A Knowledge Engineering Taxonomy for Intelligent Tutoring System Development*

Pamela K. Fink, Ph.D. and L. Tandy Herren, Ph.D.
Southwest Research Institute
6220 Culebra Road
San Antonio, TX 78228
(210) 522-3762

ABSTRACT

This paper describes a study addressing the issue of developing an appropriate mapping of knowledge acquisition methods to problem types for intelligent tutoring system development. Recent research has recognized that knowledge acquisition methodologies are not general across problem domains; the effectiveness of a method for obtaining knowledge depends on the characteristics of the domain and problem solving task. Southwest Research Institute developed a taxonomy of problem types by evaluating the characteristics that discriminate between problems and grouping problems that share critical characteristics. Along with the problem taxonomy, heuristics that guide the knowledge acquisition process based on the characteristics of the class are provided.

1.0 INTRODUCTION

Practical experience in knowledge-based systems indicates that how a system is designed and implemented depends on the characteristics of the task that the system is to perform or teach. Researchers in artificial intelligence are beginning to analyze the relationship between knowledge representation, system architecture, and task type. Such work is related to a continuing focus in the psychological and educational literature on the relationship between task characteristics and instructional strategy in curriculum development and on the relationship between task characteristics and human factors engineering.

The growing focus on task characteristics in knowledge-based system development signals the maturation of the field from one searching for the single best representational format and inferencing strategy to one that recognizes the diversity of problems and the need to address each according to its characteristics. Not surprisingly, human cognition appears to utilize multiple representational formats and reasoning techniques. Different representations store different types of information (e.g., scripts store simple, well-structured sequences of events while rules store single-level inferences) and dictate appropriate methods for reasoning about that information. The flexibility to employ the best and most appropriate representation and reasoning strategy in a given situation undoubtedly contributes to a human's ability to effectively solve problems.

In the development of an intelligent tutoring system (ITS), domain expertise of the appropriate kind and level for teaching must be implemented in the expert module to be used by the instructional portion of the system for training. This essentially constitutes the design and implementation of a specialized knowledge-based system that contains the knowledge and problem solving skills that must be taught to the student. The development of an ITS is equivalent to the development of several very different expert systems that all must work together, requiring extensive effort on the part of experts from all of the fields concerned. Because of the extensive effort involved in ITS development, it is crucial for researchers to evaluate the nature of the task to be taught and to pair it with an appropriate knowledge acquisition strategy.

Due to the amount of knowledge that must be embedded in ITSs, they can be very expensive to build. They require many hours of knowledge engineering in which the task is characterized and the knowledge is collected for task performance and for instruction. Then, the design of each of the four major modules needs to be developed and finally implemented. Depending on the size and complexity of the task to be taught, this extensive process can take many person-years of effort to complete. Some work has been done to reduce the amount of effort required to build an ITS. In most cases, these are software tools that support the design and

implementation process by providing frameworks and authoring facilities for the various components of an ITS. Such tools work much the way a knowledge-based system development tool works for the development of knowledge-based systems. They reduce the effort required to write the code. However, they provide little direction on how to go about acquiring the knowledge that must be implemented into the code.

To acquire the knowledge, the ITS or knowledge-based system developer must use either a traditional verbal or an automated method of knowledge acquisition. The process of knowledge engineering has long been marked as a bottleneck in any knowledge-based system development. This is not surprising since the process involves humans who are expert in a given domain communicating with other humans who are not expert in the domain about their expertise and problem solving skills. The knowledge engineering experts are, however, expert in computer programming and knowledge representation and they must be able to organize, structure, and convert the domain expertise as they understand it into a computer program. All of this communication results in a situation similar to the "telephone game," where something is lost or misinterpreted with each communication act. The problem, of course, is magnified by the fact that the individuals involved do not tend to speak the same "language" and the concepts being communicated can be quite complex. A good, simplified description and illustration of the knowledge engineering process is given in Yost and Newell (1989).

The knowledge acquisition problem has been acknowledged since the early days of knowledge-based system development. Because the expert module of an ITS is essentially a knowledge-based system, ITS development suffers from the same difficulty with knowledge acquisition. Many person-years of effort go into the development of a single knowledge base. The ideal solution, of course, is to skip the knowledge engineer and have the domain expert generate the knowledge base directly. The problem with this approach is that the domain expert usually has no real knowledge or experience in computer programming and knowledge representation. Research into the development of tools to support the knowledge acquisition process has been in two directions: 1) to support the knowledge engineer in structuring and encoding the knowledge acquired and 2) to provide an interface that would allow the domain expert to enter his/her knowledge directly into the computer program. However, the knowledge engineering process remains very time consuming and expensive.

