separation and hand-off rules. Addition of further rules, case-by-case, without a more rigorous formalized approach could lead to the creation of the expert system which is not fully competent and effective in executing complex ATC actions. Thus, based on ATC literature studies and exhaustive discussions with experts, a more rigorous approach has been proposed.

Most of the controller's actions result in verbal communication to a pilot, another controller, or an aviation authority. Each of these actions has unique phraseology. Some of them will require parameters given either in symbolic or numeric form (e.g., airport, airway name, aircraft identification, altitude, heading). Interviewing the ATC experts we have found that each action is originated by some primary situation fact (e.g., radar contact, pilot request, predicted separation violation, need for hand--off). The goal of the controller's action is to eliminate this primary fact. The controller's action depends, however, on some other secondary situation facts describing the situation at the moment of decision (e.g. other traffic, type of aircraft, flight plan, weather, airport situation, knowledge of area).

To present the system in a formal way separate groups of controller actions need to be identified. There are groups of actions related to: initial clearance, departures, pilot requests and emergencies, separation, hand-offs, arrivals, weather, etc. For all these groups a specific actions with their respective phraseology have been identified. These actions need to be represented as facts identified by the name, appended with possible parameters (symbolic or numeric) including time of issue, time of validity, etc. Both primary situation facts (originating a specific group of actions), and the secondary situation facts (defining conditions to launch a specific action) have been identified.

As the ATC actions conform the cause-effect paradigm, we represent the controller's knowledge in a rule-based form. To simplify the explanation, the situation facts constitute premises (IF clause), and the action facts constitute the conclusion (THEN clause). An inference engine is responsible for searching the knowledge base and firing the rules. Asserting facts to and/or retracting facts from the knowledge base constitutes a principal knowledge control handling mechanism. The knowledge control issues are crucial for the efficiency and correct performance of any expert system. An available expert system shell – Inference's Automated Reasoning Tool – has been used in the initial project phase to expedite the ATEC development.

The presented considerations review the scope of the ATEC project concentrating mainly on the operational knowledge acquisition problem. Using formalized approach for the specified acquisition problem. Using a formalized approach for the specified domain knowledge acquisition we have identified facts and rules as components of the ATC Expert System. The approach is general enough to be applied for building an expert software for most of the controller-dispatcher type of systems. Most of the situation facts represent knowledge about the current system state and are acquired from the simulation module. To create a fully intelligent system capable of simulating the human expert is a visionary goal. But the first steps to achieve this goal have been accomplished. Subsequent modifications, modules interfacing, verification, and validation of the system represent the future tasks in the project realization.

## A Knowledge Acquisition Model With Applications For Requirements Specification And Definition

*Tetsuo Kinoshita*
Systems Laboratories
OKI Electric Industry Co., Ltd.
11-22, Shibaura 4-Chome,
Minato-ku, Tokyo 108, JAPAN

### Introduction

From a practical point of view, the knowledge acquisition task for developing knowledge-based systems may be viewed as a cooperative task between domain experts, knowledge engineers, and support systems (environments). Based on this perspective, a system called the Multi-Layered Knowledge Acquisition Model (MLKAM), has been proposed [1].

MLKAM consists of three kinds of knowledge representation layers: Domain Concept Structure Layer (DCSL), Knowledge Representation Structure Layer (KRSL), and System Oriented Model Layer (SOML). Each layer provides the descriptive primitives for representing domain knowledge and transformation rules between the primitives of each layer. Primitives and rules are stored in the acquisition support knowledge base of each layer.

These layers are structured (connected) hierarchically, i.e., DCSL -> KRSL -> SOML, in order to acquire domain knowledge based on the successive refinement strategies encountered at each layer, and to create the knowledge bases of the objective system.

The acquisition process of MLKAM consists of three steps. First, domain knowledge is *acquired* in DCSL. Second, the domain knowledge in DCSL is *formalized* by the knowledge models in KRSL, and third, formalized knowledge is *translated* into the descriptions of knowledge representation languages of the objective knowledge-based system in SOML.

In DCSL, without concern to what kind of knowledge representation languages will be used to implement the knowledge bases of the objective knowledge-based systems, domain knowledge is described freely by domain experts in the form of graph structured representations through the interactive man-machine interface system and interview facilities.

In KRSL, domain knowledge in DCSL is translated by knowledge engineers (KEs) into the form of knowledge representation models (e.g., Rule Model, Frame Model, Network Model, Logic Model or any knowledge model based on any framework). These knowledge models are abstractions of existing knowledge representation languages. According to structure and properties of (any portion of) domain knowledge in DCSL, KEs select the most suitable knowledge representation model and convert (any portion of) the domain knowledge in DCSL into descriptions of the selected knowledge representation model. The control structures provided by the knowledge representation model are then attached to the domain knowledge.

In SOML, the formalized knowledge in KRSL is translated into internal descriptions of existing knowledge representation languages of the objective knowledge-based systems. Actual descriptions which are stored in the knowledge base of the objective systems can be generated from those descriptions in SOML.

Separating the process of the acquisition of domain knowledge from the process of the transformation/generation of formalized knowledge (represented by knowledge representation models/languages), can avoid excessive confusion of domain knowledge and mixed control knowledge provided by knowledge representation languages. Further using the transformation rules of each layer, the semi-automatic conversions between knowledge layers can be realized. Using the graph structured knowledge representation as the surface representation in DCSL and KRSL, it is easy for domain experts and KEs to understand, manipulate, and modify the represented knowledge. While the frame-based knowledge base system [3] is adopted as the internal representation scheme which supports the whole implementation of MLKAM.

