CAN2. These suggest the tasks which differentiate the clusters. Analyze the plot itself by encircling the clusters into groups.

Plot of Conical Variables Identified by Cluster



Note: 20 observations hidden

Analyze the clusters that combine into a group. This implies their differences are not strong enough to separate them. For example, cluster 3 and 5 differ on tasks 2 and 3 but that is not as important as the fact that they share Task 1, 4, 5, and 6. The cluster means determined how they were different and similar.

Next separate the groups visually into sectors. This demonstration example has 4 sectors. analyze the differences between the sectors using the cluster means.

Sections I and II differ from sections III & IV on Task 5. Task 5 is strong in sections I & II.

Sections I and III differ on Task 1. Task 1 is strong in sectors I & IV.

The analysis suggests Task 1 is present and fairly strong but as a separate task not highly interconnected to the other tasks in the domain. For example, a financial planning expert may first classify a client prior to beginning the plan. This would yield a task 1 result similar to this.

Task 2 is present as part of the overall domain problem. This implies that the majority of the aspects of this domain can change state and simulate reality.

Task 3 is present weakly as part of the overall domain problem. This implies that in some aspects of the domain, knowledge can be obtained from the presence of other aspects.

Task 4 is strong in this domain. This implies that the majority of the domain is a plan selection and refinement.

Task 5 is weakly present in this domain. This implies that some aspects of the domain are concerned with "goodness of fit" hypothesis matching.

Task 6 is the dominant task of this domain. This implies that Assembly of explanations is the primary goal of the expert. A further description of the task knowledge structures and strategies are in Section 2.3 of this document. The knowledge engineer can present the analysis to the expert as a basis of discussion. The expert should confirm the results and the first interview session begins.

The following table maps the task numbers to a task title for easy reference

Task	Description
1	Classification
2	State Abstraction
3	Knowledge - Directed Information Passing
4	Plan Selection and Refinement
5	Hypothesis Matching
6	Assembly of Explanatory Hypothesis

Appendix C
Implementation Code

File: Thesis.h

```
/*****************************************************************************/
/*
**      Thesis :        Automated KA Tool for Identification of Generic Tasks
**
**      Author :        Arlene J. Buck
**
**      Contents :      This file is the header file for this thesis. It
**                      contains all structures and global variables
**
*****************************************************************************/
#include <stdio.h>

/*
** define the maximum characters in a concept and the number of concepts
** that can be displayed on a screen. This controls the scrolling of the
** screen.
*/
#define MAX_CONCEPT_LENGTH      81
#define MAX_TITLE_PER_SCREEN     5
#define MAX_FILENAME_LENGTH     13

/*
** define the basic concepts of a null pointer
*/
#define NULL                     0

/*
** define the structure that describes a subchapter. It consists of a concept
** and a pointer to the next subchapter.
*/
struct subchap
   {
   char           concept [MAX_CONCEPT_LENGTH];
   struct subchap *next;
   struct subchap *prior;
   };

/*
** define the structure that describes a chapter. It consists of a concept,
** a pointer to the next chapter and a pointer to a list of subchapters
*/
struct chap
   {
   char           concept [MAX_CONCEPT_LENGTH];
   struct chap    *next;
   struct chap    *prior;
   struct subchap *sublist;
   };
```

```
/*
** define the global variables used in this system
*/
int             chapters;         /* number of chapters in the system */
struct chap     *head;            /* ptr to head of chapter list */
struct chap     *latest_chap;     /* ptr to latest chapter prior to editing */
struct subchap *latest_subchap;  /* ptr to latest subchapter prior to edit */

/*
** define the filename containing the titles
*/
char     filename [MAX_FILENAME_LENGTH];
```

File: Thesis.c

```
/*****************************************************************************
**
**    Program  :  Automated KA Tool for Indentification of Generic Tasks
**
**    Author   :  Arlene J Buck
**
**    Contents :  This file contains the main routine of this thesis program.
**                It controls the actions of interfacing with the expert to
**                generate a list of concepts, rate those concepts against
**                characteristics of generic tasks. The list of concepts
**                and ratings are stored as files for later access. The
**                file of ratings will be the input into the SAS FASTCLUS
**                procedure. The SAS procedure result is the analysis of
**                the ratings clustering to the generic tasks. The SAS
**                procedure can be run by typing
**
**                      SAS thesis.sas
**
**                The results are in a file called thesis.lis.
**
*****************************************************************************/

#include "thesis.h"

/*
** define all the routines called in main as externals. They are described
** in main as they are called
*/
extern screen1 ();
extern get_chapter ();
extern write_titles_to_file ();
extern descibe_ratings ();
extern screen2 ();
extern get_ratings ();

main ()

   {
   /*
   ** define an input character for the expert's answer and an index
   */
   char  ch;
   int   i;

   /*
   ** define a file pointer for the titles if the expert wants only to rate
   ** titles
   */
   char  *fpTitles;

   /*
   ** set the pointer to the latest chapter and subchapter, and head of the
   ** chapter linked list to null
   */
   latest_chap = NULL;
```

```
latest_subchap = NULL;
head = NULL;

/*
** call routine to print the first screen explaining to the expert
** the concepts and directions of this program
*/
screen1 ();

/*
** prompt the expert to determine if the expert is to input titles or
** just rate already existing titles
*/
printf ("\n\nAt this time you may rate previously stored titles or \n");
printf ("enter titles for your book. \n");
printf ("Do you wish to enter titles (Y or N)?\n");
do
  {
  ch = getchar ();
  while (getchar() != '\n');
  }
  while (!((ch == 'Y')||(ch =='y')||(ch =='N')||(ch =='n')));
if ((ch == 'Y') || (ch == 'y'))
  {

  /*
  ** call the routine to prompt the expert to input the concepts of his
  ** domain as chapters in a book
  */
  get_chapter ();

  /*
  ** call the routine that writes the concepts to a file
  */
  write_titles_to_file ();


  /*
  ** prompt the expert to determine if he wishes to quit or continue
  */
  printf ("\n\nAt this point you can quit \n");
  printf ("or continue to the rating of the titles. \n");
  printf ("Do you wish to quit (Y or N)?");
  do
     {
     ch = getchar ();
     while (getchar () != '\n');
     }
     while (!((ch =='Y')||(ch =='y')||(ch =='N')||(ch =='n')));
  if ((ch == 'N') || (ch == 'n'))
     {

     /*
     ** call the routine to print a screen of directions to describe the
     ** ratings to the expert
     */
     screen2 ();
```

```
      /*
      ** call the routine to prompt the expert for the ratings
      */
      get_ratings ();
      }
   }
 else
   {
   /*
   ** the expert wishes to only rate titles
   ** prompt the expert for the name of the titles file and verify it as a
   ** valid filename
   */
   do
     {
     printf ("\n\nEnter the name you typed in for the group of titles\n");
     i = 0;
     while ((ch = getchar ()) != '\n')
       {
       if (i < MAX_FILENAME_LENGTH)
         {
         filename [i] = ch;
         i++;
         }
       }
     filename [i] = '.';
     filename [i+1] = 'd';
     filename [i+2] = 'a';
     filename [i+3] = 't';
     filename [i+4] = '\0';

     /*
     ** open the titles file for reading
     */
     fpTitles = fopen (filename,"r");
     }
     while (fpTitles == NULL);

 /*
 ** close the file now that it has been verified
 */
 close (fpTitles);

  /*
  ** call the routine to print a screen of directions to describe the
  ** ratings to the expert
  */
  screen2 ();

  /*
  ** call the routine to prompt the expert for the ratings
  */
  get_ratings ();
  }
printf ("\n\nThis program is ending - Thank you \n");
}
```

File: Screen1.c

```
/*****************************************************************************
**   Program:      Automated KA Tool for Identification of Generic Tasks
**
**   Author:     Arlene J Buck
**
**   Contents:  This file contains the routine that explains this
**              program to the expert.
**
*****************************************************************************
*/

Screen1 ()

{
printf ("Automated KA Tool For Identification of Generic Tasks\n\n");
printf ("Welcome - please read the instructions carefully and respond\n");
printf ("to the questions as completely as possible\n\n");
printf ("You will be asked to create a book on your area of expertise\n");
printf ("by listing all the chapter and subchapter titles for this book\n\n");
printf ("You will then be asked to compare your titles to stored concepts\n");
printf ("to determine if they are similar\n\n");
printf ("This program is extremely flexible and you may change any title\n");
printf ("you have entered at any time \n\n");
printf ("\n\n\n\n PRESS RETURN WHEN READY TO CONTINUE\n");
getchar();
}
```

File: Get_Concepts.c

```
/*****************************************************************************
**
**    Program  :  Automated KA Tool for Indentification of Generic Tasks
**
**    Author   :  Arlene J Buck
**
**    Contents :  This file contains the routines that prompt the expert for
**                chapter titles and stores these chapters in a linked list.
**                Each chapter can have any number of sub-chapters which
**                are stored as a linked list. A pointer to this sub-chapter
**                list is stored in the concept structure. Each chapter and
**                subchapter generated by the expert is considered as one
**                concept describing this domain.
**
*****************************************************************************/

#include "thesis.h"


/*
**    Routine :   get_chapter
**
**    Purpose :   to prompt the expert for chapter and sub-chapter titles
**                for a book on his domain. A chapter is prompted for
**                and then that chapter's sub-chapters. It then queries
**                the expert whether all titles have been input. It then
**                queries the expert to edit the entered titles.
**
**    Called by:  main program in thesis.c
**
**    Calls   :   read_chapter in this same file
**                sub_chapter in this same file
**                edit in edit.c
**
*/

get_chapter ()

   {
   /*
   ** define a boolean to control looping to query for all titles and a ch
   ** to store the expert's response to queries
   */
   int   not_finished;
   char  ch;

   not_finished = 1;

   /*
   ** until the expert has entered all titles prompt him for titles
   */
   while (not_finished)
      {
      printf ("\n\n\nPlease enter a Chapter Title for your book \n");
      printf ("Enter 'none' when finished\n");
```

```
      /*
      ** call routine that reads in the title and stores it into the linked
      ** list of chapters
      */
      read_chapter (&not_finished);

      /*
      ** call the routine that prompts and reads in all sub-chapter titles
      ** and stores them into the linked list of sub-chapters for that
      ** chapter
      */
      if (not_finished)
        sub_chapter ();

      /*
      ** ask the expert if editing of prior titles is needed and if so
      ** call the routine to edit
      */
      printf ("\n\n\nDo you wish to edit any previous titles (Y or N)\n");
      do
        {
        ch = getchar ();
        while (getchar () != '\n');
        }
        while (!((ch =='Y')||(ch =='y')||(ch =='N')||(ch =='n')));
      if ((ch == 'Y') || (ch == 'y'))
         edit ();
      }
   }


/*
**    Routine :   read_chapter
**
**    Purpose :   reads in the title, allocate memory, and store the
**                title in the linked list of chapters.
**
**    Called by:  get_chapters in this same file
**
**    Calls   :   no other routines
**
*/

 read_chapter (
   int *not_finished
   )

   {
   /*
   ** define a linked list node structure for this new entry
   */
   struct chap    *new_chap;

   /*
   ** define the index and character of the title
   */
```

```
int                i;
char            ch;

/*
** allocate memory for this new entry and initialize it.
*/
new_chap = (struct chap *) malloc (sizeof (struct chap));
new_chap->next = NULL;
new_chap->sublist = NULL;
new_chap->prior = NULL;

/*
** read in the title and store it in the new entry
*/
i = 0;
while ((ch = getchar ()) != '\n')
  {
  if (i < MAX_CONCEPT_LENGTH - 1)
    {
    new_chap->concept[i] = ch;
    i++;
    }
  else
    printf ("\nYOUR TITLE HAS BEEN TRUNCATED \n");
  }
new_chap->concept[i] = '\0';

if (strcmp (new_chap->concept,"none") == NULL)
  {
  free (new_chap);
  *not_finished = 0;
  return;
  }

/*
** place the new entry into the linked list
*/
if (head == NULL)
   /*
   ** this is the first entry
   */
   {
   head = new_chap;
   new_chap->prior = NULL;
   }
else
   {
   /*
   ** add it to the end. The end of the list is latest_chap.
   */
   latest_chap->next = new_chap;
   new_chap->prior = latest_chap;
   }

/*
** update the pointers and the number of chapters in the system
*/
```

```
   latest_chap = new_chap;
   latest_subchap = NULL;
   chapters ++;
   }

/*
**    Routine :   sub_chapter
**
**    Purpose :   controls the loop for reading in all sub_chapters for
**                the latest chapter added.
**
**    Called by:  get_chapters in this same file
**
**    Calls   :   read_subchapter in this same file
**
*/
 sub_chapter ()
   {
   /*
   ** define a boolean to control loop and a char for the expert's input
   */
   int   not_finished;
   char  ch;

   /*
   ** until the expert has entered all sub chapters prompt him for a title
   */
   not_finished = 1;
   printf ("\n\n\nPlease enter a subchapter that would further describe\n");
   printf ("this chapter. The subchapters can be in any order and\n");
   printf ("there can be any number of them. Enter 'none' when finished\n");
   read_subchapter (&not_finished);
   while (not_finished)
      {
      /*
      ** call a routine to read in and store the subchapter title
      */
      printf ("\n\nEnter another subchapter for that same chapter title\n");
      printf ("Enter 'none' when finished\n");
      read_subchapter (&not_finished);

      }
   }

/*
**    Routine :   read_subchapter
**
**    Purpose :   reads in the title and stores it into the linked
**                list of subchapters for the latest chapter.
**
**    Called by:  sub_chapter in this same file
**
**    Calls   :
**
*/

 read_subchapter (
```

```
int   *not_finished
)

{
/*
** define a pointer to a new node for the linked list
*/
struct subchap    *new_subchap;

/*
** define the index and the input character for the title
*/
int             i;
char            ch;

/*
** allocate memory for this new entry and initialize it.
*/
new_subchap = (struct subchap *) malloc (sizeof (struct subchap));
new_subchap->next = NULL;
new_subchap->prior = NULL;

/*
** read in the title and store it in the new entry
*/
i = 0;
while ((ch = getchar ()) != '\n')
  {
  if (i < MAX_CONCEPT_LENGTH)
    {
    new_subchap->concept[i] = ch;
    i++;
    }
  else
    printf ("\n\n YOUR TITLE HAS BEEN TRUNCATED\n");
  }
new_subchap->concept [i] = '\0';

if (strcmp (new_subchap->concept, "none") == NULL)
  {
  free (new_subchap);
  *not_finished = 0;
  return;
  }

/*
** place the new entry into the linked list
*/
if (latest_chap->sublist == NULL)
   /*
   ** this is the first entry in the sublist for this chapter
   */
   {
   latest_chap->sublist = new_subchap;
   new_subchap->prior = NULL;
   }
else
```

```
    {
    /*
    ** add it to the end. The end of the list is latest_chap.
    */
    latest_subchap->next = new_subchap;
    new_subchap->prior = latest_subchap;
    }

/*
** update the latest subchapter pointer
*/
latest_subchap = new_subchap;
}
```

