Proc. of the 1992 IEEE
Int. Conf. on Tools with AI
Arlington, VA, Nov. 1992

Design of a Generic Knowledge Acquisition Shell

Chih-Cheng Chien*+ and Cheng-Seen Ho*

*Department of Electronic Engineering, National Taiwan Institute of Technology
43 Keelung Road, Section 4, Taipei, TAIWAN 106

+Telecommunication Laboratories, Ministry of Transportation and Communications
9 Lane 74, Hsin-Yi Road, Sec. 4, Taipei, TAIWAN

*ABSTRACT The process of knowledge acquisition is technically challenging and time consuming. It is one of the major bottlenecks in the development of expert systems. In this paper we design a Generic knowledge Acquisition Shell (**GAS**) which allows users to easily construct and then execute a domain-specific knowledge acquisition tool. **GAS** contains six modules including a structure editor, a tool generator, a tool interpreter, a knowledge processor, a data management system, and a graphic-based user interface and a primitives kernel including sufficient primitives in problem solving strategy, knowledge representation, and knowledge acquisition techniques. This approach relieves knowledge engineers from selecting and applying different knowledge acquisition tools to different domains. We have also developed **GAS** as an open architecture so that further enhancement can be done easily.*

1. INTRODUCTION

Several approaches have been taken in the research of knowledge acquisition (KA). The interactive KA tools approach tries to make the knowledge acquisition process from people semi- or fully automated through the introduction of knowledge acquisition tools. This approach is interesting because it could make the psychological approach more productive by realizing proper psychological theories into specific KA tools, such as ETS [2]. It also facilitates the development of a framework that involves techniques from both learning and KA tools approaches so that sources for knowledge acquisition can be broadened, such as KRITON [8].

Many KA tools have been developed under a rather unified idea since 1983. Examples include KNACK [12], MOLE [9] MORE [10], ROGET [1],, SALT [14], SIZZLE [16], etc. Although these KA tools have made some progress in solving the knowledge acquisition bottleneck, several problems remain. First, many of the tools are too domain-specific and difficult to be applied to other domains. Secondly, knowledge categories to be extracted by some of the KA tools are too restrictive. Thirdly, integration of knowledge generated by the KA tools with other sources is difficult, because of a different knowledge representation being used in different knowledge bases.. Finally, a graphic-based user interface is important in any software system, but many of the KA tools do not provide the facility.

We design a generic KA shell, ***GAS*** (Generic knowledge Acquisition Shell) which tries to attack these problems to further improve the process of knowledge acquisition. The following sections introduce the design plilosophy, the primitives kernel, the generic KA structure, and the system modules of ***GAS***.

2. GAS'S DESIGN PHILOSOPHY

GAS contains sufficient primitives in knowledge representations, problem solving strategies, and knowledge acquisition techniques so that the construction of a domain-specific KA tool from these primitives becomes a straight forward job. Basically, ***GAS*** was designed with the following features in mind:

a. providing a set of *problem solving primitives* for constructing specific problem solving methods for specialized KA tools;
b. providing a set of *representation schemas* for representing domain models;
c providing a set of *acquisition primitives* for acquiring domain knowledge;
d. providing a set of *interaction primitives* for interacting with domain experts during the knowledge acquisition process;
e. providing *a formal construction process* for constructing specific KA tools from the above primitives;
f. providing a *friendly graphic-based user interface* and *operation guidance* for using the shell correctly.

The design philosophy is this. From the analysis of *generic tasks* [6, 7], we expect to derive a set of *problem solving primitives* and *representation schemas.* We also expect to derive a set of *acquisition primitives* from the analysis of the problem solving primitives and representation schemas. Similarly, from the analysis of fundamental concepts involved in *KA tools* and *KA methods* [4, 5, 13, 15], we expect to derive a set of *interaction primitives.* These four categories of basic primitives play the core role in the shell and are integrated into a primitives kernel. A *construction process* is then applied to construct a specific KA tool in terms of a specific data structure from the primitives kernel. In fact, the specific data structure is derived from the generic structure that represents the primitives kernel and the construction process.

Thus, we need a *tool builder* to generate specific KA tool structures for specific tasks and a *tool interpreter* to actually do the knowledge acquisition work by interpreting the structures. A *knowledge processor* is also needed in the shell for integration and conversion of multiple knowledge representations. A *file manager* is needed to support the management of the generic structure, specialized KA tool structures, intermediate knowledge base, etc. A *window-based graphic user interface* is needed for friendly use. Finally, we designed ***GAS*** as an *open architecture* (i.e. separating the generic structure from specific structures) so that the enhancement of the system can be done readily.