2.0 FOUNDATION FOR THE NEXT GENERATION KNOWLEDGE ACQUISITION TOOLS

Completely automated knowledge acquisition tools that allow a domain expert to generate a sophisticated, complex knowledge-based system with no support from an individual knowledgeable in computer programming and artificial intelligence is a lofty goal. Current understanding of human learning and cognition is not at a level to allow the development of completely general, domain independent automated knowledge acquisition tools capable of developing arbitrarily complex knowledge-based systems in any domain. In addition, most domain experts do not have the time or inclination to submit themselves to extensive, tedious question-answer sessions with a computer program. Another approach to the knowledge acquisition problem that leaves the human knowledge engineer in the loop might be a better short-term solution.

It has been readily acknowledged among practitioners that the knowledge engineering process, as performed currently, is more an art than a science. Of course, there are techniques with names attached to them such as structured and unstructured interviews, example cases, etc., and there are recommended practices such as taping the interviews, generating transcripts, and then editing and modifying the resulting knowledge acquired based on domain expert feedback. However, the quality of the resulting system is still highly dependent on the personalities and skills of the individuals involved. The key to a successful knowledge engineering process (i.e., a useful, working knowledge-based system) is the ability to extract the general problem solving methodologies utilized and the specific domain knowledge needed to find a solution to a given problem.

Thus, the goal of the research reported on in this paper was to develop a methodology that could be used to typify the expert problem solving strategies and the types of specific domain knowledge needed that would allow the knowledge engineer to categorize the task. This initial categorization could then provide further methodologies for understanding the key characteristics of the given kind of task. These techniques could eventually be automated, but initial experience with, and evaluation of, the techniques should be gained through the use of good human knowledge engineers. The approach to generating such methodologies and classes was to

study a broad spectrum of problem solving tasks, analyze the problem solving techniques used, and pair knowledge engineering techniques to the acquisition and codification of specific problem solving strategies needed in an ITS. The result of this process is a proposed taxonomy of problem solving tasks characterized by problem solving strategies and types of domain knowledge, as well as some recommendations concerning knowledge acquisition methodologies relevant to the particular task.

Recently researchers have become more and more interested in basing their selection of a knowledge acquisition technique on the characteristics of the task. The ultimate goal is to find a way to attenuate the well-known knowledge acquisition bottleneck. Tailoring the knowledge acquisition process to the task will reduce the problems associated with eliciting expertise.

For example, Chandrasekaran (1985) describes six generic problem-solving tasks in knowledge-based reasoning, including classification, state abstraction, knowledge-directed retrieval, object synthesis by plan selection and refinement, hypothesis matching, and assembly of compound hypotheses for abduction. These tasks correspond to problem solving methods that can be combined to perform knowledge-based reasoning for an application. Bylander and Chandrasekaran (1987) suggest that generic tasks can be associated with a specific knowledge acquisition methodology.

Boose and Bradshaw (1987) proposed a set of knowledge acquisition strategies that link with appropriate problem-solving methods. The knowledge acquisition strategies establish distinctions between alternatives, problem decomposition, combining and propagating information, testing of knowledge, combining multiple sources of knowledge, incremental expansion of knowledge, and providing process guidance. However, Kitto and Boose caution that a knowledge acquisition system must be able to acquire knowledge about aspects of the problem-solving process that are unique to a given application.

McDermott (1988) presented a preliminary taxonomy of problem-solving methods based on the assumption that there are families of tasks that share abstract control knowledge and that the abstract control knowledge provides strong guidance as to what knowledge is required and how that knowledge should be encoded. He refers to the control methods as role-limiting methods because they strongly guide knowledge collection and encoding, i.e., task specific knowledge fills one of a few number of roles within the control framework. McDermott argues that it is likely there are hundreds of role-limiting methods. He discusses the following role-limiting methods and knowledge acquisition tools:

- cover and differentiate - implemented in MOLE
- propose and refine - implemented in SALT
- qualitative reasoning - implemented in YAKA
- acquire and present - implemented in KNACK
- extrapolate from a similar case - implemented in SIZZLE

What researchers in psychology and artificial intelligence have neglected to specify is how to determine in the first place whether or not an application task involves a certain characteristic. How do you identify a diagnostic task? How do you know if the task requires declarative or procedural knowledge? Or both? Does it matter? This issue is not entirely self evident but is, in fact, the reason that knowledge acquisition remains largely an art. Good knowledge engineers have little difficulty discerning the fundamental requirements of a task, but there is no formal method for making these judgments and no indication of how accurate they are once the judgments are made. The research reported on this paper explored these issues.