File: Edit.c

```
/*****************************************************************************
**
**    Title     :  Automated KA Tool for Identification of Generic Tasks
**
**    Author    :  Arlene J Buck
**
**    Contents  :  This file contains the routines that prompt the expert for
**                 editing of titles. First the chapter is queried for a
**                 match and then, if necessary the subchapters are queried.
**                 The expert can now edit the chosen chapter or subchapter.
**                 A few titles are displayed at any one time on the screen.
**                 The expert can then choose one of the displayed, or the
**                 last screen, or the next screen. i.e.
**
**                    5.Title5
**                    6.Title6
**                    6.Title6
**                    7.Title7
**
**                    Enter Forward (F), Backward (B), Number of title, or
**                          End Edit (E) ?
**
*****************************************************************************/

#include "thesis.h"

/*
**    Routine :   edit
**
**    Purpose :   to control the prompting of the expert for chapter or
**                sub-chapter title that he wishes to edit.
**
**    Called by:  get chapter program in get_concepts.c
**
**    Calls   :   edit_subchapter in this same file
**                edit_chapter in this same file
**                edit_control in this same file
**
*/

edit ()
   {
   /*
   ** define a number for each screen so that each screen can be recalled
   */
   int         screen_num;

   /*
   ** define an input character that defines the chapter number in a
   ** screen group and a pointer to a chapter
   */
   char        ch;
   struct chap *chap_ptr;

   /*
```

```
** define an input character for repeating the editing process
*/
char          answer;

/*
** define a return status variable for the results from any lower level
** call
*/
int          status;

/*
** If at any time the expert chose to cancel this editing session,
** the status returned from the following calls is the opposite
** of what is necessary to continue this routine.
**
** call the routine that prints the edit screen and handles the expert's
** response until a chapter title has been selected or the expert wants
** to exit the edit routine. This routine is generic for chapter titles
** and subchapter titles so a null ptr is passed for the titles not
** wanted at this time.
*/
do
  {
  status = edit_control (head, NULL, &screen_num, &ch);
  if (status == 1)
     {
     /*
     ** a chapter title was found. Call a routine to edit the chapter.
     */
     status = edit_chap (screen_num, ch, &chap_ptr);
     if (status == 0)
        {
        /*
        ** the expert wants to edit a subchapter
        */
        status = edit_control (NULL, chap_ptr->sublist, &screen_num, &ch);
        if (status == 1)
           /*
           ** subchapter was located so edit the title
           */
           edit_subchapter (screen_num, ch, chap_ptr);
         }
     if ((ch != 'E') || (ch != 'e'))
       {
       /*
       ** allow the expert to continue editing
       */

       printf ("Do you wish to continue editing (Y or N)\n");
       do
         {
          answer = getchar();
         while (getchar() != '\n');
         }
         while (!((answer == 'y')||(answer == 'Y')||(answer=='N')||
                  (answer == 'n')));
       }
```

```
         else
            answer = 'n';
         }
      else
         answer = 'n';
      }
      while ((answer == 'y')||(answer == 'Y'));
   }

/*
**    Routine :   edit_control
**
**    Purpose :   processes the expert's response to an edit screen
**
**    Called by:  edit in this same file
**
**    Calls   :   print_edit_command in this same file
**                print_edit_title in this same file
**
*/

int edit_control (
   struct chap    *chap_ptr,
   struct subchap *sublist_ptr,
   int            *screen_num,
   char           *ch
   )

   {
   /*
   ** define a flag for determining the end of an editing session - the
   ** expert chose to cancel or the title was found.
   */
   int            end_session;

   /*
   ** until there is end this session, print out a screen of titles
   */
   *screen_num = 0;
   end_session = 0;
   while (!end_session)
      {
      /*
      ** to make this routine generic there has to be an ability to
      ** handle chapter and subchapter titles. Therefore send
      ** both pointers to the next routine that prints the screen and
      ** let that routine choose the right pointer.
      */
      print_edit_title (&end_session, *screen_num, chap_ptr, sublist_ptr);

      /*
      ** print the response options the expert can chose
      */
      print_edit_command (*screen_num, end_session);

      /*
      ** get  and verify the response and process it
```

```
        */
        do
          {
          *ch = getchar();
          while (getchar () != '\n');
          if ((*ch == 'f')||(*ch == 'b')||(*ch == 'e'))
              *ch = toupper (*ch);
          }
          while (!((*ch == 'F') || (*ch == 'B') || (*ch == 'E') ||
                ((*ch >= '0') && (*ch <= '9'))));
        if (*ch == 'F')

           /*
           ** go to the next screen of titles
           */
           (*screen_num)++;

        if (*ch == 'B')

           /*
           ** go to the previous screen of titles
           */
           {
           (*screen_num)--;
           end_session = 0;
           }
        if (*ch == 'E')

           /*
           ** expert has chosen to quit
           */
           end_session = 1;
        if ((*ch >= '0') && (*ch <= '9'))

           /*
           ** a title has been chosen
           */
           end_session = 1;
        }
     if (*ch == 'E')
        return (0);
     else
        return (end_session);
     }

/*
**    Routine :   print_edit_title
**
**    Purpose :   prints out a screen of titles reguardless of whether
**                they are chapter or subchapter titles
**
**    Called by:  edit_control in this same file
**
**    Calls   :
**
*/
```

```
int print_edit_title (
   int            *end_session,
   int            screen_num,
   struct chap    *chap_ptr,
   struct subchap *sublist_ptr
   )

   {
   /*
   ** define a for loop counters, temporary title pointers and an a number
   ** for every title
   */
   int            c;
   struct chap    *temp_cptr;
   struct subchap *temp_sptr;
   int            i;

   i = 1;

   /*
   ** determine whether chapter or subchapter
   */
   if (chap_ptr == NULL)
      {
      /*
      ** find the titles for this screen group by walking a pointer
      ** thru the list of titles until that screen goup is reached
      */
      temp_sptr = sublist_ptr;
      while (
         (temp_sptr != NULL) &&
                     (((float)i/MAX_TITLE_PER_SCREEN) <= screen_num)
         )
         {
         temp_sptr = temp_sptr->next;
         i++;
         }
      printf ("\n\n\n\n\n");

      /*
      ** print out a screen full of titles
      */
      for (c=1; c <= MAX_TITLE_PER_SCREEN; c++)
         {
         if (temp_sptr != NULL)

            /*
            ** print out the title
            */
            printf("%d. %s\n", c,temp_sptr->concept);
         else
            {
            /*
            ** the end of the titles was reached
            */
            *end_session = 1;
            break;
```

```
            }
         temp_sptr = temp_sptr->next;
         }
      }
   else
      {
      /*
      ** the chapter titles are called for
      */
      temp_cptr = chap_ptr;
      while (
         (temp_cptr != NULL) &&
              (((float)i/MAX_TITLE_PER_SCREEN) <= screen_num)
         )
         {
         temp_cptr = temp_cptr->next;
         i++;
         }
      printf ("\n\n\n\n\n");

      /*
      ** print out a screen full of titles
      */
      for (c=1; c <= MAX_TITLE_PER_SCREEN; c++)
         {
         if (temp_cptr != NULL)

            /*
            ** print out the title
            */
            printf("%d. %s\n", c,temp_cptr->concept);
         else
            {
            /*
            ** the end of the titles was reached
            */
            *end_session = 1;
            break;
            }
         temp_cptr = temp_cptr->next;
         }
      }
   /*
   ** return - the list of titles was printed
   */
   }

/*
**    Routine :   print_edit_command
**
**    Purpose :   prints out appropriate options the expert can choose
**                from at each list of titles to edit.
**
**    Called by:  edit_control in this same file
**
**    Calls   :
**
```

```
*/

print_edit_command (
   int   screen_num,
   int   end_session
   )
   {
   /*
   ** depending on the screen group, the options to the expert are
   ** different
   */
   printf ("-----------------------------------------------------\n");
   if (screen_num == 0)
     {
     if (end_session == 1)
       /*
       ** this is the one and only screen
       */
       {
       printf ("Enter (E) to end the editing\n");
       printf ("Enter the number of the title you need to edit\n");
       }
     else
       /*
       ** first group of titles so no backward option
       */
       {
       printf ("Enter (F) for the next screen of titles\n");
       printf ("Enter (E) to end the editing\n");
       printf ("Enter the number of the title you need to edit\n");
       }
     return;
     }
   if (end_session == 1)

      /*
      ** last group of titles so no forward option
      */
      {
      printf ("Enter (B) for the last group of titles\n");
      printf ("Enter (E) to end the editing\n");
      printf ("Enter the number of the title you need to edit\n");
      }
    else
      {
      /*
      ** all options are available
      */
      printf ("Enter (F) for the next group of titles\n");
      printf ("Enter (B) for the last group of titles\n");
      printf ("Enter (E) to end the editing\n");
      printf ("Enter the number of the title you need to edit \n");
      }
   /*
   ** options are displayed so return
   */
   }
```

```
/*
**    Routine :   edit_chap
**
**    Purpose :   determines if the expert wants to edit a chapter or a
**                subchapter. If it's the chapter title this routine prompts
**                the expert for the new title and enters it into the
**                list of chapter titles replacing the old one.
**
**    Called by:  edit in this same file
**
**    Calls   :   add_subchapter
**
*/

int edit_chap (
   int              screen_num,
   char             ch,
   struct chap      **chap_ptr
   )

   {
   /*
   ** define a variable to be the numeric value of the input - ch, and define
   ** a for loop counter
   */
   int              num, i;

   /*
   ** define a temporary pointer to the list of chapters
   */
   struct chap      *temp_ptr;

   /*
   ** define a  varaible for the expert's responses
   */
   char             answer;

   /*
   ** the input ch contains the offset of the title from the beginning of the
   ** list of chapter titles. This number can be used to find the exact title
   ** the expert chose.
   */
   num = ch - '0';
   num = num + (screen_num * MAX_TITLE_PER_SCREEN);
   temp_ptr = head;
   for (i=1; i < num; i++)
      temp_ptr = temp_ptr->next;
   *chap_ptr = temp_ptr;

   /*
   ** ask the expert if he wants to edit this title or it's subchapters
   */
   printf
      ("\n\n\nAt this point, you can edit this chapter title, add a\n");
   printf
      ("subchapter, or edit one of this chapter's subchapters\n");
```

```
printf
   ("Do you wish to edit this chapter's title (Y or N)?\n");

do
  {
  answer = getchar ();
  while (getchar () != '\n');
  }
  while (!((answer == 'Y')||(answer == 'y')||
        (answer =='n')||(answer =='N')));
if ((answer == 'Y') || (answer == 'y'))
   {
   /*
   ** ask the expert if he wishes to delete the title
   */
   printf ("\n\nDo you wish to delete the title (Y or N)\n");
   do
     {
     answer = getchar();
     while (getchar() != '\n');
     }
     while (!((answer == 'Y')||(answer =='y')||(answer =='N')||
           (answer=='n')));
   if ((answer == 'Y') || (answer == 'y'))
     {
     if ((temp_ptr->prior == NULL) && (temp_ptr->next == NULL))
       {
       /*
       ** only element of list of chapters
       */
       latest_chap = NULL;
       latest_subchap = NULL;
       head = NULL;
       chapters = 0;
       free (temp_ptr);
       return (1);
       }
     if ((temp_ptr->prior == NULL) && (temp_ptr->next != NULL))
       {
       /*
       ** first element in list of chapters
       */
       head = temp_ptr->next;
       temp_ptr->next->prior = NULL;
       chapters--;
       free (temp_ptr);
       return (1);
       }
     if (temp_ptr->next == NULL)
       {
       /*
       ** last element of list of chapters
       */
       latest_chap = temp_ptr->prior;
       latest_subchap = temp_ptr->prior->sublist;
       temp_ptr->prior->next = NULL;
       chapters--;
```

```
        free (temp_ptr);
        return (1);
        }
      /*
      ** middle element of list of chapters
      */
      temp_ptr->next->prior = temp_ptr->prior;
      temp_ptr->prior->next = temp_ptr->next;
      free (temp_ptr);
      chapters --;
      return (1);
      }

    /*
    ** display the chapter title and ask the expert to retype the title
    ** Then store the new title
    */
    printf ("\n\n\n%s\n", temp_ptr->concept);
    printf ("\n\n Enter the new title \n");
    i = 0;
    while ((answer = getchar ()) != '\n')
      {
      if (i < MAX_CONCEPT_LENGTH - 1)
        {
        temp_ptr->concept[i] = answer;
        i++;
        }
      else
        printf ("\n YOUR TITLE HAS BEEN TRUNCATED \n");
      }
    temp_ptr->concept[i] = '\0';
    return (1);
    }
else
    {
    /*
    ** edit the subchapter - ask expert if he wishes to add subchapter
    */
    printf
    ("Do you wish to add a subchapter title (Y or N)?\n");
    do
      {
      answer = getchar ();
      while (getchar () != '\n');
      }
      while (!((answer == 'Y')||(answer == 'y')||
            (answer =='n')||(answer =='N')));
    if ((answer == 'Y') || (answer == 'y'))
       {
       add_subchap (temp_ptr);
       return (1);
       }
    else

       /*
       ** edit the subchapter
       */
```

```
         return (0);
      }
   }

/*
**    Routine :   edit_subchapter
**
**    Purpose :   This routine prompts the expert for the new title and
**                enters it into the list of subchapter titles replacing
**                the old one.
**
**    Called by:  edit in this same file
**
**    Calls   :
**
*/

edit_subchapter (
   int             screen_num,
   char            ch,
   struct chap     *chap_ptr
   )

   {
   /*
   ** define a temporary pointer
   */
   struct subchap *temp_ptr;

   /*
   ** define a variable to be the numeric value of the input - ch, and define
   ** a for loop counter
   */
   int             num,i;

   /*
   ** define a variable for the expert's responses
   */
   char            answer;

   temp_ptr = chap_ptr->sublist;
   num = ch - '0';
   num = num + (screen_num * MAX_TITLE_PER_SCREEN);
   for (i = 1; i < num; i++)
      temp_ptr = temp_ptr->next;


   /*
   ** ask the expert if he wishes to delete the title
   */
   printf ("\n\nDo you wish to delete the title (Y or N)\n");
   do
     {
     answer = getchar();
     while (getchar() != '\n');
     }
     while (!((answer == 'Y')||(answer =='y')||
```

```
        (answer =='N')||(answer =='n')));
 if ((answer == 'Y') || (answer == 'y'))
   {
   if ((temp_ptr->prior == NULL) && (temp_ptr->next == NULL))
     {
     /*
     ** only element in list of subchapters
     */
     if (latest_subchap ==  temp_ptr)
        latest_subchap = NULL;
     chap_ptr->sublist = NULL;
     free (temp_ptr);
     return;
     }
   if ((temp_ptr->prior == NULL) && (temp_ptr->next != NULL))
     {
     /*
     ** first chapter in list of subchapters
     */
     chap_ptr->sublist = temp_ptr->next;
     temp_ptr->next->prior = NULL;
     free (temp_ptr);
     return;
     }
   if (temp_ptr->next == NULL)
     {
     /*
     ** last entry in list of subchapters
     */
     if (latest_subchap == temp_ptr)
        latest_subchap = temp_ptr->prior;
     temp_ptr->prior->next = NULL;
     free (temp_ptr);
     return;
     }
   /*
   ** middle entry of list of subchapters
   */
   temp_ptr->next->prior = temp_ptr->prior;
   temp_ptr->prior->next = temp_ptr->next;
   free (temp_ptr);
   return;
   }

/*
** print the title and prompt the expert to type in a new one
*/
printf ("\n\n\n%s\n", temp_ptr->concept);
printf ("\n\n Enter the new title \n");
i = 0;
while ((answer = getchar ()) != '\n')
  {
  if (i < MAX_CONCEPT_LENGTH - 1)
    {
    temp_ptr->concept[i] = answer;
    i++;
    }
```

```
      else
        printf ("\n YOUR TITLE HAS BEEN TRUNCATED \n");
      }
    temp_ptr->concept[i] = '\0';
   }

/*
**    Routine :   add_subchap
**
**    Purpose :   This routine prompts the expert for the new title and
**                enters it into the list of subchapter titles
**
**    Called by:  edit_chap in this same file
**
**    Calls   :
**
*/
add_subchap(
   struct chap       *chap_ptr
   )

   {
   /*
   ** define a pointer to a new subchapter node and a pointer to list of
   ** subchapter titles
   */
   struct subchap    *new_subchap;
   struct subchap    *sub_chaps;

   /*
   ** define the index and input character to title
   */
   int               i;
   char              ch;

   /*
   ** allocate memory for the new title
   */
   new_subchap = (struct subchap *) malloc (sizeof (struct subchap));
   new_subchap->next = NULL;
   new_subchap->prior = NULL;

   /*
   ** read in the title
   */
   printf ("Enter the new subchapter title\n");
   i = 0;
   while ((ch = getchar ()) != '\n')
     {
     if (i < MAX_CONCEPT_LENGTH)
       {
       new_subchap->concept[i] = ch;
       i++;
       }
     else
       printf ("\n\n YOUR TITLE HAS BEEN TRUNCATED\n");
     }
```

```
new_subchap->concept [i] = '\0';

/*
** find the last subchapter
*/
sub_chaps = chap_ptr->sublist;
if (sub_chaps == NULL)
   {
   if (latest_chap == chap_ptr)
     latest_subchap = new_subchap;
   chap_ptr->sublist = new_subchap;
   }
else
   {
   while (sub_chaps->next != NULL)
     sub_chaps = sub_chaps->next;
   if (latest_subchap == sub_chaps)
      latest_subchap = new_subchap;
   sub_chaps->next = new_subchap;
   new_subchap->prior = sub_chaps;
   }
}
```

