A RULE-BASED BLACKBOARD KERNEL SYSTEM; SOME PRINCIPLES IN DESIGN

by

M.D. Rychener, R. Bánares-Alcántara and E. Subrahmanian

DRC-05-04-84

December, 1984

A Rule-Based Blackboard Kernel System: Some Principles in Design

Michael D. Rychener
René Bañares-Alcántara**
Eswaran Subrahmanian***

Design Research Center
Carnegie-Mellon University
Pittsburgh, PA 15213

21 November 1984

To appear: IEEE Workshop on Principles of Knowledge-Based Systems
December, 1984, Denver, CO

Keywords: Knowledge Utilization, Focus of Attention

This research has been supported by the CMU Design Research Center, under the direction of Prof. Sarosh Talukdar.

**Dept. of Chemical Engineering, CMU.

***School of Urban and Public Affairs, CMU.

Table of Contents

A Rule-Based Blackboard Kernel System:

Some Principles in Design

ABSTRACT. A rule-based kernel system to support reasoning within the blackboard problem architecture has been developed, with enough generality to support a number of expert system projects. The blackboard allows diverse expert-system modules to cooperate in solving complex design problems, making use of network communication to reduce difficulties with combining programs in various languages and operating systems. Features of the internal structure of the kernel allow it to be adapted readily to new problem areas and to new configurations of expert modules. From our experience we derive some design principles for enabling multi-expert cooperation, and for using expert knowledge effectively.

1. Motivation and Target Applications

Engineering design places many diverse demands on expert system software. Some past attempts at using specialized 'empty[1] systems came up against basic limitations in those systems, particularly in their interface to existing analysis software. [9] There were also difficulties in evolving the knowledge base, in accessing hardware sensors, and in making use of history, causality and other databases of information. Thus we must aim to build software that will be expandable and readily adaptable to diverse demands, while containing AI techniques (including inference engine, search strategy, explanation, knowledge acquisition aids, etc.) in a useable form.

We also desire portability of the expert system tools. We need software that is concise and readily embedded in new (emerging) computing environments. (The use of Lisp as a basis is detrimental to this goal, given its large size and complexity - much of which is unnecessary for expert systems.)

Rule-based kernel software has some potential for satisfying some of these requirements, due to the following features:

- Modification by adding or changing rules;
- Explanation of the kernel itself by applying its own capabilities to its rules;
- Implementation with high-level and concise code;
- Openness and flexibility of control (if rules suitably designed).

We define a kernel to be a small set of rules, enough to provide a framework within which further elaboration and refinement can take place. For OPS5, a system with about 100 to 200 rules is sufficient for this purpose. A kernel is enough to establish representation and control conventions,

and is coded in such a way that extension is easy. For example, it may contain goals whose processing is only done in a 'default' way, leaving it up to the user to add rules for whatever further actions are desired. Control and data concepts must be expressed (and documented) clearly, so that the user can readily grasp the existing rules.

A kernel has some problem-solving organization as its focus, e.g., backward-chaining diagnostic reasoning, hierarchical planning, or learning by being told. Thus there are a few basic functional capabilities in the kernel, determined by the chosen organization. Along with these are basic user interaction, tracing, explanation, and general support functions. For each of these functions, the basis of the capability is given, and the user is free to expand it according to specific domain demands. The difficult part of building such a kernel is often the selection and design of components and their interfaces, rather than the implementation as rules.

We have chosen the blackboard model of problem solving as our kernel's system organization. This model was first discussed by Newell [7], and its most effective application to date was in the Hearsay-II system [1]. Its orientation is towards cooperative problem-solving by a group of programs, termed knowledge sources. It is intended to be analogous to a group of human experts sitting around a table and discussing a problem in which each one can bring some specialized expertise. Each one offers ideas and these can be freely used by each other participant. It is as if each of the various contributions were written on a blackboard that is visible to all. There is usually a referee to ensure that two experts don't speak at once or try to write on the blackboard at once in the same place.

In computer implementations of the blackboard model, the blackboard is viewed as a data structure containing a variety of types of information. These will be described in the next section. This data structure can be read by the other modules of the system, and some modules have enough privilege to perform additions and modifications to it. The function of coordinating the various knowledge sources (KSs) is called the focus of attention (FA) (this is similar to the referee in the above analogy). The FA in particular has full write privileges on the blackboard. There is a special module in the system that handles interactions with the user, called the user interface (UI). In our OPS5 implementation, the Working Memory is the blackboard, and it is partitioned into several areas, to be described in the next section. All rules within the various KSs can perform pattern-matching on the blackboard portion of Working Memory. Modifications to the blackboard are done by asserting goals in the Working Memory that are processed by the FA.

