*Int. J. Human–Computer Studies* (1996) **45,** 381–396

# Human-centered knowledge acquisition: a structural learning theory approach

David P. Hale and Shane Sharpe

*Area of Management Information Systems, Department of Management Science and Statistics, College of Commerce and Business Administration, University of Alabama, Tuscaloosa, AL 35487-0226, U.S.A. email: ssharpe@alston.cba.ua.edu*

Dwight A. Haworth,

*Department of Decision Sciences, College of Business Administration, University of Nebraska, Omaha, Omaha, Nebraska 68182-0001, U.S.A. email: haworth@cwis.unomaha.edu*

*(Received 31 August 1994 and accepted in revised form 27 May 1996)*

This paper develops the application of structural learning theory (SLT) to support the knowledge engineer (KE) in the knowledge acquisition process and the development of expert systems. The underlying research focuses on the knowledge to elicit from skilled domain problem solvers, and the *structure* (i.e. form and type) of this knowledge using SLT to guide elicitation and interpretation. SLT explicitly models both declarative and procedural knowledge, while presuming an innate backward-chaining mechanism.

Guidelines based on SLT allow knowledge engineers to concentrate on the human-centered knowledge of domain specific problem solvers. In fact, the SLT model presumes that skilled problem solvers do not automatically divulge all rules. This human-centered, needs-based approach provides a point of departure from previous knowledge acquisition methods and serves as a distinguishing feature of this knowledge acquisition method. Specifically grounded in SLT, distinct rule types are developed to be extracted from skilled domain problem solvers. Based on these rule types, guidelines are developed to aid the KEs in the acquisition process.



## 1. Introduction

As the initial phase of rule-based expert system development, *knowledge acquisition* has been identified as the root of many failures, disappointments, and aborted efforts (Alexander, 1984; Buchanan, 1986; Simon, 1987; Coats, 1988; Keyes, 1989) associated with the development and overall performance of knowledge-based systems. In fact, so dominant are these failures that the term *the knowledge elicitation bottleneck* (Hayes-Roth, Waterman, & Lenat, 1983; Garg-Janardan & Salvendy, 1988) has been coined. Hayes-Roth *et al.* (1983) recognize the knowledge acquisition problem as they list the many categories of knowledge that must be elicited from experts—the list is long, diverse, and suggests that integration is not trivial. In addition, Garg-Janardan and Salvendy (1988: p. 378) note that



1071-5819/96/100381 + 16$18.00/0 

> "... there is seldom any emphasis on systematically defining the knowledge to be elicited prior to eliciting it. This results in the acquisition of *ad hoc* subsets of knowledge."

Responding to these issues, several knowledge representation methods have been developed (Hayes-Roth *et al.*, 1983; Scott, Clayton & Gibson, 1991; Rosenbloom, Laird & Newell, 1993; Schreiber, Wielinga & Breuker, 1993). To enable a knowledge engineer to mitigate personal biases, the need for formalized knowledge acquisition methods has been suggested. Garg-Janardan and Salvendy (1987) assert the need for such a theoretical underpinning in any knowledge acquisition method, and this perspective is shared by Sharman and Kendall (1988) as they indicate a need to understand and incorporate cognitive psychology principles to support knowledge acquisition. Other researchers (Hayward, Wielinga & Breuker, 1987; Garg-Janardan & Salvendy, 1988) share similar conclusions concerning the dominant need to utilize an integrated approach for eliciting, abstracting, and formalizing domain knowledge.

Shaw and Woodward (1990) detail an approach to the knowledge acquisition challenge that uses a series of models to depict the translation of knowledge from domain experts to formal system constructs. Grounded in the psychology and linguistics literature, their approach clearly supports the perspective of a knowledge engineer attempting to communicate with domain experts. Based on a dialogue between knowledge engineers and domain experts, a knowledge engineer is required to develop a *communication model* of the domain. Then the knowledge engineer is required to transform the communication model into an *intermediate knowledge base* by extracting primitives from the communication model and enforcing a structure on this knowledge. However, Shaw and Woodward's (1990) approach still relies on the knowledge engineers' subjective interpretations of extracted primitives, which are influenced by prejudices that may be manifested while eliciting knowledge, informally extracting primitives, or due to prematurely selecting a particular physical implementation structure.

A review of the relevant literature reveals a means to mitigate the subjectivity associated with these interpretations; thus enabling knowledge engineers to support the acquisition, interpretation, and formalization of knowledge through a comprehensive set of procedures and modeling constructs. The approach provides support for the identification of the task situation characteristics and the domain-independent processes used by the domain expert to address the task. The approach is grounded in Structural Learning Theory (SLT) (Scandura, 1964, 1971, 1977, 1984), which has been employed by a diverse set of disciplines including: cognitive psychology (Jeeves & Greer, 1983; Foshay, 1989; Scandura & Dolores, 1990), developmental and educational psychology (Anders, Fozard & Lillyquist, 1972; Papert, 1975; Shavelson & Geeslin, 1975; Manelis & Yekovich, 1984), mathematics (Dienes, 1972), and linguistics (Greer, 1974). Based upon this body of research, this paper focuses on the problem-solving knowledge from skilled domain problem solvers in a form and structure matching the problem solver's own processes† rather than focusing on a technique centered on knowledge engineers and AI implementation methods.