File: Write_To_File.c

```
/*****************************************************************************
**
**    Program  :  Automated KA Tool for Indentification of Generic Tasks
**
**    Author   :  Arlene J Buck
**
**    Contents :  This file contains the routine that writes the titles of
**                chapters and subchapters to a file.
**
*****************************************************************************/

#include "thesis.h"

/*
**    Routine :   write_titles_to_file
**
**    Purpose :   Write the titles to a file
**
**    Called by:  main program in thesis.c
**
**    Calls   :
**
*/

write_titles_to_file ()
    {
    /*
    ** define a file for the titles and a temporary pointer to a chapter
    ** and subchapter title
    */
    char              *fp;
    struct chap       *chap_ptr;
    struct subchap    *subchap_ptr;

    /*
    ** define input variable ch and index counter i
    */
    char        ch;
    int         i;

    /*
    ** prompt the expert for a name for the file of titles
    */
    do
      {
      printf ("\n\n\nEnter a filename for the titles.\n");
      printf ("It cannot be more than 8 characters long \n");
      printf ("For example : mytitles\n");
      printf ("\n The titles will be stored in a file with the name you \n");
      printf ("just entered and an extension of '.dat'\n");
      i = 0;
      while ((ch = getchar()) != '\n')
        {
        if (i < MAX_FILENAME_LENGTH - 1)
          {
```

```
        filename [i] = ch;
        i++;
        }
      }
   filename [i] = '.';
   filename [i+1] = 'd';
   filename [i+2] = 'a';
   filename [i+3] = 't';
   filename [i+4] = '\0';

   /*
   ** open the file for writing
   */
   fp = fopen (filename,"w");
   }
   while (fp == NULL);

/*
** set the chapter pointer
*/
chap_ptr = head;

/*
** use loops to write each title to the file and free the structures
** that stored the titles as you go
*/
while (1)
   {
   /*
   ** make sure there are chapter titles
   */
   if (chap_ptr != NULL)
      {
      /*
      ** write it to the file
      */
      fputs (chap_ptr->concept, fp);
      fputs ("\n",fp);

      /*
      ** get the subchapter titles
      */
      subchap_ptr = chap_ptr->sublist;
      while (1)
         {
         /*
         ** make sure there are subchapter titles
         */
         if (subchap_ptr != NULL)
            {
            /*
            ** write it to the file and see if the next title exists
            */
            fputs (subchap_ptr->concept,fp);
            fputs ("\n",fp);
            if (subchap_ptr->next == NULL)
               {
```

```
                /*
                ** last one in this list so free it and leave the loop
                */
                free (subchap_ptr);
                break;
                }

             /*
             ** at this point there are more subchapter titles so get them
             ** and free the last one
             */
             subchap_ptr = subchap_ptr->next;
             free (subchap_ptr->prior);
             }
          else
             /*
              ** there are no subchapter titles
             */
             break;
          }
       /*
       ** if this is the last one - free it and end this loop
       */
       if (chap_ptr->next == NULL)
          {
          free (chap_ptr);
          break;
          }
       /*
       ** at this point there are more chapter titles so get them
       ** and free the last one
       */
       chap_ptr = chap_ptr->next;
       free (chap_ptr->prior);
       }
    }
/*
** all titles are written so close the file
*/
fclose (fp);
}
```

File: Screen2.c

```
/*****************************************************************************
**
**     Program  :  Automated KA Tool for Indentification of Generic Tasks
**
**     Author   :  Arlene J Buck
**
**     Contents :  This file contains the routine that displays a screen to
**                 expert that descibes the ratings and how they are used.
**
*****************************************************************************/

/*
**     Routine :   screen2
**
**     Purpose :   Descibes the ratings and how they are used to the expert.
**
**     Called by:  main program in thesis.c
**
**     Calls   :
**
*/

screen2 ()
   {
   printf ("\n\n\n\n\n\n\n\n\n\n");
   printf ("The next part of the Automated KA Tool asks you to rate \n");
   printf ("your titles against other concepts\n");
   printf ("Each title will be displayed on the screen next to a concept\n");
   printf ("If they are NOT similar -      type 0\n");
   printf ("If they are somewhat similar - type 1\n");
   printf ("If they are VERY similar -     type 2\n");
   printf ("The concept will not be directly related to your title but\n");
   printf ("will be a description of a thought process. If your title is\n");
   printf ("involved in the thought process described by the concept then\n");
   printf ("the concept and your title are similar\n");
   printf ("\n\nPRESS RETURN WHEN READY\n");
   getchar ();
   printf ("For example my book is HOW TO DRIVE\n");
   printf ("TITLE : driving safely in the snow \n");
   printf ("CONCEPT : This object can be labeled as an element of some\n");
   printf ("          hierarchy\n");
   printf ("\nHow I would rate the title to the concept:\n");
   printf ("When I entered driving in snow, my thought process was that\n");
   printf ("driving in snow is part of driving. It is similar to, yet \n");
   printf ("different from driving in rain and driving on dry roads which\n");
   printf ("are other chapters in my book. Thus, driving in snow can be \n");
   printf ("considered part of the driving hierarchy - and I would type 2\n");
   printf ("because the title and concept are similar\n");
   printf ("       WHEN READY TO CONTINUE - PRESS RETURN\n");
   getchar ();
   }
```

File: Get_Ratings.c

```
/*****************************************************************************
**
**    Program  :  Automated KA Tool for Indentification of Generic Tasks
**
**    Author   :  Arlene J Buck
**
**    Contents :  This file contains the routine that prompts the expert
**                rate each title against each generic task concept.
**
*****************************************************************************/

#include "thesis.h"

/*
**    Routine :   get_ratings
**
**    Purpose :   displays the title and a concept and prompts the expert
**                for the rating which is written to a file.
**
**    Called by:  main program in thesis.c
**
**    Calls   :
**
*/

get_ratings ()
    {
    /*
    ** define a file for the generic tasks and the titles
    ** and the ratings
    */
    char              *fpTasks;
    char              *fpTitles;
    char              *fpRates;

    /*
    ** define the number of observations to be each time a title is rated
    ** against a set of generic task characteristics. Define a counter to
    ** place 6 task ratings to one observation.
    */
    int               conceptNO;
    int               counter;

    /*
    ** define a temporary title
    ** and a task lines, 3 help lines, a rating and an input character
    */
    char              title[MAX_CONCEPT_LENGTH];
    char              task[MAX_CONCEPT_LENGTH];
    char              help[MAX_CONCEPT_LENGTH];
    char              help2[MAX_CONCEPT_LENGTH];
    char              help3[MAX_CONCEPT_LENGTH];
    int               rating;
    char              ch;
```

```
/*
** open the ratings file for writing and the title file for reading
*/
fpRates = fopen ("rates.dat","w");
fpTitles = fopen (filename,"r");

conceptNO = 0;

/*
** for every title read in the title and compare it to every task
** and store each rating
*/
while (fgets (title, MAX_CONCEPT_LENGTH, fpTitles) != NULL)
   {

   /*
   ** open the titles file for reading
   */
   fpTasks = fopen ("task.dat","r");
   counter = 1;

   while (fgets (task,MAX_CONCEPT_LENGTH,fpTasks) != NULL)
      {
      /*
      ** get the three help lines in case
      ** they are needed
      */
      fgets (help,  MAX_CONCEPT_LENGTH, fpTasks);
      fgets (help2, MAX_CONCEPT_LENGTH, fpTasks);
      fgets (help3, MAX_CONCEPT_LENGTH, fpTasks);

      /*
      ** print the title, the task concept and prompt for the rating
      */
      printf ("\n\nTITLE :\n%s\n\n",title);
      printf ("CONCEPT:\n%s",task);
      printf (" Are they :\n");
      printf ("  not similar      - 0\n");
      printf ("  somewhat similar - 1\n");
      printf ("  very similar     - 2\n");
      printf (" Enter 0,1,2 or H for help then RETURN\n");

      /*
      ** get the rating
      */
      do
        {
        ch = getchar ();
        while (getchar () != '\n');
        }
        while (!((ch == '0')||(ch == '1')||(ch == '2') || (ch == 'H') ||
                 (ch == 'h')));

      /*
      ** if help was requested print out the help lines
      */
      if ((ch == 'H') || (ch == 'h'))
```

```
        {
        printf ("\n\n%s",help);
        printf ("%s",help2);
        printf ("%s",help3);
        printf ("\nEnter the rating 0, 1, or 2\n");

        /*
        ** get the rating
        */
        do
          {
          ch = getchar ();
          while (getchar () != '\n');
          }
          while (!((ch == '0')||(ch == '1')||(ch == '2')));
        }

      /*
      ** write the rating to the file
      */
      rating = ch - '0';
      fprintf (fpRates, "%d ",rating);
      if (counter == 6)
        {

        /*
        ** now the 6 task ratings are in the file add the concept number
        */
        fprintf (fpRates, "%d\n", conceptNO);
        conceptNO++;
        counter = 0;
        }
      counter++;
      }

   fclose (fpTasks);
   }
fclose (fpRates);
fclose (fpTitles);
}
```