3.0 RESEARCH APPROACH TO DEVELOPING A KNOWLEDGE ACQUISITION TAXONOMY

The approach taken in this research was a very pragmatic one. The work was based on years of experience in performing knowledge acquisition for the design and implementation of numerous knowledge-based and intelligent tutoring systems in a wide variety of problem solving domains. One of the underlying goals has been to minimize the amount of time required with the expert. Experience has shown that expert time is usually very limited and it helps to make the most of the time that you do have. Another underlying goal of the research was

to develop an approach to knowledge acquisition that would help a knowledge engineer characterize the problem solving task in some useful way and to scope the effort. Thus, much of the questioning is oriented towards acquiring knowledge of the inputs to the task, the result or outputs from the task, reasoning mechanisms that operate on the input and generate the output, the environment in which the problem solving takes place, and the attributes of the experts who perform the task. The effort was less concerned with total accuracy of the knowledge obtained than with discovering the problem solving approach(es) used. Accuracy can come later when the software is under development and the expert can observe the system's explicit behavior and suggest concrete corrections.

The research proceeded by identifying a set of representative tasks, interviewing experts in the selected tasks, classifying these tasks into a taxonomy based on their characteristics, and generating recommendations for performing knowledge acquisition based on the task classification. In some sense, the work to develop a taxonomy needed to be done before we could make sense of the results gained from a series of knowledge acquisition interviews. But, at the same time, the knowledge acquisition interviews supplied the information on tasks that could support the generation of a taxonomy. Thus, because this was an initial research project, a bootstrapping approach had to be taken. Therefore, two interviews were performed for each problem solving task included in the study. This allowed us to gain some insight into the problem solving tasks, generate some hypotheses and classification schemes, and then informally test these hypotheses and classifications through a second interview. The following sections describe the four main phases of the research that resulted: 1) performing the first interview, 2) analyzing the results of the first interview, 3) performing the second interview, and 4) analyzing the results of the second interview and developing the taxonomy.

3.1 The First Interview

In developing the first interview, three major aspects had to be addressed. First, a set of problem solving tasks had to be selected for analysis. Second, a means by which each task could be characterized and rated had to be developed so that a comparison could be made between tasks. Third, a knowledge acquisition approach had to be identified/developed that could be used to acquire the information on each problem solving task. Each of these aspects of the approach is addressed below.

In selecting a set of problem solving tasks for analysis in this research, the goal was to find a representative set with as much variety as possible. The following list provides the task types that we felt needed to be represented in the set selected for interview.

- diagnostic task
- training task
- high performance task
- form fill-out
- managerial/supervisory task
- design task
- planning task
- monitoring with a time factor
- perceptually-oriented task
- bin-packing/np-complete task (exponential/combinatorial growth problem)
- numerical task
- data intensive (no real time factor), e.g., acquire and present

As a result, we interviewed experts in the following areas:

- (diagnostics) medical diagnosis
- (diagnostics) equipment diagnosis
- (training) training pilots
- (training) training a foreign language
- (high performance) flight controller console operations
- (high performance/knowledge-rich) surgery

- **(form fill-out)** contracting
- **(people-oriented)** personnel management/leadership training
- **(design)** software design
- **(planning)** acquisition program management
- **(monitoring/time constrained)** air traffic control
- **(perceptual)** weather forecasting
- **(bin-packing)** cargo loading
- **(numerical)** accounting
- **(data intensive/no time constraint)** DRAIR generation
- **(planning)** scientific protocol design

Obviously, these tasks were still rather broad and we did not intend to try to perform knowledge acquisition on the entire area indicated by the job label. Instead, we planned on allowing the individual expert's specific job requirements to narrow the scope in each problem solving area. Individuals considered experts in the selected areas were identified and asked to take part in the effort. Only one expert in each area was selected. The focus of the research was on task type, not on how numbers and types of experts could alter the knowledge acquisition process.