† This approach has been validated through the construction of several system implementations.

## 2. SLT and knowledge elicitation

SLT describes the knowledge required for human problem solving and the basic processes to manipulate this knowledge. The guidelines for eliciting and structuring knowledge in a manner similar to the expert's own problem-solving process are developed here using SLT. These guidelines promote a smooth cognitive transition between the acquisition of an expert's knowledge and the organization of the knowledge without an intermediate knowledge representation structure. SLT provides the KE with a compendium of the guidelines to elicit rule-based knowledge. SLT explicitly recognizes the general procedural knowledge to support the overall problem-solving process, declarative knowledge associated with both problem situation and goal conditions, and application procedural knowledge. This is consistent with prior research efforts (Garg-Janardan & Salvendy, 1987; Byrd, Cossick & Zmud, 1992) that support the elicitation of both declarative and procedural knowledge from an expert for the development of an expert system.

### 2.1. GENERAL PROCEDURAL KNOWLEDGE

General procedural knowledge is associated with the actual execution and evaluation of rules.† The SLT's general procedural approach presumes a multiple level backward-chaining control mechanism, which humans demonstrate at an early age (Scandura, 1977: p. 42), as its fundamental procedure. Further empirical evidence supports claims that this approach is continued throughout an individual's life (Scandura, 1981; Jeeves & Greer, 1983; Manelis & Yekovich, 1984; Foshay, 1988). Unlike other control mechanisms (e.g. Newell & Simon, 1972), SLT does not automatically assume that people can compose available rules whenever mathematically possible, nor is composition the only means of developing new rules from older rules. Moreover, based on Dreyfus and Dreyfus' (1986) categorization of human expertise, both skilled problem solvers (that is, those who exhibit competence or proficiency in cognitive tasks) and experts (that is, those who exhibit superior performance in cognitive tasks) use a structured approach, which is consistent with Scandura's assertion that a knowledge representation must be correlated with an individual's level of expertise (Scandura, 1977: p. 111). This leads to the graphical interpretation of SLT's General Procedural Knowledge Structure, as presented in Figure 1.

The *Find Rule, Switch Goal* and *Execute Rule* are the primary general procedural knowledge activities dictated by SLT. Upon initiation of the *Solve Problem* process, the *Find Rule* activity searches for rules with the potential to satisfy the goal in a given problem situation. If a rule is found, then *Execute Rule* invokes the rule. If execution results in the goal being met, the *Solve Problem* process successfully terminates. In situations where the *Find Rule* activity does not find a candidate rule to be executed, the *Switch Goal* control mechanism is activated, which defines a strategy to solve the problem (e.g. a strategy designed for information gathering to enhance the known information pertaining to the problem situation) and then invokes the *Solve Problem* process recursively. On completion of the recursively-invoked *Solve Problem* process, a check is made to determine whether or not the

† Note that the term *rule,* used in the context of SLT, is a complex structure (shown in Figure 2) that is a superset of the IF–THEN structure of a conventional rule-based expert system.

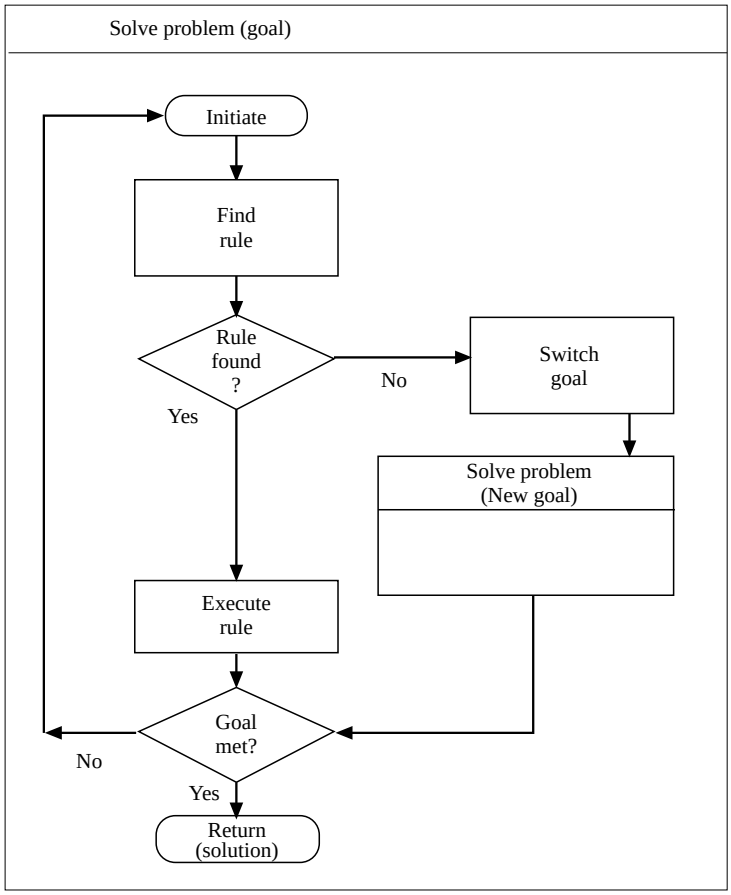


FIGURE 1. General Procedural Knowledge Structure.