File: Task.Dat

This object can be labeled as an element of some hierarchy of objects.
If the object (represented by this title) can be considered as part of some hierarchy that relates this object to other objects (represented by other titles), then type 2.
This object can be labeled as a component in some subsystem of objects.
If the object (represented by this title) mirrors a component of a actual system or subsystem that relates this object to other objects (represented by other titles), then type 2.
Characteristics of this object are shared by other objects.
If the object (represented by this title) and other objects (represented by other titles) have some of the same characteristics,
then type 2.
This object can be labeled as an element of some overall plan.
If the object (represented by this title) can be considered as part of some plan that includes other objects (represented by other titles),
then type 2.
This object provides partial evidence for a theory.
If the object (represented by this title) provides some evidence that together with evidence from other objects (represented by other titles) can prove or disprove a theory, then type 2.
This object provides a partial explanation of a truth.
If the object (represented by this title) provides some explanation that together with explanations by other objects (represented by other titles) can provide a complete explanation, then type 2.
If this object is established or rejected, other objects can be also.
If the object (represented by this title) being established or rejected implies that other objects (represented by other titles) can be established or rejected, type 2.
This object can change its state.
If the object (represented by this title) changing its state implies that other objects (represented by other titles) can have thier states changed, type 2.
This object can inherit characteristics.
If the object (represented by this title) is missing, and its characteristics can be sumised from other objects (represented by other titles),
type 2.
This object can be chosen as part of a plan.
If the object (represented by this title) being chosen implies that other objects (represented by other titles) can be chosen as part of a plan, type 2.
This object can be matched.
If the object (represented by this title) is matched to some other object(represented by another title)
rejected, type 2.
This object can provide justification.
If the object (represented by this title) justifies other objects (represented by other titles) can be established or
rejected, type 2.
If this object is true, the problem can be classified further.
If the object (represented by this title) helps the reader classify the problem further and that the classification is a goal of your book, type 2.
If this object is true, the problem can be traced further.
If the object (represented by this title) helps the reader trace the problem further and that the tracing of the problem is a goal of your book, type 2.

If this object is true, the problem can be described further.
If the object (represented by this title) helps the reader describe the problem further and that the describing of the problem is a goal of your book, type 2.
If this object is true, the problem can be designed further.
If the object (represented by this title) helps the reader choose a partial design and that the design is a goal of your book,
type 2.
If this object is true, the problem can be evaluated further.
If the object (represented by this title) helps the reader evaluate the evidence further and that the evaluation is a goal of your book, type 2.
If this object is true, the problem can be understood further.
If the object (represented by this title) helps the reader understand the problem further and that the understanding of the problem is a goal of your book, type 2.

File: Thesis.sas

```
options nodate linesize=80;
data tasks;
     filename rates 'rates.dat';
     infile rates;
     input task1 task2 task3 task4 task5 task6 con_no @@;
run;
data seeds;
     filename inits 'seeds.dat';
     infile inits;
     input task1 task2 task3 task4 task5 task6;
run;
proc print;
     title 'generic Task Indentification';
     run;
proc fastclus data=tasks seed=seeds maxc=6 maxiter=10 out=clus list;
     var task1 task2 task3 task4 task5 task6;
     id con_no;
     run;
proc freq;
proc candisc uni out=can;
     class cluster;
     var task1 task2 task3 task4 task5 task6;
     title2 'Canonical discriminant analysis of task clusters';
     run;
proc plot;
     plot can1*can2=cluster;
     title2 'Plot of Canonical variables identified by cluster';
     run;
endsas;
```

File: Seeds.dat

```
2 0 0 0 0 0

0 2 0 0 0 0

0 0 2 0 0 0

0 0 0 2 0 0

0 0 0 0 2 0

0 0 0 0 0 2
```

Appendix D
Examples

D.1 Diagnostic Example

File: Diagnostic.dat

```
backplane
pin assignments
master communication channel
set initialization mode
read initialization mode
hardware diagnostics
slave communication channel
comm clock
read write memory
power up diagnostics
read only memory
power up checksum test
input ports
input loop back test
ttl inputs
sensor inputs
output ports
output loop back test
ac driver circuit
darlington driver
led driver
real time clock
power up diagnostics
bypass circuit
voltage diagnostics
logic voltage test
driver voltage test
diagnostics voltage test
power normal reset
watchdog timer circuit
keeping it alive
recovery from timeout
hardware indicator circuit
softload
boot promv
softload directory
nvm
```

File: Diagnostic.rat

```
2 2 2 2 0 1 0
0 0 1 2 0 0 1
1 2 1 1 0 2 2
2 2 2 2 0 0 3
0 0 0 1 0 0 4
1 1 1 0 0 1 5
2 2 2 0 0 0 6
0 0 1 0 0 0 7
1 1 1 1 0 1 8
2 2 1 2 0 0 9
0 2 0 2 0 0 10
2 0 0 1 1 1 11
2 2 2 2 0 0 12
0 2 0 2 0 0 13
1 1 1 0 0 1 14
2 2 1 2 0 0 15
0 0 0 2 0 1 16
2 2 2 0 1 2 17
2 2 2 2 0 0 18
0 2 1 1 0 1 19
1 1 1 0 0 1 20
2 2 2 2 0 0 21
0 0 0 0 0 0 22
1 0 0 0 1 1 23
2 2 2 2 0 0 24
0 0 1 2 0 2 25
0 0 1 0 0 0 26
2 2 2 2 0 0 27
0 0 0 1 0 1 28
1 2 1 0 0 0 29
2 2 2 2 0 0 30
0 0 0 2 1 1 31
0 0 0 0 0 1 32
2 2 2 2 1 0 33
0 0 0 2 0 1 34
0 1 0 0 0 0 35
2 2 2 2 2 0 36
0 0 0 2 0 2 37
0 0 0 0 0 0 38
2 2 2 2 1 1 39
0 0 0 2 0 2 40
2 1 1 0 1 0 41
2 2 2 2 0 0 42
0 0 0 2 0 2 43
1 1 1 0 0 0 44
2 2 2 2 0 0 45
0 0 0 2 1 2 46
2 2 1 1 0 1 47
2 2 2 2 0 1 48
2 2 0 2 0 2 49
2 0 2 1 0 0 50
2 2 2 2 1 0 51
2 0 1 2 0 2 52
1 2 0 0 0 0 53
2 2 2 2 0 0 54
```

0	2	0	2	0	2	55
1	1	0	0	0	1	56
2	2	2	2	0	0	57
1	0	0	1	0	1	58
1	1	1	0	0	1	59
2	2	2	2	0	0	60
1	1	0	1	0	1	61
0	1	0	0	1	1	62
2	2	2	2	0	0	63
0	1	0	2	0	1	64
0	0	0	0	0	0	65
2	2	2	2	0	1	66
0	0	0	2	0	2	67
0	1	1	0	1	0	68
2	2	2	2	0	0	69
0	2	0	1	0	2	70
0	0	0	0	0	0	71
2	2	2	2	0	2	72
2	2	0	2	0	1	73
2	2	0	0	1	0	74
2	2	2	2	1	1	75
1	2	0	2	1	2	76
2	0	2	0	0	0	77
2	2	2	2	2	2	78
2	0	0	2	0	2	79
2	2	0	0	0	0	80
2	2	2	2	0	0	81
2	2	0	2	0	2	82
2	2	2	0	1	0	83
2	2	2	2	2	2	84
2	2	0	2	2	2	85
2	2	2	0	1	0	86
2	2	2	2	2	0	87
1	2	0	2	1	2	88
2	1	0	0	0	1	89
2	2	2	2	0	0	90
0	0	0	2	0	1	91
0	0	0	1	0	0	92
2	2	2	2	2	0	93
1	0	0	1	0	2	94
0	0	0	0	0	0	95
2	2	2	2	2	2	96
2	2	0	2	2	2	97
2	2	2	0	2	2	98
2	2	2	2	0	0	99
0	0	0	2	0	2	100
0	0	0	0	0	0	101
2	2	2	2	0	0	102
1	0	1	2	0	2	103
0	0	0	1	0	0	104
2	2	2	2	0	0	105
2	0	0	2	0	2	106
0	0	1	0	0	0	107
2	2	2	2	0	0	108
1	0	0	2	0	2	109
0	0	0	0	0	1	110

File: Diagnostic.lis

OBS	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	2	0	0	0
4	0	0	0	2	0	0
5	0	0	0	0	2	0
6	0	0	0	0	0	2

FASTCLUS PROCEDURE

REPLACE=FULL RADIUS=0 MAXCLUSTERS=6 MAXITER=10 CONVERGE=0.02

INITIAL SEEDS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.00000	2.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	2.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000	2.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000

MINIMUM DISTANCE BETWEEN SEEDS = 2.828427

ITERATION	CHANGE IN CLUSTER SEEDS					
	1	2	3	4	5	6
1	2.46613	1.49753	1	1.2999	2.95581	0.816497
2	0.777768	0.278459	0.730297	0.460302	0.900619	0.377964
3	0.287714	0.549004	0	0	0.455865	0.176777
4	0.110696	0.467905	0.101015	0.105085	0.503644	0
5	0.157324	0	0	0.0649886	0.372124	0.249227
6	0	0	0	0	0	0

CLUSTER LISTING

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
1	0	1	0.78025
2	1	4	1.87785
3	2	2	1.31671
4	3	1	0.57534
5	4	3	0.85117
6	5	5	0.94340
7	6	5	1.30000
8	7	3	0.76265
9	8	5	1.26095
10	9	1	1.10449
11	10	2	1.91228
12	11	6	1.62542
13	12	1	0.57534
14	13	2	1.91228
15	14	5	0.94340
16	15	1	1.10449
17	16	4	0.82717
18	17	5	1.94679
19	18	1	0.57534
20	19	2	1.67650
21	20	5	0.94340
22	21	1	0.57534
23	22	3	0.39123
24	23	6	0.92962
25	24	1	0.57534
26	25	4	1.00000
27	26	3	0.76265
28	27	1	0.57534
29	28	6	0.98758
30	29	5	1.09087
31	30	1	0.57534
32	31	4	1.21395
33	32	6	0.92962
34	33	1	0.66493
35	34	4	0.82717
36	35	3	0.93131
37	36	1	1.59789
38	37	4	0.64889
39	38	3	0.39123
40	39	1	0.84847
41	40	4	0.64889
42	41	5	1.04403
43	42	1	0.57534
44	43	4	0.64889
45	44	5	0.99499
46	45	1	0.57534
47	46	4	1.10024
48	47	5	1.26095
49	48	1	0.78025
50	49	2	1.22595
51	50	5	1.99750
52	51	1	0.66493

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
53	52	4	1.76218
54	53	5	1.57797
55	54	1	0.57534
56	55	2	1.22595
57	56	6	0.92962
58	57	1	0.57534
59	58	6	0.80123
60	59	5	0.94340
61	60	1	0.57534
62	61	6	0.98758
63	62	6	1.23728
64	63	1	0.57534
65	64	4	1.25656
66	65	3	0.39123
67	66	1	0.78025
68	67	4	0.64889
69	68	3	1.46733
70	69	1	0.57534
71	70	2	1.42878
72	71	3	0.39123
73	72	1	1.69899
74	73	2	1.25693
75	74	5	1.60935
76	75	1	0.84847
77	76	2	0.76150
78	77	5	1.81384
79	78	1	2.26026
80	79	4	1.58944
81	80	5	1.51327
82	81	1	0.57534
83	82	2	1.22595
84	83	5	1.41067
85	84	1	2.26026
86	85	2	1.91228
87	86	5	1.41067
88	87	1	1.59789
89	88	2	0.76150
90	89	5	1.41067
91	90	1	0.57534
92	91	4	0.82717
93	92	3	0.85117
94	93	1	1.59789
95	94	4	1.16980
96	95	3	0.39123
97	96	1	2.26026
98	97	2	1.91228
99	98	5	2.46779
100	99	1	0.57534
101	100	4	0.64889
102	101	3	0.39123
103	102	1	0.57534
104	103	4	1.02598
105	104	3	0.85117
106	105	1	0.57534
107	106	4	1.58944

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
108	107	3	0.76265
109	108	1	0.57534
110	109	4	0.68825
111	110	6	0.92962

CLUSTER SUMMARY

CLUSTER NUMBER	FREQUENCY	RMS STD DEVIATION	MAXIMUM DISTANCE FROM SEED TO OBSERVATION	NEAREST CLUSTER	CENTROID DISTANCE
1	36	0.4314	2.2603	5	2.1490
2	13	0.6287	1.9123	4	2.0591
3	14	0.3174	1.4673	6	1.2977
4	19	0.4714	1.8778	6	1.6941
5	20	0.6084	2.4678	6	1.9029
6	9	0.4615	1.6254	3	1.2977