To generate a set of task characteristics to support the rating of tasks, a literature search was performed in both the psychology and artificial intelligence fields to identify existing approaches to problem solving taxonomies. This search helped to identify a list of 123 characteristics that fell into 16 different categories on which a particular problem solving task could be rated. This list was intended to be relatively exhaustive by including all the attributes that might be relevant to how problem solving tasks could be categorized. We suspected that some redundancy existed, and that some of the characteristics would not prove to be relevant for the problem of knowledge acquisition. However, we wanted to be sure that as much as possible was considered when analyzing a problem solving task.

The of characteristics compiled for this study originated from a compilation of existing taxonomies (Gawron, Drury, Czaja, and Wilkins, 1989). The list of characteristics fell into 15 categories:

Reasoning Techniques -	characteristics related to the manipulation of information during task performance
Problem-Solving Techniques -	search strategies for finding a solution
Inputs -	data input to the problem solving process
Task Complexity -	characteristics of the data array and problem solving space that influence the task
Technical Dimension -	the degree to which the task requires a technical background or skill
Motor Processes -	physical requirements of the task
Information Processing -	characteristics of the way data is manipulated during task performance
Problem Solving Tasks -	cognitive operations on the data
Recall -	the role of memory in the task
Perceptual Processes -	the perceptual requirements of the task
Environment -	physical and psychological characteristics of the environment in which the task is performed
Personal Characteristics -	characteristics of the experts required to perform the task
Type of Domain -	knowledge-rich and/or high performance
Hardware/Equipment/Tools -	items or devices used during the task
Communication Processes -	verbal and nonverbal communication required during task performance

We selected these categories and characteristics because of their relevance to knowledge acquisition and system development. We felt that some of the characteristics would be related to how knowledge acquisition should be conducted and others would be more related to how an intelligent system for the task should be designed.

The technique used in performing the initial knowledge acquisition for each task involved the use of a structured interview. Based on our experience in knowledge acquisition, we believed it to be the most general approach to acquiring knowledge when little was yet known about a particular task. As a result, a series of questions was generated directly from the list of task characteristics. These questions were designed to elicit information that would help to discern the importance of the various task characteristics to the task under consideration.

These questions provided a guide to ensure that we obtained information about each of the 123 characteristics during the interview. We framed the questions such that they were understandable to a lay person and grouped and ordered them into a logical progression from general requirements and inputs, to data processing requirements and outputs. This interview usually required approximately one and a half hours to perform and two researchers performed the interview together. All interviews were conducted at the expert's workplace.

3.2 Evaluating the First Interview

Based on the interview performed with the human experts, two raters reached agreement on the ratings for each task on the 123 attributes. The raters used a 5 point Likert-type scale in which 0 indicated the attribute is not relevant at all for the task and 4 indicated that the attribute is critical for task performance. The ratings were used as a measure of the importance of a characteristic for task performance.

The major goal of the data analysis was to develop a principled way of classifying tasks based on the knowledge acquired from the first interview and to make recommendations concerning how to proceed with the knowledge acquisition process based on this classification. The data generated from the first set of interviews consisted of a 16 by 123 matrix with a rating of zero to four for each of the 123 characteristics for each of the 16 tasks. We wanted to answer a number of questions concerning this data, including what characteristics are relevant for a meaningful classification of the tasks, which tasks are related based on their characteristics and which were not, and what the key characteristics are about a class of tasks that would affect the way we would want to proceed with the second interview.

3.2.1 Statistical Analyses

The ratings of the characteristics entered into a factor analysis and cluster analysis. Some characteristics were dropped from the analysis because the ratings for all 16 tasks was 0. Additionally, we collapsed the scores across a category of characteristics if a summary score for that category made sense. For example, a summary score for the category "inputs" reflects the degree to which the task requires external information. A summary score for the category "reasoning techniques" would not produce a meaningful index.

A principle components factor analysis with a varimax rotation was performed on the characteristic ratings. The factor analysis revealed three main factors in the characteristics. The first factor consisted of characteristics related to design tasks, the second factors consisted of characteristics of tasks involving a physically-based skill, and the third factor corresponded to characteristics of tasks involving statistical reasoning and mental modelling. These factors were evident in the task clustering. In a cluster analysis of the tasks, the first cluster, software design and protocol design, includes the design tasks we examined. The second cluster, pilot training and surgery, are tasks that require physically-based skills. The relevance of the third factor is less clear. Protocol design, medical diagnosis, weather forecasting, and drair generation require statistical reasoning while software design, weather forecasting, console operations, surgery and medical diagnosis require the use of mental models. This factor does not clearly map to the cluster analysis.