The blackboard model appears to have the following advantages:

- The addition of KSs without requiring knowledge of other KSs; only FA and UI need to

know a little about new modules; the system may grow gradually, and is configurable in several ways for experiments;

- A clean framework into which domain expertise is easily added, either as sets of expert rules or as entire programs, databases, etc.;
- Mechanisms for directing problem solving while taking account of diverse types of expertise;
- Means for combining evidence, resolving conflicts, handling asynchronous communications, etc., which are problems that arise in distributed, cooperating programs;
- Accommodation to a variety of modules, various languages, various processors, etc., provided they meet minimal interface demands.

There are further advantages from implementing the blackboard model as a rule-based system. Production-rule systems are uniquely suited to blackboard types of reasoning, in being pattern-directed. RETE matching [4], the internal mechanism of execution, is near optimal in picking out patterns in the blackboard, and within the pattern matcher, parallel processing is a good possibility. The blackboard is symbolic, so there are potentially multiple levels of representation, and diverse types of representations can be accommodated.

The current implementation has some features that make its development promising. It may reside on a powerful personal workstation (such as the processors in CMU's planned campus-wide network), i.e., it is small, self-contained, and written so as to be portable to other similar systems. It may utilize graphics and other features, for a powerful user interface. It will build upon the experience gained from some current prototype systems at CMU, in specialized engineering domains, focussing on issues of useability, efficiency, and advanced AI capabilities.

Within the Design Research Center at CMU, a variety of problems are encountered that demand the integration inherent to the blackboard model. We have designed our kernel so that these various demands can be serviced. The domains targeted by our design are the following:

- Selection of chemical catalysts by chemical engineers, where experts are used in the areas of literature search, thermodynamic calculations, chemical reactions, catalyst properties, and others;
- Mobile robots, where the experts are planners, sensor modules, vision, etc.;
- Production scheduling, where the experts look at various aspects of a factory schedule;
- Design of metal alloys, with a collection of experts analogous to that of the chemical catalyst case;

- Integrated Circuit layout, where various electrical engineering analysis tools can be brought to bear on a chip design;
- Structural design of buildings, which brings together various civil engineering techniques, along with some knowledge-based heuristic modules.

The main result of our continuing work will be the refinement of the blackboard kernel design and its implementation as a rule-based system, along with the evidence of its effectiveness over a diverse set of engineering design problems, starting with the above list. We hope to encode some of the expertise in using and adapting the kernel to new domains as rule-based expert modules, and to fit them gracefully into the blackboard system.

There are three aspects to the Kernel design: the blackboard data structures, the rules that define the processes on the blackboard, and the messages that define the communications among the various modules. The data structures are the frame-like elements of the OPS Working Memory, and are established by defining the slots within the various element classes. The rules are established as a result of combining the demands of the blackboard model, the data structures and the messages that the rules have to process. Two main sets of rules are in the Kernel: the focus of attention and the user interface. Within the focus of attention, the main AI aspect is the management of tasks within a heuristic search paradigm: the focus maintains the tree of tasks and enough information for backtracking to be done and for progress to be maintained toward the main objective of the system. The messages are also defined as frame structures, with their contents organized into pre-established slots.

We distinguish between general and domain-specific data and rules: the former are built into the Kernel, and are processed exclusively by Kernel rules; the latter are built by the user (for some problem domain), and are processed by the user's rules, contained either in the focus of attention or in other rule sets created by the user. The Kernel is a minimal set of general rules that provide the most basic capabilities for handling problem-solving tasks, for maintaining the blackboard, and for communicating with the user. Building an expert system with the Kernel as the basis requires:

1. filling in details on the various expert KSs of the subject domain (some example modules are provided with the Kernel as guidance in this task);
2. providing any special-purpose control and user-interface rules that the domain needs;
3. and specifying the overall configuration of KSs and Kernel, most of which is done by filling in OPS elements that describe the modules and how they function.

Our blackboard is partitioned into several major classes of element:

- Problem specification, overall problem objective, and criteria to be satisfied by any solution;
- Designs (plans, hypotheses, etc., depending on the type of problem), mostly domain-specific, but containing a few general-purpose slots that the focus of attention will use to direct problem-solving;
- Tasks generated directly by the overall goal or by the experts in the course of their designing (maybe organized into an agenda);
- Estimates of difficulty of doing various tasks, by various experts;
- Queries among experts for problem-related information;
- Communications to and from the user, e.g., requests for explanation.

We expect that more advanced versions of the Kernel will include one or more KSs that are oriented towards the process of building expert systems using the Kernel. I.e., there would be modules to support the process of adapting the general kernel to specific tasks. The contents of these 'meta' KSs will be developed as we extend the set of tasks that the present system is applied to.

2. Two current applications

Our results so far include two rule-based kernel systems, as described in the following subsections. Both systems are coded in OPS5 [3], but we may soon begin to use OPS83 [2], which is more suitable to our eventual computation environment, being implemented in the C language.