STATISTICS FOR VARIABLES

VARIABLE	TOTAL STD	WITHIN STD	R-SQUARED	RSQ/(1-RSQ)
TASK1	0.8996314	0.5303803	0.6682261	2.0141011
TASK2	0.9196196	0.3580590	0.8552931	5.9105198
TASK3	0.9016320	0.4380284	0.7747097	3.4387166
TASK4	0.8961653	0.3144826	0.8824525	7.5071957
TASK5	0.6176492	0.6124712	0.0613922	0.0654077
TASK6	0.8335708	0.6087253	0.4909572	0.9644715
OVER-ALL	0.8512168	0.4909082	0.6825197	2.1498015

PSEUDO F STATISTIC = 45.15
APPROXIMATE EXPECTED OVER-ALL R-SQUARED = 0.51252
CUBIC CLUSTERING CRITERION = 15.894
WARNING: THE TWO ABOVE VALUES ARE INVALID FOR CORRELATED VARIABLES

CLUSTER MEANS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2.00000	2.00000	1.94444	2.00000	0.44444	0.36111
2	1.00000	2.00000	0.15385	1.76923	0.46154	1.53846
3	0.00000	0.14286	0.28571	0.21429	0.07143	0.00000
4	0.47368	0.05263	0.21053	1.94737	0.10526	1.63158
5	1.60000	1.40000	1.15000	0.15000	0.35000	0.55000
6	0.66667	0.33333	0.00000	0.44444	0.33333	1.00000

CLUSTER STANDARD DEVIATIONS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	0.000000	0.000000	0.232311	0.000000	0.772545	0.682549
2	0.912871	0.000000	0.375534	0.438529	0.776250	0.776250
3	0.000000	0.363137	0.468807	0.425815	0.267261	0.000000
4	0.772328	0.229416	0.418854	0.229416	0.315302	0.597265
5	0.502625	0.680557	0.745160	0.366348	0.587143	0.686333
6	0.707107	0.500000	0.000000	0.527046	0.500000	0.000000

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

111 OBSERVATIONS — 110 DF TOTAL
6 VARIABLES — 105 DF WITHIN CLASSES
6 CLASSES — 5 DF BETWEEN CLASSES

	CANONICAL CORRELATION	ADJUSTED CANONICAL CORRELATION	APPROX STANDARD ERROR	SQUARED CANONICAL CORRELATION
1	0.967241	0.964309	0.006145	0.935555
2	0.924365	0.920406	0.013878	0.854451
3	0.761740	0.751056	0.040022	0.580248
4	0.501197	0.487105	0.071395	0.251198
5	0.144399	0.119678	0.093358	0.020851

EIGENVALUES OF INV(E)*H
= CANRSQ/(1-CANRSQ)

	EIGENVALUE	DIFFERENCE	PROPORTION	CUMULATIVE
1	14.5172	8.6467	0.6561	0.6561
2	5.8705	4.4882	0.2653	0.9214
3	1.3824	1.0469	0.0625	0.9839
4	0.3355	0.3142	0.0152	0.9990
5	0.0213	.	0.0010	1.0000

TESTS OF H0: THE CANONICAL CORRELATION IN THE CURRENT ROW AND ALL THAT FOLLOW ARE ZERO

	LIKELIHOOD RATIO	APPROX F	NUM DF	DEN DF	PR > F
1	0.00288671	44.4101	30	402	0.0
2	0.04479378	26.0476	20	335.929	0.0
3	0.30775752	12.6328	12	270.158	0.0001
4	0.73318862	5.7633	6	206	0.0001
5	0.97914879	1.1074	2	104	0.334

MULTIVARIATE TEST STATISTICS AND F APPROXIMATIONS

S=5 M=0 N=49

STATISTIC	VALUE	F	NUM DF	DEN DF	PR > F
WILKS' LAMBDA	0.002886714	44.410	30	402	0.0
PILLAI'S TRACE	2.642304	19.426	30	520	0.0
HOTELLING-LAWLEY TRACE	22.12688	72.576	30	492	0.0
ROY'S GREATEST ROOT	14.51722	251.632	6	104	0.0

NOTE: F STATISTIC FOR ROY'S GREATEST ROOT IS AN UPPER BOUND

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

TOTAL CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.7097	-0.4076	-0.0167	0.4573	-0.3338
TASK2	0.8214	-0.3878	0.3979	-0.1212	0.0131
TASK3	0.7182	-0.4539	-0.4380	0.1258	0.1903
TASK4	0.7352	0.6567	-0.1174	-0.0046	-0.1055
TASK5	0.1971	-0.1001	0.1366	0.0374	-0.5045
TASK6	-0.0742	0.5789	0.4426	0.5829	0.1413

BETWEEN CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.8398	-0.4609	-0.0156	0.2804	-0.0590
TASK2	0.8591	-0.3876	0.3277	-0.0657	0.0020
TASK3	0.7893	-0.4767	-0.3790	0.0716	0.0312
TASK4	0.7570	0.6462	-0.0952	-0.0024	-0.0162
TASK5	0.7694	-0.3734	0.4201	0.0757	-0.2940
TASK6	-0.1024	0.7637	0.4812	0.4170	0.0291

WITHIN CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.3128	-0.2700	-0.0188	0.6870	-0.5734
TASK2	0.5482	-0.3889	0.6776	-0.2756	0.0341
TASK3	0.3841	-0.3649	-0.5978	0.2293	0.3967
TASK4	0.5444	0.7308	-0.2218	-0.0116	-0.3045
TASK5	0.0516	-0.0394	0.0914	0.0334	-0.5152
TASK6	-0.0264	0.3096	0.4019	0.7070	0.1960

STANDARDIZED CANONICAL COEFFICIENTS

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.2622	-0.7540	-0.0500	1.0723	-0.9748
TASK2	1.8223	-0.6861	1.6267	-0.6752	0.4586
TASK3	1.0450	-0.5689	-1.2145	0.2799	1.0226
TASK4	1.9623	2.1124	-0.4911	-0.3857	-0.3259
TASK5	-0.1501	-0.0104	-0.1002	-0.2858	-0.6562
TASK6	-0.0273	0.5888	0.6420	0.9103	0.6790

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

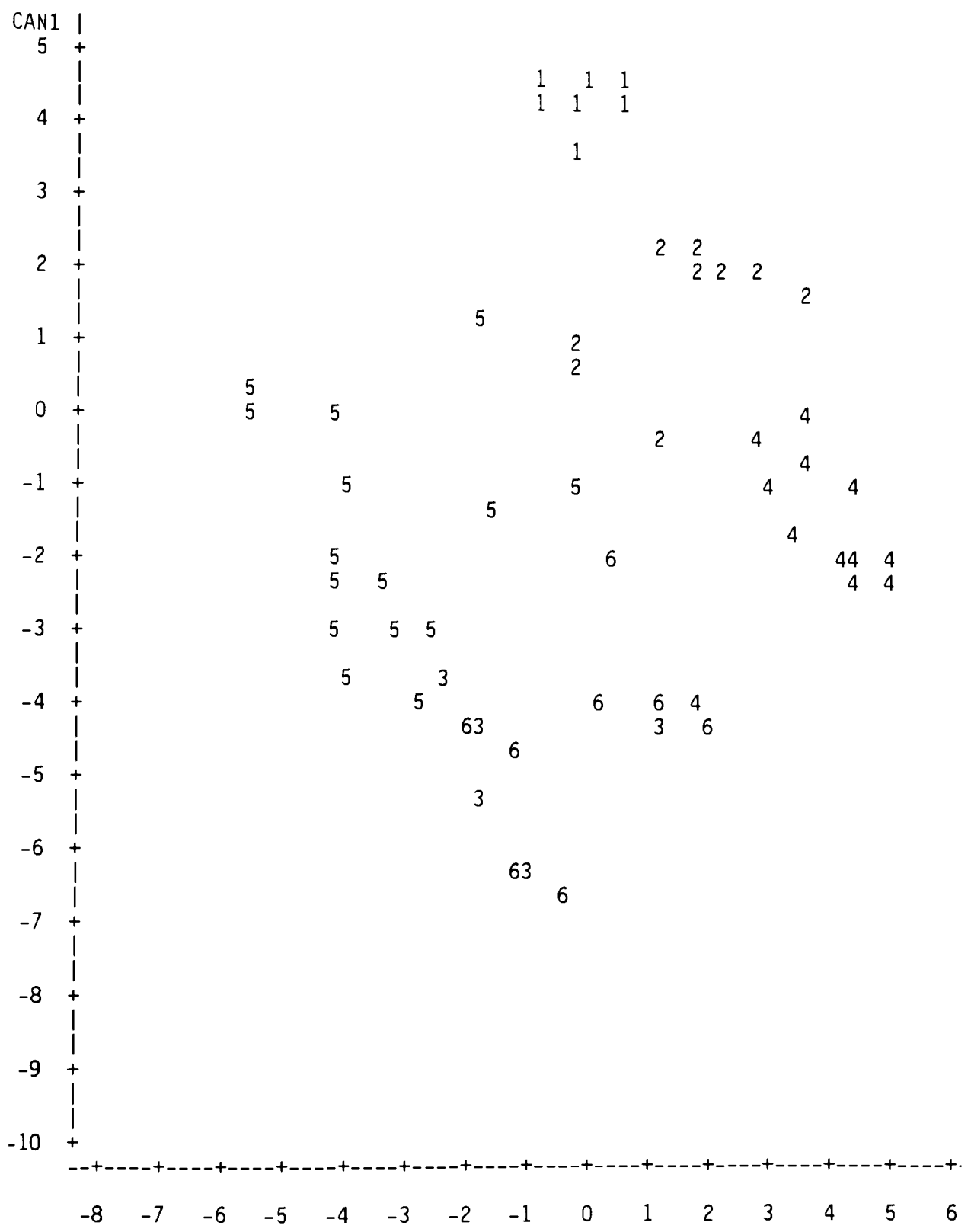
RAW CANONICAL COEFFICIENTS

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.291474935	-0.838115636	-0.055565131	1.191897636	-1.083586495
TASK2	1.981567845	-0.746099164	1.768851642	-0.734178361	0.498676380
TASK3	1.158996047	-0.630956086	-1.346992464	0.310414883	1.134189899
TASK4	2.189612294	2.357204094	-0.548003178	-0.430430376	-0.363654121
TASK5	-0.242941195	-0.016783762	-0.162174311	-0.462642731	-1.062491864
TASK6	-0.032730845	0.706396896	0.770127217	1.092081870	0.814525649

CLASS MEANS ON CANONICAL VARIABLES

CLUSTER	CAN1	CAN2	CAN3	CAN4	CAN5
1	4.5808	-0.5086	-0.7007	-0.1154	-0.0429
2	1.6661	1.7467	2.7972	-0.4859	0.0346
3	-5.4122	-0.8583	-0.8793	-1.1038	0.0684
4	-1.8069	4.0966	-0.6626	0.5239	0.0877
5	-1.6795	-3.4503	0.5049	0.6480	0.1173
6	-4.7640	-0.1344	0.4070	0.3342	-0.4305

Plot of Canonical variables identified by cluster



D.2 Plan Example

File: Plan.dat

```
Economic Environment
The Federal Reserve System
Economic Indicators
The Business Cycle
The Time Value of Money
Personal Financial Planning
The Process
Financial Statements
Regulation of Planners
Risk Management
Universal Life
Variable Life
Health, Property, and Liability Insurance
Medical Expanse Insurance
Homeowner's Insurance
Nursing Home Insurance
Principles of Life Insurance
Investment Planning
Risk Tolerance
Interpreting Statistical Data
Fixed Income Investment
Equity Investments
Tax Planning
Personal Income Tax Planning
Forms of Business Ownership
Business Taxation
Retirement Planning
Retirement Planning Personal
Qualified Pension and Profit Sharing Plans
Individual Retirement Accounts
Tax Sheltered Accounts
Estate Planning
Estate Planning Principles
Life Insurance Ownership and Beneficiaries
United Transfer Tax System
Gross Estate Calculations
Wills, Intestacy
Trusts
Ownership of Property
Charitable Transfers
Property Transfers
```

File: Plan.rat

```
1 2 0 2 2 1 0
0 2 2 2 2 2 1
1 0 1 0 0 0 2
0 1 2 2 0 0 3
0 2 2 2 2 2 4
0 0 0 0 1 1 5
2 2 2 1 2 1 6
2 2 2 2 2 2 7
2 2 2 0 2 2 8
2 2 2 1 2 2 9
1 2 2 2 2 2 10
0 0 2 2 2 2 11
2 0 2 2 2 2 12
2 1 0 2 2 2 13
0 2 0 2 2 2 14
1 2 2 2 2 0 15
1 2 2 2 2 2 16
0 2 2 2 2 2 17
0 0 0 0 2 2 18
2 1 1 1 2 2 19
2 2 2 2 1 2 20
2 2 2 2 0 1 21
0 0 0 0 2 2 22
0 0 0 1 0 0 23
1 2 2 0 0 0 24
2 0 1 0 2 2 25
2 2 2 2 2 2 26
2 2 2 2 2 2 27
2 2 2 2 2 2 28
2 2 2 2 2 2 29
0 2 1 1 1 2 30
2 2 0 2 1 2 31
2 2 0 2 2 2 32
0 2 2 2 1 0 33
2 2 2 2 2 2 34
0 0 0 2 2 2 35
0 2 2 2 2 2 36
2 2 2 2 2 2 37
2 1 1 1 1 2 38
1 2 2 2 1 1 39
2 1 2 2 1 1 40
1 0 0 2 0 2 41
0 2 2 2 1 0 42
2 2 2 2 2 2 43
1 2 0 2 0 2 44
0 2 2 2 0 0 45
2 2 2 2 2 2 46
2 0 0 2 0 2 47
2 2 2 2 0 1 48
1 2 2 2 2 2 49
2 0 1 2 2 2 50
2 0 2 2 2 1 51
2 2 2 2 2 2 52
2 2 2 2 2 2 53
0 2 2 2 2 0 54
```