3.2.2 An Analysis Based on Experience

The issue of whether or not a task is formally trained was not clearly delineated in the characteristics by which we were rating each task. However, it was clear from our initial interview how an expert became an expert in the field and we found ourselves recommending an examination of the curriculum for the second interview for any task that was formally trained in the operational environment. As a result, we categorized the tasks by whether or not an individual received formal training in performing the actual task(s) that constituted their area of

expertise. The tasks fell into two relatively equal sets where aircraft piloting, air traffic control, console operations, weather forecasting, cargo loading, foreign language, surgery, and medical diagnosis all are heavily trained before an individual is allowed to perform the task, and where equipment diagnosis, form fill-out, program management, DRAIR generation, software design, leadership, accounting, and protocol design are not formally trained. Tasks appearing in the latter set tend to require at least a general background education in a particular field such as business, computer science, or operations research as a foundation on which they can build expert skill through experience on the job.

Once these two sets of tasks were generated, we then examined the other characteristics of the tasks to see if there were any unifying themes. Among the formally trained tasks the unifying characteristics seemed to be an issue involving human safety, as well as quite often a physical component. They were also highly proceduralized, even if the experts did not necessarily follow a procedure once they became expert. Individuals entering the field are taught a procedure to follow that allows them to become efficient problem solvers. Tasks that were not formally trained tended to be less well-defined in terms of goals and results. Such tasks do not tend to have definitive right and wrong answers and problem solving tends to be oriented towards breaking the task down into relatively independent subtasks. This approach helps to deal with the complexity and inexactness of the problem.

In addition, in those tasks that involved a procedure the skills required for the task tended to build sequentially on one another. So for example, when a person learns to fly a plane they begin by learning how to fly level, then learn to take-off, then learn to land. Each skill builds on the previous skill. However, the skills required in the set of tasks not formally trained tended to be componential. That is, the problem solver has a toolbox of skills relevant for different aspects of the problem solving process. They learn each of the skills independently. For example, the expert in protocol analysis had skill in experimental design, statistical analysis, research methodology, and electronics. These are independent skills and are learned during a formal education in a content area, such as biomedical engineering. Based on these observations, we labeled the first set of tasks procedurally-oriented tasks and the second set we call componential tasks.

3.2.3 Results and Conclusions from the First Interview

When all of the various analyses of the data from the first interview were compared, we began to see patterns in the data that confirmed our experiential analysis. For example, the eight cluster statistical analysis generated the following clusters:

- pilot training, surgery
- software design, protocol design
- cargo loading, accounting, program management, leadership, equipment diagnosis, DRAIR generation, form fill-out
- medical diagnosis
- foreign language training
- air traffic control
- weather forecasting
- console operations

The cluster analysis corresponds greatly to our informal analysis. The first cluster, software design and protocol design, and the large third cluster primarily contain tasks that do not receive formal training in the operational environment, such as program management, accounting, and drair generation. However, these tasks require a lot of background knowledge often obtained through formal education. The remaining clusters are tasks that receive a large amount of training in the operational environment. Surgical training stems from years of internships and residencies, air traffic control involves training specific to each airport at which the controller works, and console operations requires both a formal educational relevant to the specific console and continual on-the-job training to remain proficient.

In addition, the factor analysis of the characteristics indicated that the clusterings were due to differences among the design, high performance, and model-based/statistical factors that roughly correspond to the attributes we saw intuitively as being key to the commonalities that were responsible for the generation of the two sets of tasks.

This information was used to help guide the work in generating the approaches used in the second set of interviews.

3.3 The Second Interview

Once a clustering of the tasks became apparent, we examined the implications for the second interview for each task. However, when we compared what we believed we should do for the second interview based on our own intuition versus what seemed appropriate based on the clusterings, we saw that the clusterings did not have as much impact as expected on the desired approach. Based on this assessment, we determined that the second interview is still too early in the knowledge acquisition process for much delineation to take place in terms of how to proceed with the knowledge acquisition task. For example, even though a wealth of information is available through existing curriculums for those tasks that are formally trained, we sometimes felt that the second interview was too early to effectively utilize the information. In most cases, we believed that going through an example in some form would be the most appropriate approach to the second interview. The differences among tasks was exploited mainly in how the second interview should elicit the example and how many examples should be examined.