original goal has been met. If the original goal has not been met, another iteration of the *Solve Problem* commences. This leads to the first two guidelines suggested by SLT to aid knowledge engineers in acquiring and structuring knowledge from human problem solvers, based on how humans solve problems.†

**Guideline 1:** Knowledge engineers must identify the general problem solving structure of *Solve Problem*: *Find Rule, Switch Goal,* and *Execute Rule.*

**Guideline 2:** Knowledge engineers must identify changes in goal and the recursive invocation of the *Solve Problem* process.

### 2.2. DECLARATIVE KNOWLEDGE

Declarative knowledge is necessary to invoke the SLT *Solve Problem* process. Specifically, the problem-solving process requires a description of the current problem situation and goal condition(s). This declarative knowledge is further subdivided into knowledge associated with the problem states and goal conditions, which leads to guidelines 3 and 4.

† Appendix 1 contains all 16 guidelines presented in this paper.

**Guideline 3:** Knowledge engineers must elicit information concerning the initial problem state to facilitate the development of the problem domain.
**Guideline 4:** Knowledge engineers must elicit information concerning the initial goal state to determine the goal conditions that must be satisfied.

Specifically, the KE should obtain information regarded by the expert as significant information about the initial problem and goal state and the problem solver's interpretation of the problem and goal state.

Throughout the general procedural knowledge activities, rules are triggered based on conditions related to a problem state and the desired goal state. Consequently, guidelines 5 and 6 suggest that knowledge engineers determine from human problem solvers information concerning the necessary conditions for triggering each rule to be invoked and the consequences of invoking each rule.

**Guideline 5:** Knowledge engineers must elicit information concerning the necessary conditions needed to trigger each rule's invocation.
**Guideline 6:** Knowledge engineers must elicit information concerning the resultant effects of invoking each rule.

In accordance with SLT, rules provide the structure to retain domain-specific information in three components (Scandura, 1977: p. 154) as follows.

- The domain, which describes problem situations to which the rule is applicable.
- The range, which gives the outcomes of the rule's execution.
- The process, which defines the tasks that the rule performs to deliver a desired goal.

Figure 2 depicts a representation of this underlying declarative knowledge of the rule structure. The *domain-process-range* structure of a rule represents lists comprised of the following.

(1) Problem situations: incongruences between what is expected and what occurs;

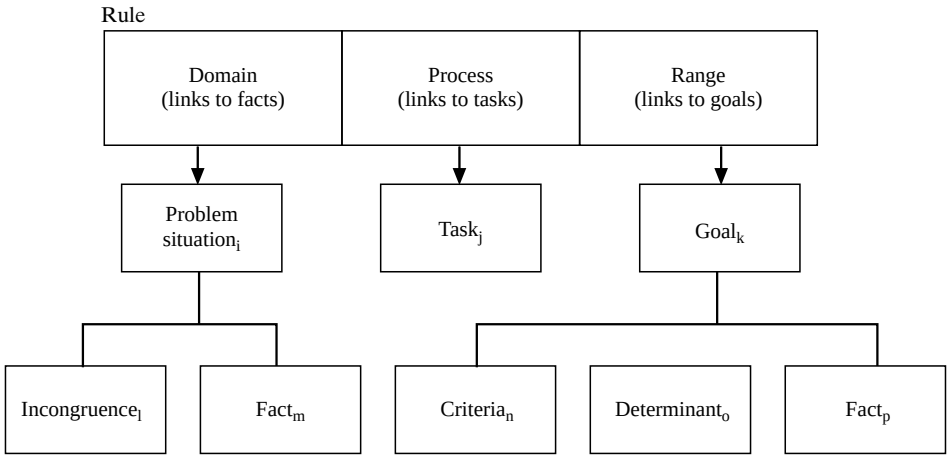


FIGURE 2. SLT Rule Structure.

and facts that specify the inputs required for application of a rule to a given problem state.

(2) Tasks: operators to describe the executable process steps (i.e. either: sequences of operations; conditional branching steps; or an iterations) that specify a domain-specific technique, systematic method, or algorithm that operates on the structure in the rule's problem situation (i.e. domain) to generate the desired output (i.e. rule's range).

(3) Goals: criteria that specify the conditions to be satisfied by the rule's process; determinants that add precision to the goal's criteria by specify the operands in units that correspond to the units specified in the problem situation's incongruences; and facts that provide characteristics of the goal criteria and determinants such as robustness and quality.

The rule's declarative knowledge structure serves as a template to guide the knowledge engineer in the acquisition and structuring of problem-solving knowledge. In relation to the structure of a rule, the *Find Rule* activity involves the mapping of the *domain* of a rule to the current problem situation and the range of the rule to the desired goal state, with the result being the identification of candidate *directive rules*.