0	1	2	2	0	2	55
2	2	0	0	2	2	56
0	2	2	0	2	0	57
0	0	0	2	2	2	58
0	0	2	2	0	2	59
1	2	2	2	0	0	60
1	0	0	2	2	2	61
0	0	0	0	0	0	62
0	1	2	2	2	0	63
2	2	2	2	2	2	64
0	2	2	2	2	2	65
1	2	2	2	0	0	66
1	1	2	2	2	2	67
1	1	1	2	2	2	68
2	2	2	2	2	2	69
0	2	2	2	2	1	70
2	0	2	2	2	0	71
0	2	2	2	0	0	72
0	0	2	2	0	0	73
0	0	0	0	0	0	74
2	0	2	2	1	1	75
2	2	2	1	1	1	76
1	2	2	2	2	2	77
0	2	2	2	0	0	78
2	2	2	2	1	2	79
2	2	2	2	2	2	80
1	2	2	2	0	2	81
2	2	2	2	2	2	82
2	2	2	2	0	0	83
2	1	1	1	0	1	84
1	2	2	2	2	2	85
2	1	1	2	2	2	86
2	2	2	2	0	0	87
0	2	2	2	1	2	88
2	1	1	2	0	2	89
2	2	2	2	2	2	90
2	0	2	1	0	2	91
0	0	0	0	0	0	92
2	0	2	2	2	0	93
2	2	2	2	1	2	94
2	2	1	2	2	2	95
2	2	2	2	2	2	96
2	2	2	2	1	1	97
2	2	1	1	1	1	98
1	0	2	2	2	2	99
0	2	0	2	2	2	100
0	0	2	2	1	1	101
0	2	0	1	0	1	102
2	0	1	1	1	2	103
0	2	2	2	0	0	104
2	0	2	2	2	2	105
2	2	0	1	1	1	106
0	0	0	2	2	0	107
2	0	2	2	0	0	108
1	2	2	2	1	2	109
1	1	2	2	2	2	110
1	1	2	2	2	0	111

2	0	0	2	0	0	112
0	2	2	0	0	0	113
2	0	2	2	0	0	114
2	2	1	2	2	2	115
0	0	0	0	0	0	116
1	1	2	2	0	1	117
1	0	0	2	1	2	118
0	0	0	2	2	0	119
0	2	2	2	0	0	120
1	0	2	2	1	2	121
2	0	0	0	0	0	122

File: Plan.lis

OBS	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	2	0	0	0
4	0	0	0	2	0	0
5	0	0	0	0	2	0
6	0	0	0	0	0	2

FASTCLUS PROCEDURE

REPLACE=FULL RADIUS=0 MAXCLUSTERS=6 MAXITER=10 CONVERGE=0.02

INITIAL SEEDS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.00000	2.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	2.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000	2.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000

MINIMUM DISTANCE BETWEEN SEEDS = 2.828427

ITERATION	CHANGE IN CLUSTER SEEDS					
	1	2	3	4	5	6
1	3.10832	2.93675	2.67909	2.02454	1.69967	3.1194
2	0.486995	0.434191	0.527709	0.345831	1.61015	0.898544
3	0.206408	0.527164	0.452462	0.314024	0.131937	0.475419
4	0.131009	0.288402	0.285257	0	0	0.176933
5	0.067601	0	0.127104	0	0	0
6	0.114744	0	0.190311	0	0	0
7	0.051774	0	0.0799988	0	0	0
8	0.104911	0.0876707	0.187414	0.163889	0	0
9	0	0.131436	0.201114	0	0	0
10	0	0	0	0	0	0

CLUSTER LISTING

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
1	0	6	1.94773
2	1	6	0.67847
3	2	5	1.34536
4	3	2	1.14695
5	4	6	0.67847
6	5	5	0.78102
7	6	1	1.19932
8	7	1	0.59385
9	8	1	1.75941
10	9	1	0.85093
11	10	6	0.71270
12	11	6	1.87295
13	12	3	1.23768
14	13	4	1.63889
15	14	6	1.71362
16	15	2	1.65012
17	16	6	0.71270
18	17	6	0.67847
19	18	5	2.14709
20	19	1	1.32388
21	20	1	0.91562
22	21	2	1.67242
23	22	5	2.14709
24	23	5	1.18743
25	24	2	1.90047
26	25	3	2.22551
27	26	1	0.59385
28	27	1	0.59385
29	28	1	0.59385
30	29	1	0.59385
31	30	6	1.52232
32	31	1	1.79955
33	32	1	1.65911
34	33	2	0.93330
35	34	1	0.59385
36	35	4	1.36666
37	36	6	0.67847
38	37	1	0.59385
39	38	1	1.49612
40	39	2	1.00956
41	40	3	1.02867
42	41	4	1.33299
43	42	2	0.93330
44	43	1	0.59385
45	44	4	2.16471
46	45	2	0.91325
47	46	1	0.59385
48	47	4	1.72008
49	48	2	1.67242
50	49	6	0.71270
51	50	3	1.41617
52	51	3	1.07863

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
53	52	1	0.59385
54	53	1	0.59385
55	54	2	1.70531
56	55	2	1.99554
57	56	1	2.34425
58	57	2	2.42303
59	58	4	1.36666
60	59	3	2.16558
61	60	2	0.80550
62	61	4	1.02449
63	62	5	0.78102
64	63	2	1.84108
65	64	1	0.59385
66	65	6	0.67847
67	66	2	0.80550
68	67	6	1.01575
69	68	6	1.12687
70	69	1	0.59385
71	70	6	1.12687
72	71	3	1.67183
73	72	2	0.91325
74	73	2	1.94858
75	74	5	0.78102
76	75	3	0.61152
77	76	1	1.38711
78	77	6	0.71270
79	78	2	0.91325
80	79	1	0.91562
81	80	1	0.59385
82	81	2	1.82085
83	82	1	0.59385
84	83	2	1.56960
85	84	3	1.79334
86	85	6	0.71270
87	86	1	1.17526
88	87	2	1.56960
89	88	6	1.08379
90	89	3	1.76375
91	90	1	0.59385
92	91	3	1.55775
93	92	5	0.78102
94	93	3	1.67183
95	94	1	0.91562
96	95	1	0.74341
97	96	1	0.59385
98	97	1	1.24605
99	98	1	1.45742
100	99	3	1.37851
101	100	6	1.71362
102	101	3	1.76375
103	102	2	2.31356
104	103	3	1.29990
105	104	2	0.91325
106	105	3	1.23768
107	106	1	2.07944

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
108	107	4	1.91988
109	108	3	1.79334
110	109	6	1.10554
111	110	6	1.01575
112	111	2	1.79008
113	112	4	2.18560
114	113	2	1.94858
115	114	3	1.79334
116	115	1	0.74341
117	116	5	0.78102
118	117	2	1.20981
119	118	4	0.64282
120	119	4	1.91988
121	120	2	0.91325
122	121	3	1.05395
123	122	5	1.84662

CLUSTER SUMMARY

CLUSTER NUMBER	FREQUENCY	RMS STD DEVIATION	MAXIMUM DISTANCE FROM SEED TO OBSERVATION	NEAREST CLUSTER	CENTROID DISTANCE
1	35	0.4519	2.3443	6	1.5595
2	27	0.6379	2.4230	6	2.1248
3	19	0.6519	2.2255	4	1.9300
4	11	0.7006	2.1856	3	1.9300
5	10	0.5916	2.1471	4	2.3327
6	21	0.4773	1.9477	1	1.5595

STATISTICS FOR VARIABLES

VARIABLE	TOTAL STD	WITHIN STD	R-SQUARED	RSQ/(1-RSQ)
TASK1	0.8841308	0.5763944	0.5924016	1.4533954
TASK2	0.8990050	0.4379995	0.7723597	3.3928960
TASK3	0.8403873	0.5530679	0.5846406	1.4075538
TASK4	0.6893064	0.5085982	0.4779032	0.9153537
TASK5	0.8785355	0.6913614	0.4060941	0.6837685
TASK6	0.8771691	0.6092434	0.5373622	1.1615181
OVER-ALL	0.8477953	0.5682999	0.5690776	1.3206034

PSEUDO F STATISTIC = 30.90
APPROXIMATE EXPECTED OVER-ALL R-SQUARED = 0.49709
CUBIC CLUSTERING CRITERION = 6.252
WARNING: THE TWO ABOVE VALUES ARE INVALID FOR CORRELATED VARIABLES

CLUSTER MEANS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2.00000	1.91429	1.60000	1.68571	1.74286	1.85714
2	0.59259	1.74074	1.92593	1.74074	0.48148	0.33333
3	1.68421	0.15789	1.73684	1.73684	1.10526	1.31579
4	0.90909	0.27273	0.00000	2.00000	1.18182	1.45455
5	0.30000	0.00000	0.10000	0.10000	0.50000	0.50000
6	0.47619	1.76190	1.61905	1.95238	1.85714	1.90476

CLUSTER STANDARD DEVIATIONS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	0.000000	0.284029	0.694516	0.582663	0.443440	0.355036
2	0.747265	0.525693	0.384900	0.655896	0.802418	0.620174
3	0.671038	0.374634	0.452414	0.561951	0.875261	0.820070
4	0.831209	0.646670	0.000000	0.000000	0.981650	0.934199
5	0.674949	0.000000	0.316228	0.316228	0.849837	0.849837
6	0.511766	0.538958	0.740013	0.218218	0.358569	0.300793

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

123 OBSERVATIONS — 122 DF TOTAL
6 VARIABLES — 117 DF WITHIN CLASSES
6 CLASSES — 5 DF BETWEEN CLASSES

	CANONICAL CORRELATION	ADJUSTED CANONICAL CORRELATION	APPROX STANDARD ERROR	SQUARED CANONICAL CORRELATION
1	0.907892	0.898981	0.015910	0.824267
2	0.837194	0.826848	0.027080	0.700894
3	0.718523	0.701393	0.043794	0.516275
4	0.627375	0.625619	0.054901	0.393600
5	0.496313	.	0.068234	0.246326

EIGENVALUES OF INV(E)*H
= CANRSQ/(1-CANRSQ)

	EIGENVALUE	DIFFERENCE	PROPORTION	CUMULATIVE
1	4.6905	2.3472	0.5167	0.5167
2	2.3433	1.2760	0.2582	0.7749
3	1.0673	0.4182	0.1176	0.8925
4	0.6491	0.3222	0.0715	0.9640
5	0.3268	.	0.0360	1.0000

TESTS OF H0: THE CANONICAL CORRELATION IN THE CURRENT ROW AND ALL THAT FOLLOW ARE ZERO

	LIKELIHOOD RATIO	APPROX F	NUM DF	DEN DF	PR > F
1	0.01162035	30.6864	30	450	0.0
2	0.06612518	23.8241	20	375.728	0.0
3	0.22107577	19.3485	12	301.907	0.0001
4	0.45702792	18.3696	6	230	0.0001
5	0.75367378	18.9564	2	116	0.0001

MULTIVARIATE TEST STATISTICS AND F APPROXIMATIONS
S=5 M=0 N=55

STATISTIC	VALUE	F	NUM DF	DEN DF	PR > F
WILKS' LAMBDA	0.01162035	30.686	30	450	0.0
PILLAI'S TRACE	2.681362	22.358	30	580	0.0
HOTELLING-LAWLEY TRACE	9.076958	33.403	30	552	0.0
ROY'S GREATEST ROOT	4.690465	90.682	6	116	0.0

NOTE: F STATISTIC FOR ROY'S GREATEST ROOT IS AN UPPER BOUND

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

TOTAL CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.2554	0.7227	0.5063	-0.1326	-0.3678
TASK2	0.9234	-0.2428	-0.0896	-0.2001	-0.1837
TASK3	0.5960	-0.1826	0.6145	0.3348	0.3455
TASK4	0.4367	0.1526	-0.1044	0.8574	-0.1959
TASK5	0.4015	0.5241	-0.3054	-0.0417	0.3597
TASK6	0.3580	0.6975	-0.3353	0.0057	0.3641

BETWEEN CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.3013	0.7861	0.4727	-0.1080	-0.2371
TASK2	0.9539	-0.2313	-0.0733	-0.1429	-0.1037
TASK3	0.7077	-0.2000	0.5775	0.2747	0.2243
TASK4	0.5735	0.1849	-0.1085	0.7781	-0.1406
TASK5	0.5720	0.6885	-0.3444	-0.0411	0.2801
TASK6	0.4434	0.7966	-0.3287	0.0049	0.2465

WITHIN CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.1677	0.6191	0.5516	-0.1617	-0.5001
TASK2	0.8113	-0.2784	-0.1307	-0.3266	-0.3342
TASK3	0.3877	-0.1550	0.6632	0.4045	0.4655
TASK4	0.2533	0.1155	-0.1005	0.9240	-0.2353
TASK5	0.2184	0.3719	-0.2756	-0.0422	0.4052
TASK6	0.2206	0.5609	-0.3429	0.0066	0.4647

STANDARDIZED CANONICAL COEFFICIENTS

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.1325	1.0972	0.8173	-0.2653	-0.6910
TASK2	1.7648	-0.6520	-0.4586	-0.5763	-0.5145
TASK3	0.5293	-0.3400	1.1133	0.2542	0.9122
TASK4	0.2422	0.1693	-0.4273	1.1824	-0.5745
TASK5	0.3771	0.4899	-0.2998	-0.1933	0.2724
TASK6	0.2794	0.7091	-0.3962	0.0272	0.6967