The major exceptions to the use of an example were foreign language training, software design, program management, and leadership training. Foreign language training was an exception because an example is almost meaningless, so examination of a component of the curriculum was recommended. Software design and program management were exceptions because the tasks are so large and ill-defined at their highest level that an example would have little meaning even if it could be formulated. Thus, we selected one of the components of software design and of program management on which to focus further discussion. This would allow an iterative and principled approach to breaking the task down into sub-tasks until a task of a workable size was found. Leadership training was an exception because it was unclear how an example could even be formulated from which a discussion could evolve. Because there are a number of tools that are used in leadership training (TQM) we chose to examine one of those tools, namely team building.

One minor exception to the use of examples occurred with aircraft pilot training. For this task, the desired approach was to examine an example, but we wanted to examine the easiest examples, preferably from the first few flights that a student pilot would take. In this case a well-defined curriculum existed, but examples would be most useful in furthering our understanding of the task. Of course, they teach the students to perform the task, as well, through the use of well-selected examples.

In the cases where the second interview should consist of eliciting an example, the differences among tasks was exploited mainly in how the second interview should elicit the example and how many examples should be examined in that second interview. In general, if the task is data intensive, then a sequence of examples that built on increasing data complexity was used. In cases where the task included a strong procedural component, the example was used to provide structure, but general rules were expected as part of the outcome of the interview.

3.4 Evaluating the Second Interview

Once a second interview for a task was performed, we again rated the tasks along the set of characteristics. However, this set consisted of 124 characteristics because we broke the training characteristic used in the initial ratings into general education vs. specific training in the operational environment for each of the sixteen tasks. This was due to the impact of a formal training approach on the knowledge acquisition process.

The second rating was performed independently of the ratings given after the first interview. This allowed us to look at how much the ratings changed from one interview to the next, thus providing some indication of how our impression of each task changed. Based on the ratings given for the 124 characteristics for each of the sixteen tasks, the same analyses were performed on the characteristics matrix to see if any significant changes occurred in how the tasks clustered based on the second interview.

The factor analysis revealed three main factors in the questionnaire. The first factor consisted of characteristics associated with tasks involving a physically-based skill, having perceptual requirements, and requiring spatial

and temporal reasoning. The second factor corresponded to knowledge rich tasks involving no perceptual requirements or environmental/psychological stressors. The third factor was related to tasks involving means-ends heuristic search in a data-driven fashion.

As in the first set of analyses, these factors were related to the clustering solution in the cluster analysis. Air traffic control, surgery, and pilot training as well as console operations and medical diagnosis have physical skill and perceptual requirements. The remaining tasks can be characterized as primarily knowledge-based and involving no perceptual requirements or environmental/psychological stressors. The relationship of the third factor to the clustering solution is less clear. Many of the tasks (e.g., medical diagnosis, console operations, and weather forecasting) are data-driven and require some means-ends analysis (e.g., program management and pilot training).

The clusters are fairly similar to those that emerged from the first ratings. Cargo loading, accounting, program management, form fill-out, and drair generation clustered in both solutions as did pilot training and surgery. Weather forecasting and foreign language training never clustered with other tasks. In the second clustering solution, medical diagnosis and console operations clustered, which was not surprising because they both involve diagnosis. There was inconsistency between the cluster from the original ratings of software design and protocol design and the cluster from the second ratings of software design and leadership training. We determined that software design and protocol design are very different tasks. Protocol design involves a knowledge of experimental and statistical methods and is much more formalized than software design. Software design is a true design task, involving decomposition and propose/refine and generate/test problem solving strategies.

Also, the second clustering solution confirmed our contention that two main categories of tasks exist: those formally trained and those not. Three clusters that constituted air traffic control, surgery, pilot training, console operations, medical diagnosis, and weather forecasting include the formally trained tasks. With the exception of cargo loading, the tasks in the other five clusters are not formally trained.

4.0 RESULTS AND CONCLUSIONS

Based on the results of the second interview, the delineation between tasks that are formally taught and those that are learned more or less on the job remained from the first to the second interview. Task clusterings remained very similar from the first to the second interview, though the characteristics themselves were not always rated the same across the tasks. When examining the amount of knowledge we felt that we acquired from an interview and the ease with which it was acquired, the fact that a task was explicitly trained played a tremendous role. Those individuals who had taught, or were teaching, the task were much better prepared to talk about the task in an organized and explicit fashion. They were also much more capable of generating examples and explaining them since this is something that they have had practice in. When the task is not formally trained, the knowledge acquisition process is much more dependent on the skills of the knowledge engineer to guide the interview and make sense of the information collected. This is not always possible with only a couple of hours of discussion with an expert. However, it is also important to be able to detect when the expert does not know enough about the area of interest and to try to find another expert. We had two experts in this effort that, if we were tasked with actually building a tutoring system, would need to be supplemented with other experts more involved with performing the actual task on a day-to-day basis.