3. THE PRIMITIVES KERNEL

3.1. Problem solving primitives

We apply the *task analysis* technique to generic tasks and derive problem solving primitives. First, the input and output of each generic task are identified. Each generic task is then divided into proper subtasks according to its input/output and applicable problem solving strategies. Problem-solving methods applicable to each subtask are finally analyzed to derive problem solving primitives. Fig. 1 is the analysis result containing a hierarchical relationship among generic tasks and their corresponding problem-solving primitives.

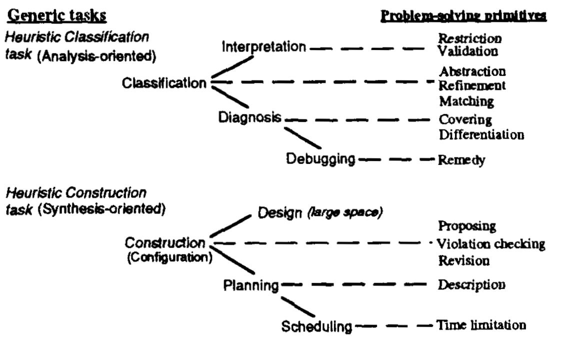

Fig. 1. Relationships between generic tasks and their problem-solving primitives

3.2. Representation schemas

Since a generic task may be applied to different domains, we can analyze those domains by applying the *domain analysis* technique in manual KA methods, which structures the knowledge acquisition process by identifying what types of knowledge the expert appears to be using, in order to derive *representation schemas* capable of representing generic concepts of the domain knowledge models.

A representation schema contains an *organization* primitive and several *relationship* primitives. The organization primitive might be a *network*, a *hierarchy* or a *table* to structure domain knowledge. Relationship primitives define how pieces of domain knowledge are semantically related. Available relationship in ***GAS*** primitives include *Constraint link, Contribute-to link, Revision link, Cause link, Part-of link, Is-a link, Function link, Abstraction link, Definition link, Classification link, Structure link, Generalization link, Condition link, Element link,* and *Construct link.*

3.3. Acquisition primitives

Acquisition primitives are primitive actions that acquire correct domain knowledge for problem solving methods and domain models to solve a specific task. They can be derived by analyzing problem solving primitives and representation schemas, because the analysis of those primitives help us comprehend what types of knowledge are needed and what schemas are appropriate for representing the domain knowledge. Fig. 2 illustrates the relationships among acquisition primitives, problem solving primitives, and representation schemas. Table I lists the acquisition primitives in ***GAS*** and their correspondent problem solving primitives.

Table I. Acquisition primitives

Acquisition primitives	*problem solving primitives*
Scope setting	Restriction
Syntax describing	Validation
Data categorizing	Abstraction
Solution subdividing	Refinement
Data mapping	Matching
Symptom listing	Covering
Data demarcating	Differentiation
Cure describing	Remedy
Parameter proposing	Proposing
Constraint proposing	Checking
Fix proposing	Revision
Format proposing	Description
Time proposing	Time limitation

3.4. Interaction primitives

After analyzing many current manual KA methods [15], we discovered a set of basic interaction techniques is commonly used. Specifically, *concept editor* is derived from the concept analysis method which organizes domain data or related concepts into a hierarchy of categories or a tree structure. *Grid constructor* is derived from the repertory grid method which elicits and analyzes the domain expert's world model based on personal construct theory. *Menu procedure* is derived from the process trace method which traces the decision making process of the domain experts and reveals their problem solving strategies. Finally, *Question/answer procedure* is derived from the unstructured interview and structured interview methods. We also gathered another set of basic interaction techniques from the study of the interfaces of current KA tools. For example, menu driven and table driven procedures are used to select menu and propose parameters in SALT, MOLE, and MORE. Grid constructor is used to construct repertory grid in ETS and AQUINAS [3]. Question/answer procedure is used to acquire necessary information in ROGET and SIS [11].

These techniques allows us to define basic interaction techniques (primitives) for ***GAS***. Because *concept editor* can be replaced by the integration of grid constructor with question/answer procedure, we define *menu driven procedure, grid constructor, table driven procedure* and *question/answer procedure* as the interaction primitives in the primitives kernel. The relationship between acquisition primitives and basic interaction primitives is described in Table II.