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

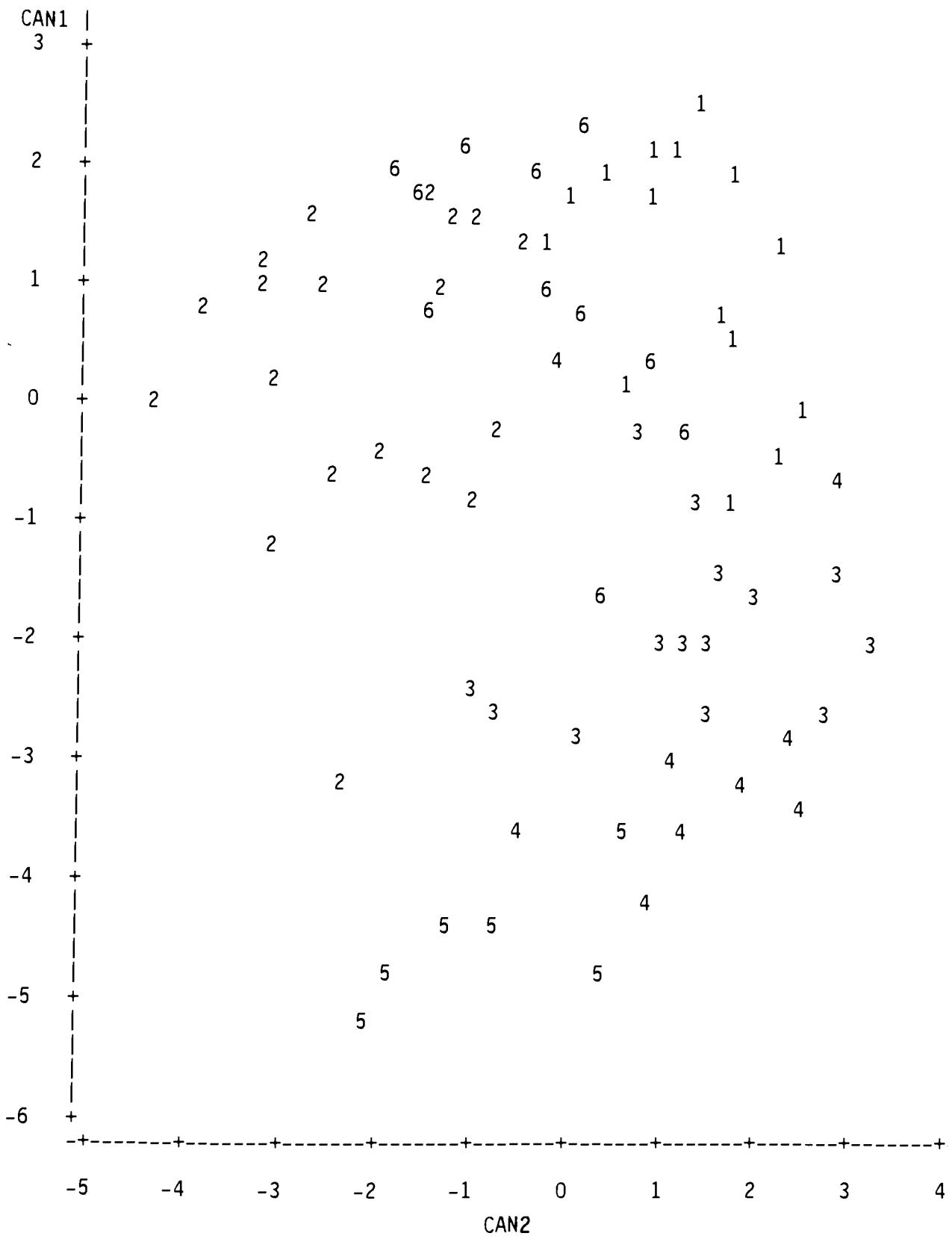
RAW CANONICAL COEFFICIENTS

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.149812157	1.241027542	0.924361142	-0.300027786	-0.781585772
TASK2	1.963057864	-0.725264793	-0.510099247	-0.641096244	-0.572312818
TASK3	0.629837529	-0.404566166	1.324708812	0.302443746	1.085488874
TASK4	0.351380556	0.245629432	-0.619906759	1.715389041	-0.833467902
TASK5	0.429227735	0.557611163	-0.341210967	-0.220014391	0.310100402
TASK6	0.318569025	0.808433137	-0.451629941	0.030999606	0.794238457

CLASS MEANS ON CANONICAL VARIABLES

CLUSTER	CAN1	CAN2	CAN3	CAN4	CAN5
1	1.8939	1.2696	0.2416	-0.6477	-0.2790
2	0.5401	-2.4048	0.5454	0.3091	-0.3732
3	-1.9433	1.3155	1.4572	0.8257	0.4512
4	-2.7584	1.1925	-1.8706	0.8982	-0.9794
5	-4.5864	-1.0247	-0.3204	-1.8528	0.3753
6	1.5364	-0.3510	-1.2898	0.3467	0.8709

Plot of Canonical variables identified by cluster



D.3 Unknown Domain Example

File: Unknown.dat

```
before conferences
set up schedule 15 minutes each and 5 minutes between
purchase index cards
gather books to be displayed the night of conferences
organize reports and index cards in order of conferences
xerox handouts for parents
staple handouts for parents
put last name of child on handouts
put name of child on index card
put reading level on index card
list child's strong points
list child's weaknesses
get conference form for each child
organize conference forms in order of conferences
night of conferencce
place conference schedule on door and on table near you
place handouts on table
place report cards and index cards on table
place pens on table
set up display books on small table outside the door
greet parents with smile and walk to the table
look at report card first
ask parents if they have any questions about the report
if concerns, discuss these first and make notes on index cards
go to index cards
explain reading level - have series book to look at
discuss child's strong points
discuss weaknesses - suggest ways 'WE' can help child improve
conclude conferrence quickly by summarizing any concerns
hand out parenting handouts - these were helpful to me as parent/teacher
thank parents for coming and walk them to the door
if time - write down any important notes for conference form
write up conference forms - date, who attended, what was discussed
if parents become antagonistic
do not argue - remain calm
take deep breath - try to get them to open up about problem
if they say something negative - repeat back to keep them talking
make phone call to parents who didn't show up
get information to parents who requested it
psychological testing , counselor
discuss concerns with principal
```

File:Unknown.rat

2 2 1 2 2 2 0
2 1 2 2 2 2 1
2 2 2 2 2 2 2
2 1 2 2 2 2 3
2 0 2 2 2 2 4
2 2 2 2 2 2 5
2 2 0 2 1 0 6
0 0 2 2 1 1 7
2 2 2 2 2 2 8
2 1 1 1 1 1 9
1 2 1 1 1 1 10
1 1 1 1 0 1 11
2 2 2 2 2 2 12
2 0 2 2 2 2 13
2 2 2 2 2 2 14
2 2 1 2 1 1 15
2 2 2 2 2 2 16
2 2 2 2 2 2 17
2 2 0 2 2 2 18
2 2 2 2 0 0 19
2 0 2 1 1 1 20
1 2 0 1 2 2 21
2 0 0 2 2 2 22
2 2 2 2 2 2 23
2 2 1 2 2 1 24
2 0 0 2 2 0 25
2 2 2 2 2 2 26
2 2 2 2 2 2 27
2 0 0 2 2 2 28
2 2 2 2 2 2 29
2 2 2 2 2 2 30
2 2 2 2 2 1 31
2 1 1 1 2 1 32
2 2 2 2 2 2 33
2 2 2 2 2 2 34
2 2 2 2 2 2 35
2 2 2 2 1 2 36
2 2 2 2 2 2 37
2 2 2 2 2 2 38
2 2 2 2 2 2 39
2 2 2 2 2 2 40
2 2 2 2 2 2 41
2 2 2 2 2 2 42
2 2 2 2 2 2 43
2 2 2 2 2 2 44
2 2 2 2 2 2 45
2 2 2 2 2 2 46
2 2 2 2 2 2 47
2 2 2 2 2 2 48
2 2 2 2 2 1 49
1 1 1 1 1 1 50
2 2 2 2 2 2 51
2 2 2 2 2 2 52
2 2 2 2 2 2 53

2	2	0	2	0	0	54
0	0	0	2	0	0	55
1	1	1	1	1	1	56
2	2	2	2	1	1	57
1	2	2	2	2	1	58
2	1	1	1	1	1	59
2	2	0	2	0	0	60
0	1	1	2	2	0	61
0	0	0	0	0	0	62
2	2	2	2	2	2	63
2	2	2	2	2	2	64
2	2	2	2	2	2	65
2	2	1	2	1	1	66
2	0	0	2	1	1	67
1	1	1	1	1	1	68
2	2	2	2	2	2	69
2	2	2	2	2	2	70
2	2	2	2	2	2	71
2	2	2	2	2	2	72
2	2	2	2	2	2	73
2	2	2	2	2	2	74
2	2	2	2	2	2	75
2	2	2	2	2	2	76
2	2	2	2	2	2	77
2	2	2	2	2	2	78
2	2	2	2	2	2	79
2	2	2	2	2	2	80
2	2	2	2	2	2	81
2	2	2	2	2	2	82
2	2	2	2	2	2	83
2	2	2	2	2	2	84
2	2	2	2	2	2	85
2	2	2	2	2	2	86
2	2	2	2	2	2	87
2	2	2	2	2	2	88
1	1	1	1	1	1	89
2	2	1	2	1	1	90
1	1	1	2	2	1	91
0	0	0	0	0	0	92
2	2	2	2	2	2	93
2	2	2	2	2	2	94
2	2	2	2	2	2	95
2	2	2	2	2	2	96
2	2	2	2	2	2	97
2	2	2	2	2	2	98
2	2	0	2	2	2	99
2	0	0	0	2	0	100
1	1	1	1	1	1	101
2	2	2	2	2	2	102
2	2	2	2	2	2	103
2	2	2	2	2	2	104
2	2	2	2	2	2	105
2	2	2	2	2	2	106
2	2	2	2	2	2	107
2	2	2	2	2	2	108
2	2	2	2	2	2	109
2	2	2	2	2	2	110

```
2 2 2 2 2 2 111
2 2 2 2 2 2 112
2 2 2 2 2 2 113
2 2 2 2 2 2 114
2 2 2 2 2 2 115
2 2 2 2 2 2 116
2 2 2 2 2 2 117
2 2 2 2 2 2 118
2 2 2 2 2 2 119
2 2 2 2 2 2 120
2 2 2 2 2 2 121
2 2 2 2 2 2 122
```

File: Unknown.lis

generic Task Indentification

OBS	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	2	0	0	0
4	0	0	0	2	0	0
5	0	0	0	0	2	0
6	0	0	0	0	0	2

FASTCLUS PROCEDURE

REPLACE=FULL RADIUS=0 MAXCLUSTERS=6 MAXITER=10 CONVERGE=0.02

INITIAL SEEDS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.00000	2.00000	0.00000	0.00000	0.00000	0.00000
3	0.00000	0.00000	2.00000	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000	2.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.00000	2.00000	0.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	2.00000

MINIMUM DISTANCE BETWEEN SEEDS = 2.828427

ITERATION	CHANGE IN CLUSTER SEEDS					
	1	2	3	4	5	6
1	3.9129	2.88675	2.44949	1.69967	3.89667	3.9423
2	0.554555	0.957427	0	1.18492	0.842847	0
3	0.0367707	0.306186	0	0.27798	0.165645	0.329485
4	0.0123457	0	1	0.226351	0.201636	0.340773
5	0.0240982	0	0	0	0.345196	0.497649
6	0.0119091	0.209165	0	0	0.255538	0.124722
7	0	0.165145	0	0	0.234882	0
8	0	0.193061	0	0	0.214267	0
9	0	0	0	0	0	0

CLUSTER LISTING

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
1	0	1	0.98845
2	1	1	0.97634
3	2	1	0.02916
4	3	1	0.97634
5	4	5	1.64804
6	5	1	0.02916
7	6	4	1.67037
8	7	3	1.00000
9	8	1	0.02916
10	9	2	0.71414
11	10	2	1.05357
12	11	2	0.95394
13	12	1	0.02916
14	13	5	1.64804
15	14	1	0.02916
16	15	6	0.66667
17	16	1	0.02916
18	17	1	0.02916
19	18	5	1.351
20	19	6	1.66667
21	20	2	1.51987
22	21	5	1.71414
23	22	5	1.17589
24	23	1	0.02916
25	24	6	0.88192
26	25	4	1.76733
27	26	1	0.02916
28	27	1	0.02916
29	28	5	1.17589
30	29	1	0.02916
31	30	1	0.02916
32	31	6	0.81650
33	32	5	1.17589
34	33	1	0.02916
35	34	1	0.02916
36	35	1	0.02916
37	36	1	0.98845
38	37	1	0.02916
39	38	1	0.02916
40	39	1	0.02916
41	40	1	0.02916
42	41	1	0.02916
43	42	1	0.02916
44	43	1	0.02916
45	44	1	0.02916
46	45	1	0.02916
47	46	1	0.02916
48	47	1	0.02916
49	48	1	0.02916
50	49	6	0.81650
51	50	2	0.33166
52	51	1	0.02916
53	52	1	0.02916