Based on this experience, we concluded that, in general, if the task has a distinct starting and stopping point and the task is well-defined, then in the second knowledge acquisition session with an expert, the knowledge engineer should try to go through at least one example. This example should be selected from a continuum of easy to hard, starting with easier ones. What makes the task easier or harder should be identified up front and explained. There are a number of refining conditions as follows:

- If the problem solving task extends over more than a couple of hours, then you should break the task down further into sub-tasks and examine them before proceeding to an example.

- If the task is distinctly procedural, then the example should be used only as a way of eliciting the general procedure and following how it is applied to a specific situation. There is no point in

getting specific information on an example and then having to try to generalize from it when the generalization exists in the expert's head.

·If the task is very data intensive, then a summary of the relevant data elements, where they come from, and how they are used should be discussed before the example is started.

·If the task uses a complex interface such as a console, cockpit, or set of tools, then these should be discussed in terms of their components and their functions before going into an example.

·If the task is formally taught, then the examples should be selected from the early part of the curriculum and discussed as they progress to more complex ones.

·If the task is ill-defined, but a curriculum exists, then use the curriculum to guide the interview process.

·If the task is ill-defined or takes too long to go through an example, then the interview process should continue to explore the task area to determine subtasks that can be examined further.

Thus, the recommendation for the second interview is heavily example-based. However, a word of caution is in order. The examples that we discussed with the experts were not usually easily reduced to a series of attributes and values that could be entered into a knowledge acquisition tool for generalization into rules. The examples often had many aspects and were quite complex. In addition, the examples were always used as a way of structuring the interview, they were not the sole source of knowledge from the interview. The experts provided many insights into the relevance of various attributes and their effect on the problem solving task. In addition, there was often a procedure implicitly used within the process that would not necessarily be apparent just from a few examples or even many hundreds of examples. Thus, just a collection of examples from which we would generalize would be a highly inefficient and ineffective way to acquire knowledge for building a tutoring system or a knowledge-based system. Use of examples can provide a much richer medium for obtaining knowledge than simply the collection of attributes and values as they relate to a solution.

The problem solving taxonomy to support knowledge acquisition developed as a result of this research appears in Figure 1. The first few levels of the taxonomy, including how the task was learned by the expert, the complexity/data requirements of the task, and the continuity with which the task is performed, are areas not addressed previously in any other research. Other attributes of tasks, such as procedural orientation and construction vs. classification, have been addressed by other researchers. However, we believe these latter attributes only affect the knowledge acquisition process in the later stages of an intelligent systems development effort.

This research provided an efficient and effective method of approaching the first few interviews in the knowledge acquisition process. The total time required of an expert in this methodology for the first two interviews ranged from two to four hours. A knowledge engineer using the initial interview and classification scheme should be able to readily classify the task according to the hierarchy. This should provide some heuristics for conducting follow-on interviews, selecting tools, and selecting problem solving methods. In addition, many of the characteristics assessed by the initial interview, while not related to knowledge acquisition, provide input to other design decisions in the development of an ITS or knowledge-based system.

This study generated a number of hypotheses about how to conduct knowledge acquisition for a variety of tasks and therefore provides some direction in how knowledge acquisition should proceed. However, the hypotheses need to be verified through more formal experimentation than was possible in this initial effort.