**Definition:** a **directive rule type** is invoked by the *Execute Rule* activity to produce a deterministic procedure, given a specific goal state and problem situation. The result of executing a directive rule will, at a minimum, modify the problem situation facts, which may lead to a change in the incongruences.

Moreover, knowledge engineers can represent multiple perspectives of a problem state through varying the domain of rules (i.e. the parameters associated with the incongruences and facts) while holding the goal state constant. Likewise, multiple solution techniques to a problem situation can be included in a knowledge base by having multiple rules that duplicate the same problem situation but uniquely represent the task components. Multicriteria decision making can be accommodated by listing multiple goals (each of which has its own unique criteria, determinants and facts) within a rule. Thus, guideline 7 suggests that KEs elicit information concerning multiple perspectives, solution methods, and criteria.

**Guideline 7:** Knowledge engineers must elicit information concerning multiple views and generate the appropriate set of rules that differ by problem situation, task, and goal.

To accomplish the defining of perspectives requires the classification of directive rule types into two sub-types: problem identification rules and direct solution rules (Table 1).

The *problem definition* rule type is essential for the formulation of a problem and is a necessary requisite to demonstrate minimal competence in the performance of a task. The problem definition rule type operates on problem descriptions (i.e. initial input to the problem-solving process) to transform them into the appropriate form for use with the available rules. Thus, problem definition rule types are essential

TABLE 1
*Directive rule types*

| Rule Type | Rule Function |
|---|---|
| Problem definition rule | A rule that operates on problem descriptions (i.e. the initial input into the problem-solving process) to transform them into the appropriate form for use with the rules available. These rules are essential for the formulation of a problem. |
| Direct solution rule | A rule that contains elements of the given problem situation in its domain and the goal in its range. A problem is solved if the output generated through the execution of the rule satisfies the primary goal of the problem. |

for encoding the environmental variables required for the development of the initial problem and goal states. For example, a sample problem scenario states,

> ''given the enclosed floor layout, the current machine configurations, and the specifications of the machine configurations for the new product manufacturing, what would be the setup cost to enable the changeover?''

The application of a problem definition rule type to this scenario results in the encoding of the input variables (e.g. floor layout, current machine configurations, desired machine configurations) and the desired output of the problem solving effort (i.e. determination of setup costs) into a format appropriate for problem solving. In this example, the encoding of machine configurations may include its physical dimensions and the time required to make necessary configuration changes, which lead to guideline 8.

**Guideline 8:** The development of problem definition rules require the KE to elicit information concerning the transformation of input variables into an appropriate format for processing.

According to SLT, the second basic type of directive rule is the *solution rule*. Rules of the solution rule type contain elements of the current problem situation in its domain and the goal state in its range. The *Find Rule* activity identifies candidate solution rules by matching the rule's domain and range to the given problem and goal situation. The process component of the rule is then fired by the *Execute Rule* activity. A problem is solved if the output generated through the execution of the rule satisfies the given goal conditions. Continuing the above example, the explication of an expert's algorithm for determining the most efficient production facility setup, given a specified set of parameters and constraints is an example of the procedural component of a direct solution rule type, which leads to guideline 9.

**Guideline 9:** The KE must elicit information concerning the procedure associated with a solution attempt.

The general procedural knowledge provides the foundation for rule management by associating elicited procedural knowledge (i.e. executable task steps) with declarative knowledge about the *domain* (i.e. problem situation) *and range* (i.e. goal) of a rule.

### 2.3. ELICITING APPLICATION PROCEDURAL KNOWLEDGE

Though the general procedural knowledge provides the foundation for rule management, the acquisition of application procedural knowledge is a major concern for rule development. SLT provides direction for acquiring application procedural knowledge by denoting specific rule types that contain this knowledge. The guidelines for the development of specific application procedural knowledge further serve to enhance the development of the expert system knowledge base; that is, in accordance with SLT, an expert that fails to provide an immediate solution to a problem through the application of a directive rule invokes the *Switch Goal* activity to establish a new *secondary goal*.

**Definition:** a **secondary goal** is established by the *Switch Goal* activity. It is intended to enhance the expert's comprehension of the problem or goal state.

The representation of the expert's problem-solving episode enables the KE to identify the specific type of rule employed during this process. Table 2 presents a compendium of *comprehension* rule types that are useful in the realization of secondary goals.

**Definition:** a **comprehension rule** type provides the ability to identify missing information and to decompose a complex problem into related sub-problems.

An often-used means to structure the problem solving process is decomposition. SLT's construct corresponding to decomposition is the *planning rule* type. By definition, a *planning rule* acts on the problem and goal situation to decompose it into a number of related subgoals resulting in a *solution plan*.

**Definition:** a **solution plan** is a collection of sub-goals, each of which have to be solved by the entire *SOLVE PROBLEM* process.