OBS	CON_NO	CLUSTER	DISTANCE FROM SEED
54	53	1	0.02916
55	54	4	1.76733
56	55	4	1.76733
57	56	2	0.33166
58	57	6	0.57735
59	58	6	1.20185
60	59	2	0.71414
61	60	4	1.76733
62	61	3	1.00000
63	62	4	2.11111
64	63	1	0.02916
65	64	1	0.02916
66	65	1	0.02916
67	66	6	0.66667
68	67	4	1.49485
69	68	2	0.33166
70	69	1	0.02916
71	70	1	0.02916
72	71	1	0.02916
73	72	1	0.02916
74	73	1	0.02916
75	74	1	0.02916
76	75	1	0.02916
77	76	1	0.02916
78	77	1	0.02916
79	78	1	0.02916
80	79	1	0.02916
81	80	1	0.02916
82	81	1	0.02916
83	82	1	0.02916
84	83	1	0.02916
85	84	1	0.02916
86	85	1	0.02916
87	86	1	0.02916
88	87	1	0.02916
89	88	1	0.02916
90	89	2	0.33166
91	90	6	0.66667
92	91	5	1.17589
93	92	4	2.11111
94	93	1	0.02916
95	94	1	0.02916
96	95	1	0.02916
97	96	1	0.02916
98	97	1	0.02916
99	98	1	0.02916
100	99	5	1.35173
101	100	4	2.11111
102	101	2	0.33166
103	102	1	0.02916
104	103	1	0.02916
105	104	1	0.02916
106	105	1	0.02916
107	106	1	0.02916

108	107	1	0.02916
109	108	1	0.02916
110	109	1	0.02916
111	110	1	0.02916
112	111	1	0.02916
113	112	1	0.02916
114	113	1	0.02916
115	114	1	0.02916
116	115	1	0.02916
117	116	1	0.02916
118	117	1	0.02916
119	118	1	0.02916
120	119	1	0.02916
121	120	1	0.02916
122	121	1	0.02916
123	122	1	0.02916

CLUSTER SUMMARY

CLUSTER NUMBER	FREQUENCY	RMS STD DEVIATION	MAXIMUM DISTANCE FROM SEED TO OBSERVATION	NEAREST CLUSTER	CENTROID DISTANCE
1	84	.0888183	0.9885	6	1.3650
2	10	0.3305	1.5199	4	1.5093
3	2	0.5774	1.0000	2	1.9261
4	9	0.8022	2.1111	2	1.5093
5	9	0.6048	1.7141	2	1.6877
6	9	0.4082	1.6667	1	1.3650

STATISTICS FOR VARIABLES

VARIABLE	TOTAL STD	WITHIN STD	R-SQUARED	RSQ/(1-RSQ)
TASK1	0.468237	0.327441	0.531013	1.132256
TASK2	0.639351	0.406583	0.612166	1.578420
TASK3	0.680746	0.301142	0.812328	4.328454
TASK4	0.418485	0.285782	0.552765	1.235961
TASK5	0.552097	0.325394	0.666870	2.001833
TASK6	0.625015	0.181007	0.919567	11.432662
OVER-ALL	0.571841	0.311857	0.714775	2.506010

PSEUDO F STATISTIC = 58.64
APPROXIMATE EXPECTED OVER-ALL R-SQUARED = 0.51988
CUBIC CLUSTERING CRITERION = 19.974
WARNING: THE TWO ABOVE VALUES ARE INVALID FOR CORRELATED VARIABLES

CLUSTER MEANS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	2.00000	1.97619	1.98810	2.00000	1.98810	2.00000
2	1.30000	1.00000	1.10000	1.00000	0.90000	1.00000
3	0.00000	0.50000	1.50000	2.00000	1.50000	0.50000
4	1.33333	0.66667	0.00000	1.33333	0.66667	0.11111
5	1.77778	0.88889	0.66667	1.77778	2.00000	1.77778
6	1.88889	2.00000	1.55556	2.00000	1.33333	0.88889

CLUSTER STANDARD DEVIATIONS

CLUSTER	TASK1	TASK2	TASK3	TASK4	TASK5	TASK6
1	0.00000	0.15337	0.10911	0.00000	0.10911	0.00000
2	0.48305	0.47140	0.31623	0.00000	0.31623	0.00000
3	0.00000	0.70711	0.70711	0.00000	0.70711	0.70711
4	1.00000	1.00000	0.00000	1.00000	0.86603	0.33333
5	0.44096	0.92796	0.86603	0.44096	0.00000	0.44096
6	0.33333	0.00000	0.52705	0.00000	0.70711	0.33333

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

123 OBSERVATIONS — 122 DF TOTAL
6 VARIABLES — 117 DF WITHIN CLASSES
6 CLASSES — 5 DF BETWEEN CLASSES

	CANONICAL CORRELATION	ADJUSTED CANONICAL CORRELATION	APPROX STANDARD ERROR	SQUARED CANONICAL CORRELATION
1	0.979947	0.978564	0.003595	0.960295
2	0.791932	0.770976	0.033756	0.627156
3	0.723534	0.717194	0.043140	0.523501
4	0.581093	0.580990	0.059965	0.337669
5	0.033172	-.089149	0.090436	0.001100

EIGENVALUES OF INV(E)*H
= CANRSQ/(1-CANRSQ)

	EIGENVALUE	DIFFERENCE	PROPORTION	CUMULATIVE
1	24.1859	22.5038	0.8802	0.8802
2	1.6821	0.5834	0.0612	0.9414
3	1.0986	0.5888	0.0400	0.9814
4	0.5098	0.5087	0.0186	1.0000
5	0.0011		0.0000	1.0000

TESTS OF H0: THE CANONICAL CORRELATION IN THE CURRENT ROW AND ALL THAT FOLLOW ARE ZERO

	LIKELIHOOD RATIO	APPROX F	NUM DF	DEN DF	PR > F
1	0.00466691	42.3897	30	450	0.0
2	0.11754009	17.0392	20	375.728	0.0
3	0.31525256	13.7618	12	301.907	0.0001
4	0.66160201	8.7946	6	230	0.0001
5	0.99889960	0.0639	2	116	0.9381

MULTIVARIATE TEST STATISTICS AND F APPROXIMATIONS
S=5 M=0 N=55

STATISTIC	VALUE	F	NUM DF	DEN DF	PR > F
WILKS' LAMBDA	0.004666907	42.390	30	450	0.0
PILLAI'S TRACE	2.449722	18.571	30	580	0.0
HOTELLING-LAWLEY TRACE	27.47752	101.117	30	552	0.0
ROY'S GREATEST ROOT	24.18587	467.593	6	116	0.0

NOTE: F STATISTIC FOR ROY'S GREATEST ROOT IS AN UPPER BOUND

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

TOTAL CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.5745	0.0861	-0.3021	0.6918	-0.1542
TASK2	0.6929	-0.3856	-0.1818	0.3464	0.1568
TASK3	0.8573	-0.3915	-0.0177	-0.1746	-0.1048
TASK4	0.5750	-0.2185	0.4598	0.5293	0.2968
TASK5	0.7631	0.2116	0.3555	0.1979	-0.4503
TASK6	0.9440	0.3168	0.0419	0.0110	0.0701

BETWEEN CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.7726	0.0935	-0.2999	0.5517	-0.0070
TASK2	0.8678	-0.3903	-0.1681	0.2573	0.0066
TASK3	0.9321	-0.3440	-0.0142	-0.1126	-0.0039
TASK4	0.7578	-0.2328	0.4475	0.4137	0.0132
TASK5	0.9157	0.2052	0.3150	0.1408	-0.0183
TASK6	0.9646	0.2616	0.0316	0.0067	0.0024

WITHIN CANONICAL STRUCTURE

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	0.1672	0.0768	-0.3045	0.8221	-0.2251
TASK2	0.2217	-0.3781	-0.2015	0.4527	0.2517
TASK3	0.3943	-0.5519	-0.0283	-0.3280	-0.2417
TASK4	0.1713	-0.1995	0.4746	0.6441	0.4435
TASK5	0.2634	0.2239	0.4251	0.2791	-0.7797
TASK6	0.6632	0.6820	0.1020	0.0315	0.2469

STANDARDIZED CANONICAL COEFFICIENTS

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	-0.1325	0.2555	-1.1588	0.9263	-0.3649
TASK2	1.0247	-0.7441	-0.4539	0.2708	0.1557
TASK3	1.5949	-1.4881	-0.2744	-0.7495	-0.4192
TASK4	-0.1694	-0.3943	1.3160	0.5654	0.6903
TASK5	0.3693	0.0428	1.1064	0.2289	-1.2737
TASK6	2.8912	1.9477	-0.4083	-0.6114	1.0977

Canonical discriminant analysis of task clusters

CANONICAL DISCRIMINANT ANALYSIS

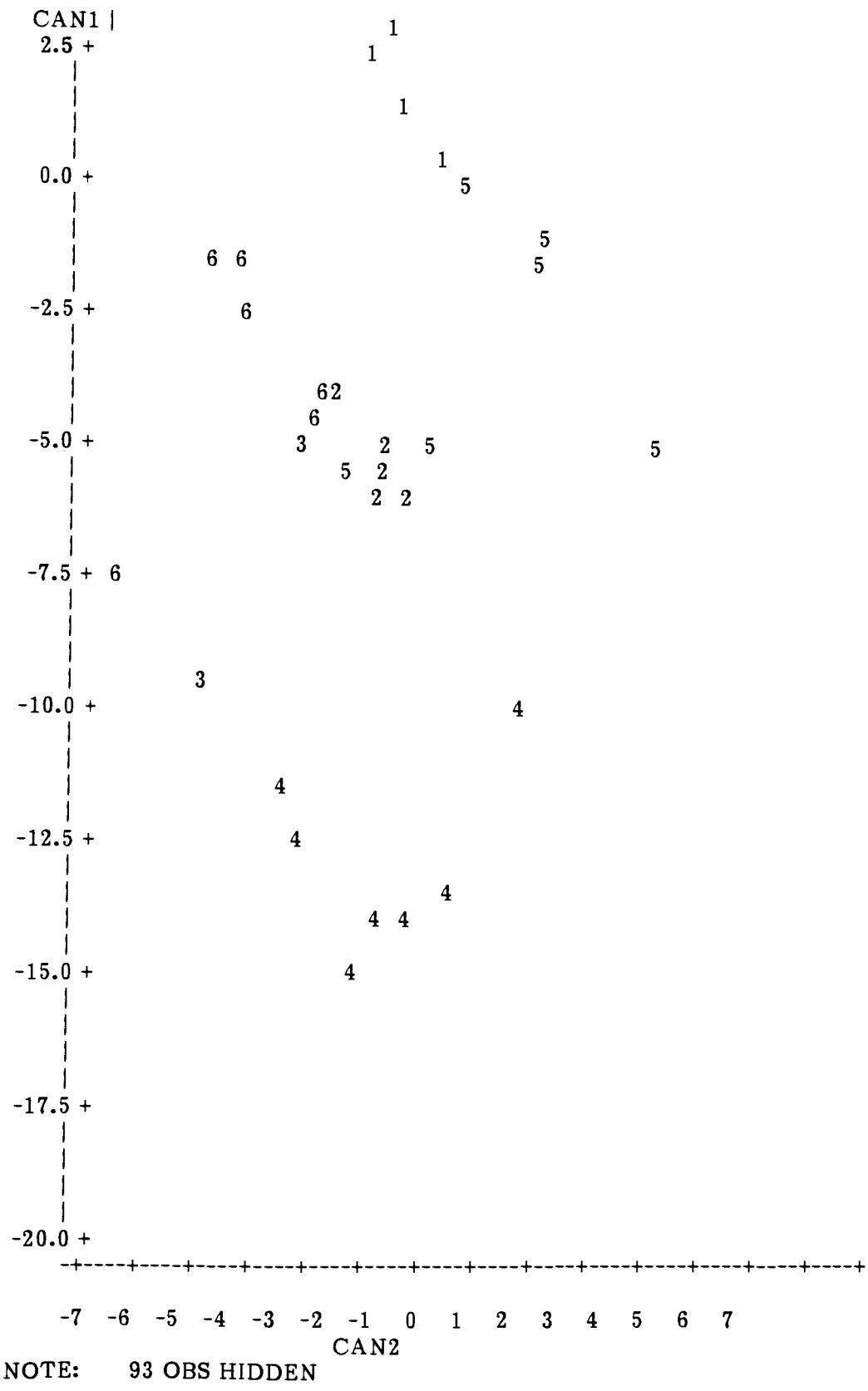
RAW CANONICAL COEFFICIENTS

	CAN1	CAN2	CAN3	CAN4	CAN5
TASK1	-0.283033570	0.545686379	-2.474847227	1.978331466	-0.779294937
TASK2	1.602682585	-1.163892338	-0.709935435	0.423494033	0.243466729
TASK3	2.342849567	-2.185974303	-0.403116797	-1.100927479	-0.615740866
TASK4	-0.404892813	-0.942287756	3.144581391	1.350989787	1.649470798
TASK5	0.668991759	0.077490112	2.004020137	0.414662579	-2.306942784
TASK6	4.625742643	3.116312299	-0.653233383	-0.978157949	1.756210863

CLASS MEANS ON CANONICAL VARIABLES

CLUSTER	CAN1	CAN2	CAN3	CAN4	CAN5
1	2.9081	-0.0626	-0.0234	0.0126	0.0099
2	-5.4878	0.3746	-1.9119	-1.6319	-0.0309
3	-7.3004	-3.0812	6.1727	-2.7670	0.0013
4	-13.0114	0.0831	-0.1531	0.7269	0.0663
5	-2.7975	3.4881	1.3014	0.4894	-0.0523
6	-3.6134	-2.7187	-0.1773	1.0944	-0.0722

Plot of Canonical variables identified by cluster



NOTE: 93 OBS HIDDEN