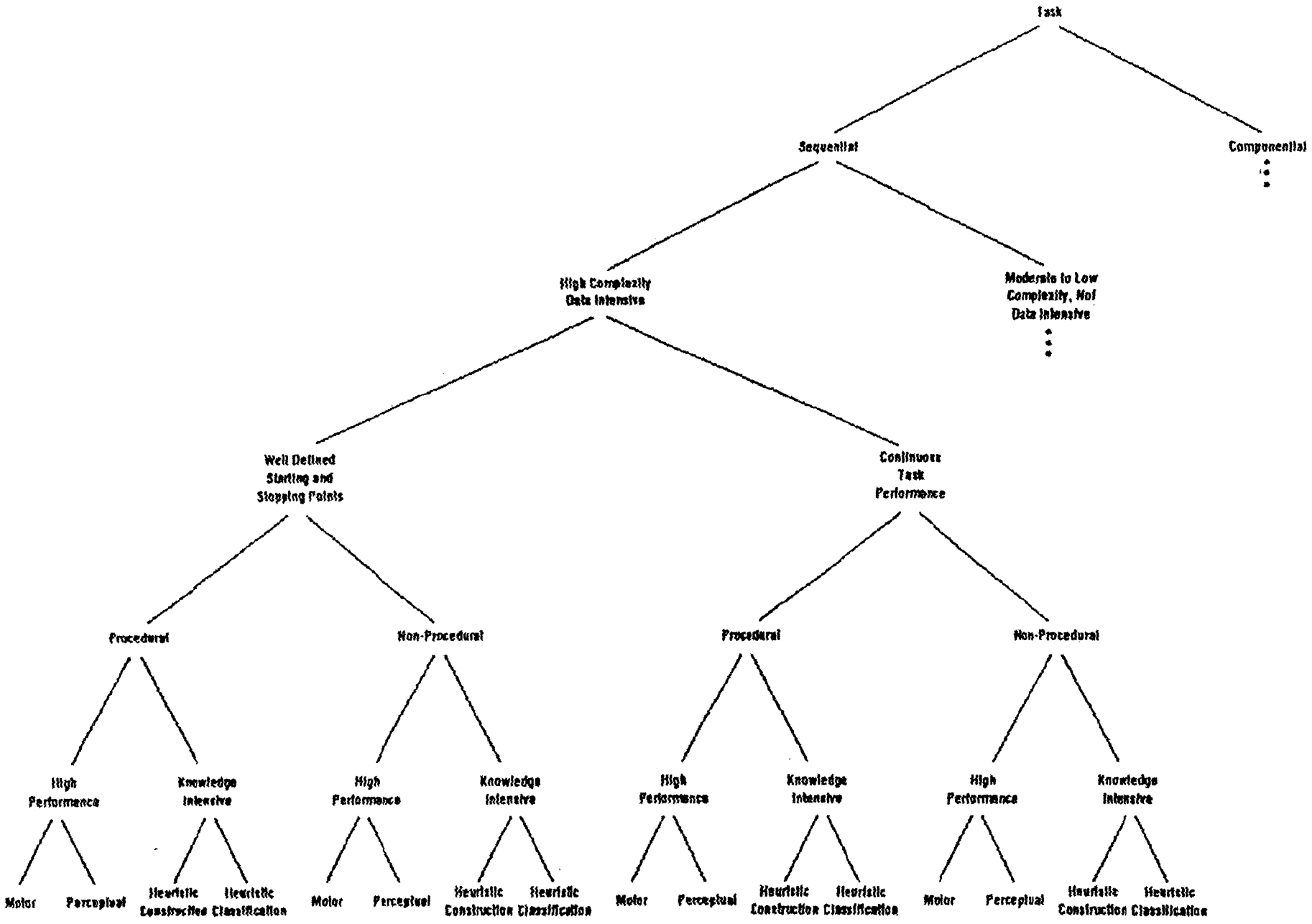

Figure 1. The problem solving taxonomy to support knowledge acquisition

REFERENCES

(1) Boose, J. H. and Bradshaw, J. M. (1987). Expertise transfer and complex problems: Using AQUINAS as a knowledge acquisition workbench for expert systems. International Journal of Man-Machine Studies, 26, 21-25.

(2) Bylander, T. and Chandrasekaran, B. (1987). Generic tasks for knowledge based reasoning: the "right" level of abstraction for knowledge acquisition. International Journal of Man-Machine Studies, 26, 231-244.

(3) Chandrasekaran, B. (1985). Generic tasks in knowledge based reasoning: Characterizing and designing expert systems at the "right" level of abstraction. Proceedings of the Second Conference on Artificial Intelligence Applications, Miami Beach Florida, New York: IEEE Computer Society, 294-300.

(4) Gawron, V. J., Drury, C. G., Czaja, S. J., and Wilkins, D. M. (1989). A taxonomy of independent variables affecting human performance. International Journal of Man-Machine Studies, 31, 643-672.

(5) McDermott, J. (1988). Preliminary steps toward a taxonomy of problem-solving methods. In S. Marcus (Ed.), Automating Knowledge Acquisition for Expert Systems. New York: Kluwer Academic Publishers.

(6) Yost, G. and Newell, A. (1989). "A Problem Space Approach to Expert System Specification," Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pp. 621-627.