For any given problem, there may be a number of different plans, each plan calling

TABLE 2
*Comprehension rule types*

| Rule type | Rule description |
|---|---|
| Planning rule | A rule that acts on the problem/goal situation set(s) to reduce it to a sequence of sub-problems (solution plan). These rules provide a means of decomposing the problem into simpler, more manageable sub-problems. |
| Information rule | A rule that searches existing rules to determine what needed information is missing from the problem/goal situation and creates a solution plan for obtaining the missing information. Thus, when executed, an information rule acquires additional knowledge about the problem/goal situation. |
| Combination rule | A rule that provides for the exploitation of a pattern in one domain to be used in another domain. This provides the ability to transfer problem-solving capabilities between problem domains. |

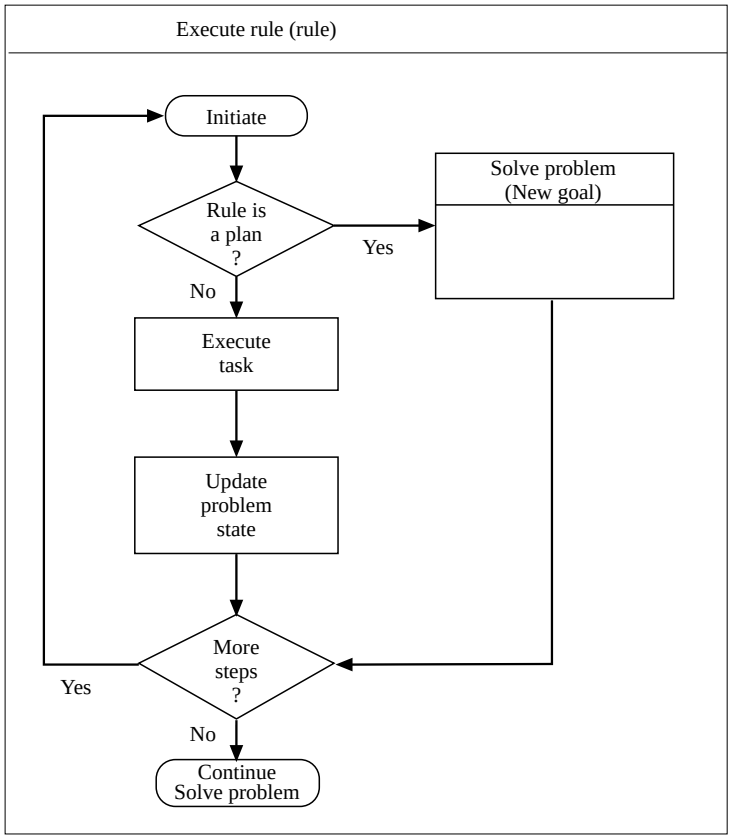


FIGURE 3. Details of Execute Rule.

on different directive rules to generate solutions to the corresponding sub-goals. The application of a solution plan and the process of developing and evaluating sub-goals is provided in Figure 3. The *Execute Rule* activity executes the tasks in a rule in order to generate the desired outcome (i.e. the range) of the rule. The actual outcome of the rule updates the problem situation (i.e. incongruences and facts).

The first task within the *Execute Rule* activity determines whether the rule to be executed is a plan. If the rule to be executed is not a plan, (e.g. the rule is a directive, information or combination rule type) the rule's process-task component is executed. The result of executing the task updates the problem state facts (i.e. problem situation facts and goal facts), which consequently provides for the problem situation incongruences to be re-evaluated; likewise, executing the task can update the facts associated with the rule's goal, which consequently can refine the goal's criteria and determinants. Once the facts have been updated, *Execute Rule* terminates.

In situations where the rule to be performed in *Execute Rule* is a plan,† the

† For example, a planning rule can decompose the setup cost formulation into the determination of costs associated with future floor layout and machine specifications constrained by the current setup of the floor, machine, and staff.

procedural elements of the plan are successively invoked. That is, each stage of the plan becomes a problem situation that, in turn, is solved. Consequently, knowledge engineers must be aware of the recursive nature of problem solving and apparent switches in the immediate goal that the problem solver is attempting to solve (guideline 2). Once the entire plan has been executed, the *Execute Rule* activity that was invoked by the plan terminates. Thus, the *planning* rule type leads to guideline 10.

**Guideline 10:** The development of planning rules requires the KE to elicit information related to the decomposition of problem situations and sub-goals.

Additional evaluation of an expert's elicited problem-solving knowledge leads to the development of other rule types involved in problem-solving activities. An *information rule* is used to enhance the information about the current problem situation or goal state. The application of an *information rule* results in the use of a specific information search or problem comprehension strategy to obtain additional information about the current problem situation or goal state, or to clarify the existing information. The enhanced information is then applied to the problem-solving effort to identify specific directive rules that achieve a solution to the problem. In the example of the expert determining setup costs for a production changeover, the expert may evaluate the problem and determine that there are variables not included in the problem situation or goal state required for problem resolution. Factors such as labor availability, wage and estimated productivity rates may be required for the calculations associated with a specific solution alternative and may need to be retrieved. In addition, some factors such as the units associated with retooling productivity indexes may need to be clarified or modified before they can be included in the calculations. Guidelines 11 and 12 suggest that KEs should elicit from problem solvers information concerning how they resolve missing or fuzzy data.

**Guideline 11:** The KE should elicit information related to the comparison of an expert's solution alternatives to the current information content of the problem situation and goal state for the determination of missing information or information that requires clarifying.