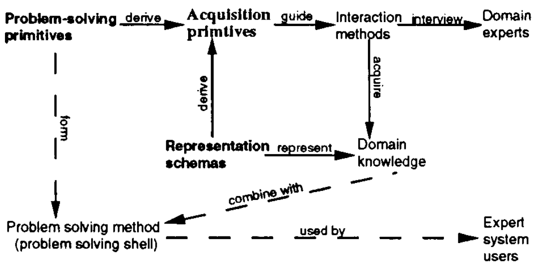

Fig. 2. Relationships among problem-solving primitives, representation schemas and acquisition primitives

Table II. Interaction primitives

Interaction primitives	*Acquisition Primitives*
Table-driven procedure	Symptom listing, Cure describing, Parameter proposing, Constraint proposing, Fix proposing, Format proposing, and Time proposing
Menu-driven procedure	Scope setting, Syntax describing, Data categorizing, Solution subdividing, Symptom listing, Data demarcating, Format proposing, Time proposing
Grid-driven procedure	Syntax describing, Solution subdividing, Data mapping, Data demarcating
Q/A procedure	Scope setting, Data categorizing, Data mapping, Symptom listing, Data demarcating, Cure describing

4. THE GENERIC KA STRUCTURE AND SPECIFIC KA STRUCTURES

For the purpose of future ready enhancement, we designed ***GAS*** as an open architecture. That is, a generic KA structure is developed serving as a template in ***GAS*** to maintain the generic data and to guide the user operations especially during the generation of specific KA structure. It includes *task requirements* and *system functions*. The task requirements contain templates for *generic tasks* and all the *primitives*. The system functions include templates for *knowledge verification, knowledge representation* and *explanation function*. Users can select proper features from specific system functions. They are then integrated with the specific KA structure requirements into a specific KA tool. Each node in the generic structure includes node identification, subordinates generating rule, and node name. Node identification uniquely identifies a node. Subordinates generating rule specifies how to generate subordinates. Node name is a symbolic description of the node.

Based on the generic KA structure, users can construct specific KA structures for building specific KA tools. A specific KA structure contains three fields for each node. They are node identification, subordinates, and node name. Functions of these fields are similar to those in the generic structure except the subordinates field is only used for describing the children of a node rather than for generating children.

5. SYSTEM MODULES

Six modules are developed in ***GAS*** to implement the concepts described so far including building and executing specific KA tools from the primitives kernel. They are structure editor, tool generator, tool interpreter, knowledge processor, file manager and window-based user interface. The operation sequence of these modules described in terms of function modules and file specifications is illustrated in Fig. 3. The following details each of the modules.

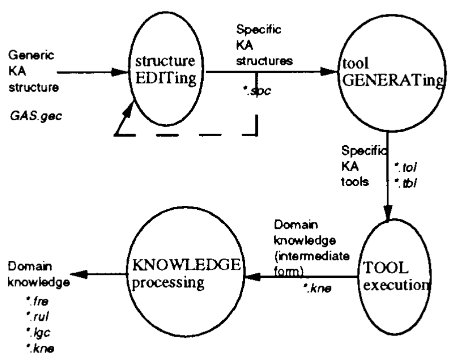

Fig. 3. Function modules and file specifications in *GAS*

--- Structure editing

The purpose of the structure editor is to help users edit and validate specific KA structures. It creates a specific KA structure through the selection of desired nodes from the generic KA structure. It also validates the construction of the specific KA structure.

--- Tool generation

After creating a valid specific KA structure, users are asked to deliver domain specific symbols into this structure by tool generator. It then generates a specific KA tool (or domain KA tool) from the structure. It also checks whether or not the nodes that represent the basic interaction techniques of a specific KA structure have been given proper domain-specific symbols.

--- Tool interpretation

Knowledge acquisition is performed in *GAS* by executing a domain KA tool (i.e., a specific KA tool with domain symbols) by the tool interpreter. That is, the tool interpreter extracts knowledge from domain experts according to the specifications in the domain KA tools. It also allows the modification on the KA tools.

--- Knowledge processing

The knowledge processor is responsible for various knowledge processing activities. Functions included are representation conversion, knowledge base browsing, knowledge base integration, and knowledge base testing.