On the other hand, in knowledge-based systems for design tasks, it is important not only to acquire design knowledge (heuristics) from domain experts, but also to acquire design specifications from requirements definers (in many cases, end users). The former is the problem of current knowledge acquisition, and the latter is the problem of requirements/specification definition. There have been several studies concerned with requirements specification definitions [4][5], but complete solutions have not been given. From the knowledge engineering point of view, the process of extracting requirements/specifications from requirements definers (end users) can be formalized as the process of knowledge acquisition. In order to get a more useful methodology of the requirments specification definition, one promising approach may be to apply the techniques of the knowledge acquisition.

In this sense, the framework of MLKAM is applied to formalize the process of the requirements/specification definition of the design task. In order to build a supportive environment for defining requirement/specifications, representation primitives of the knowledge layers of MLKAM had to be customized in accordance with the specific design task. The design problems associated with computer communication systems was selected where the representational primitives of DCSL and KRSL were customized according to the knowledge-based design methodology for computer communication systems (KDM-CS) [2]. In this case, SOML was omitted because it was enough to represent the formal requirements/specifications in the form of the internal representations of MLKAM and send them on to the next design phase of KDM-CS.

In KDM-CS, the requirements/specification definition is the first phase in the design process, and three kinds of knowledge models (virtual machine models) of computer communication systems were defined to formalize end user requirements. According to these knowledge models, end user requirements (initial requirements) are collected by the first model, analyzed by the second model, and formalized into the requirements/specifications by the third model. Final requirements/specifications are represented by the sets of virtual commands which realize end user requirements.

According to the knowledge models of KDM-CS, representational primitives and transformation rules of MLKAM are defined and stored in the acquisition support knowledge base of each layer of MLKAM. DCSL primitives are defined based on the first knowledge model of KDM-CS, and KRSL primitives are also defined based on the second and third knowledge models of KDM-CS.

In DCSL, end users (requirements definers) describe their requirements by using graph structured representational primitives (templates) with interview facilities. Missing or inconsistent information is detected and corrected through the interviews of DCSL. Basically, those interviews are extracted from representational templates. In KRSL, initial requirements collected in DCSL are analyzed and transformed into the formal representations of requirements/specifications by designers of requirements specifications. Control structure of requirements/specifications, such as a sequence of virtual commands, are also mixed with initial requirements in this layer. Final descriptions of requirements/specifications are stored in the frame-based knowledge base of MLKAM.

**References**


1. Kinoshita, T., et al., "Multi-Layered knowledge Acquisition Model" (in Japanese), *Proc. Symp. Frameworks of Artificial Intelligence System, IPSJ*, pp. 141-150, 1987.

2. Kinoshita, T., et al., "Knowledge-based Design Support System for Computer Communication System," *Journal SAC*, Vol. 6, No. 5, IEEE, pp. 850--861, 1988.

3. Kinoshita, T., et al., "Experimental Knowledge Base System based on the Frame Model," *OKI Tech. Rev.*, Vol. 52, OKI Electric Industry, pp. 1-8, 1985.

4. Rzepka, W., et al, (Eds), "Speical Issue: Requirements Engineering Environment: Software Tools for Modeling User Needs," *IEEE Computer*, Vol. 18, No. 4, 1985.

5. Balzer, R., et al., "Software Technology in the 1990's: Using a New paradigm," *Computer*, Vol. 16, No. 11, IEEE, pp. 39-45, 1983.


## A Grid-Based Tool For Knowledge Acquisition: Validation With Multiple Experts


*Mildred L. G. Shaw*
Knowledge Science Institute
Department of Computer Science
University of Calgary
Calgary, Alberta CANADA


**Introduction**

Knowledge Support System Zero (KSS0) is a system providing an integrated set of tools for knowledge acquisition for knowledge based-systems. *Elicit* provides facilities for eliciting the important dimensions of an expert's thinking on a topic; *Exchange* extends this to share entities and attributes between experts and elicits differences in perspective and terminology as well as disagreements on the topic. *SOCIO* processes results from several experts to reveal the similarities and differences in the concept systems of different experts, or the same experts at different times, construing a domain defined through common entities or attributes. It can be used to focus discussion between experts on those differences between them which require resolution, enabling them to be classified in terms of differing terminologies, levels of abstraction, disagreements, and so on. It provides a framework for identifying *consensus*, *correspondence*, *conflict*, and *contrast* in a knowledge acquisition system with multiple experts.

KSS0 has been evaluated against a model for a knowledge support system and experiments are reported in knowledge acquisition of spatial interpolation techniques for contour maps. Results are described on validation experiments to show the extent to which this system can replace standard interviewing techniques: 1) Does an expert always use the same terminology? 2) Do experts agree on their terminology in talking about a topic? 3) To what extent to experts agree among themselves about the topic? 4) To what extent does each expert agree with the knowledge at a different time? 5) To what extent does an expert find the generated rules meaningful?

KSS0 provides facilities to combine information from a number of different sources, including text, expert interviews, expert decision-making using a variety of different techniques including text analysis, entity and