**Guideline 12:** The KE should elicit information related to the development of a plan to obtain missing information or to clarify information.

The ability to transfer knowledge between problem domains enhances the current problem-solving effort and is consistent with the problem-solving process of a human expert (Newell & Simon, 1972). An evaluation of the elicited knowledge leads to the development of a *combination rule*. The *combination rule* type acts on rules from other problem domains to facilitate the application of these rules to the current problem domain. For example, an expert may use rules from the domain of construction project management for the calculation of idle worker costs and may invoke a rule from the petroleum refinery maintenance domain to calculate lost production costs during the production facility changeover period. Thus, guideline 13 suggests that KEs must elicit information concerning patterns from one domain that can be reused in another.

TABLE 3
*Coordination rule types*

| Rule type | Rule description |
|---|---|
| Selection rule | A rule that discriminates between two or more rules and orders the rules for execution. This provides for the selection of the ''better'' rule when two or more rules fit the problem situation and goal. |
| Derivation rule | A rule that operates on rules (in or as structures) and derives new rules. The application of this rule to directive and comprehension rule types results in new rules capable of processing more efficiently, and capabilities required for developing adaptive expert systems. |

**Guideline 13:** The development of combination rules requires the KE to elicit information concerning the rules associated with the structurally related problem domains.

To assist the KE in acquiring application procedural knowledge, the SLT model directs that each *comprehension* rule type (e.g. planning, information and combination rule types) is invoked as a result of not finding a directive rule in the *Find Rule* activity. Consequently, a goal switch is made to increase problem state comprehension.

In addition to the comprehension rules, the SLT model provides additional guidance for the elicitation of application procedural knowledge required for the development of an expert-system knowledge base. This guidance is specifically directed at the development of coordination rule types that are essentially rules that act on other rules. Table 3 depicts the two major types of coordination rule types. These rules essentially increase the efficiency of rule processing.†

The *selection rule* operates on two or more rules that may be applied to a given problem situation and orders the candidate rules based on a selection criterion. In both the *Find Rule* and *Switch Goal* activities, the *selection rule* type is applied. In the *Find Rule* activity, candidate direct solutions must be evaluated to determine which one will be chosen as the next to be attempted. In the *Switch Goal* activity, a choice is made among the most appropriate rule from a related set of comprehension or coordination rules.

Due to the performance issues, guidelines 14 and 15 suggest that KEs elicit rule ordering information from experts.

**Guideline 14:** The knowledge engineer must elicit information concerning the criteria for evaluating, ordering, and determining the appropriateness of specific candidate directive rules.

**Guideline 15:** The knowledge engineer must elicit information concerning the criteria for evaluating, ordering, and determining the appropriateness of specific candidate comprehension and coordination rules.

† It has been the authors' experience that coordination-level knowledge is difficult to elicit from experts using conventional knowledge acquisition techniques and that these guidelines facilitate the ability to deliberately elicit this type of knowledge.

If an expert struggles to comprehend a problem situation after the application of previous problem comprehension rules, the expert may need to develop a means to comprehend the problem situation. The application of a *derivation rule* acts on two or more rules to derive new rules—this rule type has also been labeled as the *composition rule type* (Sharpe, Haworth & Hale, 1996). These new rules may result in the development of a new coordination rule capable of enhancing the expert's current representation of the problem situation or goal state. Thus, guideline 16 provides a mechanism for adaptive expert systems to be created using information elicited from experts.

**Guideline 16:** The knowledge engineer should elicit information related to the expert's ability to develop new procedures to identify or clarify information.

In addition to the aforementioned coordination rules, SLT provides direction in the creation of additional rule types for the development of a knowledge base (Scandura & Dolores, 1990). For example, upon reflection of the problem-solving effort, an expert may employ specific rules similar to a derivation rule to develop new rules that are more efficient in light of the overall problem-solving process and may enhance future problem-solving efforts. The capability to develop more efficient knowledge structures is empirically grounded in structural learning research efforts (cf. Scandura, 1977, 1984). This capability also provides a foundation for the development of expertise and building adaptive expert systems capable of knowledge base enhancement. Although this is an interesting extension to the guidelines for knowledge elicitation directions pertaining to knowledge base development, the description and assessment of these rules is beyond the scope of this paper.

## 3. Summary and conclusion

This paper develops a model of human problem solving founded upon structural learning theory which is capable of accounting for processes and knowledge structures utilized by individuals demonstrating varying levels of problem-solving competence. This model serves as the basis for the development of a set of guidelines to support the knowledge engineer in the knowledge acquisition process. SLT depicts the knowledge required for human problem solving and presents the basic structures and processes to manipulate this knowledge.