--- File management

The file manager manages the information about KA tools, knowledge base, specific KA structures and the generic KA structure for *GAS*. Several types of files are created in *GAS*. They are generic structure (*GAS.gec*), specific structure (**.spc*), specific tool (**.tol*), domain-specific symbols (**.tbl*), intermediate knowledge base (**.kne*), and knowledge base with various representation (**.fre* for frames, **.rul* for rules, **.lgc* for logics) (see Fig. 3). The file manager is responsible for storing and updating these files.

--- Window-based user interface

A window-based user interface module provides a friendly graphical menu for users. It guides the *GAS* users during the specialization of the generic KA structure and the interpretation of specific KA tools.

6. CONCLUSION AND FUTURE WORK

We have described the design philosophy, the system architecture, and the system operation of the generic knowledge acquisition shell *GAS*. *GAS* is developed to serve as a foundation for further study of integrating advanced capabilities into the knowledge acquisition process. Nevertheless, *GAS* has at least made a contribution to the research of knowledge acquisition; that is, selecting (or developing) suitable domain specific KA tools for specific domains can be experimented on the framework without resorting to physically purchasing (or struggling to integrate) many off-the-shell KA tools.

The work with *GAS* has raised a number of interesting issues including *how to apply KA tools generated from GAS to diverse application domains, how to integrate machine learning techniques into GAS, how to enhance the primitives kernel by adding or refining primitives for empowering GAS*, and so on. These have been planned as our further work.

REFERENCES

[1] J. S. Bennett, "ROGET: A knowledge-based system for acquiring the conceptual structure of a diagnostic expert system," *Journal of Automated Reasoning*, vol. 1, pp. 49-74, 1985.

[2] J. H. Boose, "Personal construct theory and the transfer of human expertise," *Proc. of AAAI-84*, Austin, Texas, 1984.

[3] J. H. Boose, "Uses of repertory grid-centred knowledge acquisition tools for knowledge-based system," *Proc. of 2nd AAAI Workshop on Knowledge Acquisition for Knowledge-based system*, Banff, Canada, 1987

[4] J. H. Boose and B. R. Gaines, Knowledge Acquisition Tools for Expert Systems, Academic press, CA, 1988.

[5] J. H. Boose, "A survey of knowledge acquisition techniques and tools," *International Journal of Knowledge Acquisition*, vol. 1, pp. 3-37, March 1989.

[6] T. Bylander and B. Chandrasekaran, "Generic tasks in knowledge-based reasoning: The 'right' level of abstraction for knowledge acquisition," *International Journal of Man-Machine Studies*, vol. 26, pp. 231-244, 1987.

[7] W. Clancey, "Heuristic classification," *Artificial Intelligence*, vol. 27, no. 3, pp. 289-350, 1985.

[8] J. Diedrich, I. Ruhmann and M. May, "KRITON: A knowledge acquisition tool for expert systems," *International Journal of Man-Machine Studies*, vol. 26, no. 1, pp. 29-40, 1987.

[9] L. Eshelman, "MOLE: A knowledge-acquisition tool for cover-and-differentiate systems," In S. Marcus(eds.), Automating Knowledge Acquisition for Expert Systems, Kluwer Academic Publishers, MA, pp. 37-80, 1988.

[10] G. Kahn, S. Nowlan and J. McDermott, "Strategies for knowledge acquisition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 5, pp. 511-522, September 1985.

[11] A. Kawaguchi, R. Mizoguchi, T. Yamaguchi and O. Kakushi, "SIS: Shell for interview systems," *Proc. of IJCAI-87*, pp. 359-362, 1987.

[12] G. Klinker, "KNACK: Sample-driven knowledge acquisition for reporting systems," In S. Marcus(eds.), Automating Knowledge Acquisition for Expert Systems, Kluwer Academic Publishers, MA, pp. 125-174, 1988.

[13] S. Marcus, Automating Knowledge Acquisition for Expert Systems, Kluwer Academic Publishers, MA, 1988.

[14] S. Marcus and J. McDermott, "SALT: A knowledge acquisition language for propose-and-revise systems," *Artificial Intelligence*, vol. 39, no. 1, pp. 1-37, May 1989.

[15] K. L. McGraw and K. Harbison-Briggs, Knowledge Acquisition Principles and Guidelines. Prentice-Hall Inc., NJ, 1989.

[16] D. Offutt, "SIZZLE: A knowledge-acquisition tool specialized for the sizing task," In S. Marcus(eds.), Automating Knowledge Acquisition for Expert Systems, Kluwer Academic Publishers, MA, pp. 37-80, 1988.