In summary, SLT provides for the integration of declarative and procedural knowledge within a single structure, the rule. The declarative knowledge resides in the domain of the rule as the set of givens in the problem space, and in the range of the rule as the set of outcomes from the rule's application. Empirical studies conducted by psychologists and human development scholars (Scandura, 1977; Jeeves & Greer, 1983) have discovered SLT corresponds to how human experts describe their problem solving approach.†

SLT serves to guide the KE in structuring the knowledge elicited from a skilled

† Through concurrent protocol analysis, Sharpe (1992) mapped human experts' task activities and found a significant correlation to SLT's representation of the task, which included *Find Rule, Switch Goal* and *Execute Rule* activities.

problem solver into the various rule structures. The development of the distinct rule types further supports the KE in structuring the interactions. Tables 1, 2 and 3 provide an overview of the various rule types required for the development of an expert system, and SLT (as represented in Figures 1 & 3) guides the KE in developing the hierarchy of rule types and the interaction between rules. In addition to providing support for the classification and separation of the various rule types, SLT allows the structure of the problem to be identified and generalizations to be made across problem-solving tasks and domains. This is especially helpful in the development of complex direct solution rules and the various complex rule types.

The SLT method is distinguished from previous knowledge acquisition methods in that the focus is on the skilled problem-solver rather than the knowledge engineer or an artificial intelligence development technique. This human-centered approach provides a means to capture and structure the processes and knowledge employed by a skilled problem-solver that is appropriate for the nature of the application. That is, this approach recognizes the differing levels of competence possessed by individuals, and specifies the processes and declarative knowledge they use to solve problems. The recognition of varying levels of competence is coupled with the requirements of the expert system to provide a design that is commensurate with the level of proficiency required by the application. The human-centered approach of knowledge acquisition based on the SLT model is further accentuated in that the model does not automatically assume that people can compose available rules simply because it is mathematically possible, nor is composition the only means of developing new rules from older rules. Recognition of this human trait allows the knowledge engineer to perform a structural analysis of the rule base and devise rules to complete the knowledge base in accordance of the application requirements.

Because SLT is based on a deterministic view of problem solving, it offers useful insights for implementing rule-based expert systems with backward-chaining processes. The SLT structures are readily implementable in an automated form and allow for characterizations and organizations that support efficiency in problem solving, which is equated with expertise. The technique is based on a model of human problem solving, and, therefore, does not require the development of an intermediate representation between the knowledge acquisition, nor does it rely on the knowledge engineer's interpretation of how to frame the domain expert's knowledge. The use of SLT allows the knowledge engineer to make a smooth cognitive transition between the acquisition of an expert's problem-solving knowledge and the structuring of a rule-based expert system. Thus, these guidelines provide assistance to the KE to systematically define the knowledge to be elicited prior to elicitation. This results in the acquisition of the cover set of knowledge needed to resist the knowledge engineering bottleneck.

## References

ALEXANDER, T. (1984). Why computers can't outthink the experts. *Fortune*, **110,** 105–118.

ANDERS, R. R., FOZARDM, J. L. & LILLYQUIST, T. D. (1972). The effects of age upon retrieval from short-term memory. *Developmental Psychology,* **6,** 214–217.

BUCHANAN, B. G. (1986). Expert systems: working systems and the research literature. *Expert Systems*, **3,** 32–51.

BYRD, T. A., COSSICK, K. L. & ZMUD, R. W. (1992). A synthesis of research on requirements analysis and knowledge acquisition techniques. *Management Information Systems Quarterly*, **16,** 117-138.

COATS, P. K. (1988). Why expert systems fail. *Financial Management*: *Journal of the Financial Management Association*, **17,** 77–86.

DIENES, Z. P. (1972). Some reflections on learning mathematics. In W. E. Lamon, Ed. *Learning and the Nature of Mathematics.* Chicago, IL: Science Research Associates.

DREYFUS, H. L. & DREYFUS, S. E. (1986). *Mind Over Machine*: *The Power of Human Intuition and Expertise in the Era of the Computer*. New York: Free Press.

EDWARDS, J. S. (1991). *Building Knowledge-Based Systems*. New York: Halsted Press.

FOSHAY, W. R. (1989). What we know (and what we don't know) about training of cognitive strategies for technical problem-solving. *Journal of Structural Learning*, **10,** 111–125.

GARG-JANARDAN, C. & SALVENDY, G. (1987). A conceptual framework for knowledge elicitation. *International Journal of Man-Machine Studies*, **26,** 521–531.

GARG-JANARDAN, C. & SALVENDY, G. (1988). A structured knowledge elicitation methodology for building expert systems. *International Journal of Man-Machine Studies*, **29,** 377–406.

GREER, G. B. (1974). Sets, logic and concepts: differences in rule difficulty. *Journal of Structural Learning,* **4,** 143–163.

HAYES-ROTH, F., WATERMAN, D. A. & LENAT, D. B. (1983). *Building Expert Systems*. Reading, MA: Addison-Wesley.

HAYWARD, S., WIELINGA, B. & BREUKER, J. (1987). Structured analysis of knowledge. *International Journal of Man-Machine Studies*, **26,** 487–498.

KEYES, J. (1989). Why expert systems fail. *AI Expert*, **4,** 50–53.

JEEVES, M. A. & GREER, G. B. (1983). *Analysis of Structural Learning*. New York: Academic Press.

MANELIS, L. & YEKOVICH, F. R. (1984). Analysis of expository prose and its relation to learning. *Journal of Structural Learning*, **8,** 29-44.

NEWELL, A. & SIMON, H. A. (1972). *Human Problem Solving*. Englewood Cliffs. NJ: Prentice Hall.

PAPERT, S. (1975). Teaching children thinking. *Journal of Structural Learning,* **4,** 219–229.

ROSENBLOOM, P. S., LAIRD, J. E. & NEWELL, A. (1993). *The Soar Papers*: *Research on Integrated Intelligence*. Vol. 1 and 2. Cambridge, MA: MIT Press.

SCANDURA, J. M. (1964). Analysis of exposition and discovery modes of problem solving instruction. *Journal of Experimental Education,* **33,** 149–159.

SCANDURA, J. M. (1971). Deterministic theorizing in structural learning: three levels of empiricism. *Journal of Structural Learning,* **3,** 21–53.

SCANDURA, J. M. (1977). *Problem Solving: A Structural/Process Approach with Instructional Implications.* New York: Academic Press.

SCANDURA, J. M. (1981). Problem solving in schools and beyond: transitions from the naive to the neophyte to the master. *Educational Psychologist*, **16,** 139-150.

SCANDURA, J. M. (1982). Structural (cognitive task) analysis: a method for analysing content. Part I: background and empirical research. *Journal of Structural Learning*, **7,** 101–114.

SCANDURA, J. M. (1984). Structural (cognitive task) analysis: a method for analyzing content. Part II: toward precision, objectivity, and systematization. *Journal of Structural Learning*, **8,** 1–28.

SCANDURA, J. M. & DOLORES, J. (1990). On the representation of higher order knowledge. *Journal of Structural Learning*, **10,** 261–269.

SCHREIBER, A. T., WIELINGA, B. J. & BREUKER, J. A. (Eds.) (1993). *KADS: A Principled Approach to Knowledge-Based Systems Development,* London: Academic Press.

SCOTT, A. C., CLAYTON, J. E. & GIBSON, E. L. (1991). *A Practical Guide to Knowledge Acquisition*. Reading, MA: Addison-Wesley.

SHARMAN, D. & KENDALL, E. (1988). A case study: acquiring strategic knowledge for expert system development. *IEEE Expert*, **3,** 32–40.

SHAVELSON, R. J. & GEESLIN, W. E. (1975). A method for examining subject-matter structure in instructional material. *Journal of Structural Learning,* **4,** 199–218.

SHARPE, R. S (1992). *A structural learning theory approach to problem solving: an investigation in software maintenance.* Ph.D. dissertation, Texas Technology University, TX, U.S.A.

SHARPE, S., HAWORTH, D. A. & HALE, D. P. (1996). Beyond integrity constraints: business rules enable workflow flexibility and coordination. *The Journal of Systems Management*, **47,** 52-56.

SHAW, M. L. & WOODWARD, J. B. (1990). Modeling expert knowledge. *Knowledge Acquisition*, **2,** 179–206.

SIMON, R. (1987). The morning after. *Forbes*, **140,** 164–168.

Paper accepted for publication by Associate Editor, Dr. A. Rappaport.

## Appendix A

GUIDELINES FOR KNOWLEDGE ACQUISITION

**Guideline 1:** Knowledge engineers must identify the general problem solving structure of *Solve Problem*: *Find Rule, Switch Goal* and *Execute Rule*.

**Guideline 2:** Knowledge engineers must identify changes in goal and the recursive invocation of the *Solve Problem* process.

**Guideline 3:** Knowledge engineers must elicit information concerning the initial problem state to facilitate the development of the problem domain.

**Guideline 4:** Knowledge engineers must elicit information concerning the initial goal state to determine the goal conditions that must be satisfied.

**Guideline 5:** Knowledge engineers must elicit information concerning the necessary conditions needed to trigger each rule's invocation.

**Guideline 6:** Knowledge engineers must elicit information concerning the resultant effects of invoking each rule.

**Guideline 7:** Knowledge engineers must elicit information concerning multiple views and generate an appropriate set of rules that differ by problem situation, task, and goal.

**Guideline 8:** The development of problem definition rules require the KE to elicit information concerning the transformation of input variables into an appropriate format for processing.

**Guideline 9:** The KE must elicit information concerning the procedure associated with a solution attempt.

**Guideline 10:** The development of planning rules requires the KE to elicit information related to the decomposition of problem situations and sub-goals.

**Guideline 11:** The KE should elicit information related to the comparison of an expert's solution alternatives to the current information content of the problem situation and goal state for the determination of missing information or information that requires clarifying.

**Guideline 12:** The KE should elicit information related to the development of a plan to obtain missing information or to clarify information.

**Guideline 13:** The development of combination rules requires the KE to elicit information concerning the rules associated with the structurally related problem domains.

**Guideline 14:** The knowledge engineer must elicit information concerning the criteria for evaluating, ordering, and determining the appropriateness of specific candidate directive rules.

**Guideline 15:** The knowledge engineer must elicit information concerning the criteria for evaluating, ordering and determining the appropriateness of specific candidate comprehension and coordination rules.

**Guideline 16:** The knowledge engineer should elicit information related to the expert's ability to develop new procedures to identify or clarify information.