2.1. Catalyst selection

DECADE is a prototype expert system for catalyst selection. From a specified reaction it attempts to propose a set of materials with high probability of being good catalysts for the input reaction. The selection of a catalyst is a problem that is currently solved only by a relatively small number of experts. It is decomposable into smaller and very different subproblems, some of them amenable to algorithmic solution, but the majority only solvable through the use of heuristic reasoning due to their lack of formalization. Also, since their order of execution is not fixed but varies greatly depending on the characteristics of the individual problem, an opportunistic control strategy, such as the blackboard model, seems to be more appropriate than a static scheduling approach.

Two kinds of Expert KSs (EKSs) can be recognized in DECADE:

1. General Purpose EKSs.
    - User Interface (7 *rules).* Manages the initialization, information and question messages (in the latter it checks for acceptable answers and prints a menu when

necessary). Will be extended to deal with information requests for concept properties and the explanation capabilities.

- Focus of Attention (76 *rules).* Initializes and reinitializes the system when no goal is in process of solution. Receives and evaluates the estimates from the different EKSs to solve a goal, assigning its solution to the most promesing. Coordinates the synchronization of the EKSs with their subgoals. Will be extended to take in account estimates for a *priori* problem decomposition into subproblems, synchronizing them, checking for possible duplication, recording dependencies, and keeping track of the abstraction level used while solving each one of them.
- Scheduler (5 *rules).* Performs a census procedure that prompts every EKS at initilization and constructs a dynamic list of accessible EKSs, the type of problems they solve, their status, etc., for information purposes. (The list is updated by the EKSs themselves.) Can be extended to aid the Focus of Attention in the synchronization of parallel and sequential subgoals.

2. Domain Specific EKSs.

Every EKS has to have: one rule for issuing estimates (minimum), one rule to answer the census, one rule to receive information of goal assignment, one rule to inform the Focus of Attention of the completion of a goal, and as many rules as needed for subgoal posting and information retrieval.

Examples of the individual EKSs are:

- Specification of the reaction EKS *(36 rules).* Makes extensive use of the User Interface to query the user for the input reaction.
- Thermodynamic Feasibility EKS (74 *rules).* Mostly algorithmic, it connects to a set of LISP functions.
- Reaction classification EKS (8 *rules).* A typical diagnostic task, it is performed using SRL (Schema Representation Language) [10] and LISP functions.
- Catalyst prescription EKS (8 *rules).* Same as the Reaction classification EKS.

While the solution of diagnosis problems allows for an a *priori* structuring of the solution space (by having a static goal tree and searching through it), the solution of design problems requires a dynamic generation of the goal tree. The blackboard model proves to be a useful mechanism towards this purpose.

Following one of the principles of expert systems, a separation between the domain knowledge and the control knowledge (also known as metaknowledge) is attempted. OPS5 is used for the

representation of the control. Its application is natural for this task, and it allows modularity and easy maintenence, but its knowledge representation is poor. Since SRL overcomes this limitation it is therefore used in the representation of the domain knowledge. An apparently efficient but cumbersome communication mechanism exists between OPS5 and SRL. A more general and flexible communication among OPS5, SRL and some algorithmic languages would enhance OPS5 convenience.

2.2. Scheduling in manufacturing

A prototype expert system for scheduling in a hypothetical machine shop in a flexible manufacturing environment has been built based on the Blackboard architecture. The scheduling problem is set in a small machine shop that has 4 different machines and produces four products. Each product can be manufactured by using two different sequences of machine operation. No manufacturing sequence has more than 3 machines. The problem is to schedule the products with nonzero demand within a given time horizon. The objective of the problem solver is to arrive at a feasible solution.

Based on the principle of problem solving by co-operative independent knowledge sources, our model provides structures by which the knowledge about the production of each of the products can be coded to create a Product EKS. The control strategy (FA) to schedule the tasks for producing the products uses a prioritizing scheme. The major advantage of this approach is that new production configurations for an existing product can be added just by putting the new product configuration into a data base (a collection of OPS5 working-memory elements). In the case of addition of new products to the manufacturing line, there only would be required the addition of Product EKSs for them in a modular fashion, without redesigning the entire system.

The current system has the following configuration:

- **Blackboard:** Several frame classes, implemented as OPS5 working memory element classes, have been established. Typical frame classes in the current system are: a) task hlerachy, b) problem specification, c) Plan structure, and d) information exchange element
- **User Interface** ***(38 rules):*** User interface is responsible for querying the user for the inputs required to solve the problem, providing means to display the partial plans and the final plan, to allow the user to intervene in altering the prioritizing scheme and providing explanations for the ordering of products scheduled. The rules for each of the above tasks performed by the user interface themselves form independent modules under a set

of top-level user interface rules.

- Focus of Attention *(20 rules):* The tasks of the focus of attention are to generate, assign and, monitor tasks required to solve the problem and to issue requests to other domain EKSs for information needed to assign priority to tasks. The focus of attention tasks are also divided into tasks required to set up the problem and tasks required to solve the problem. The problem setup phase is mainly used to identify inputs required to solve the problem and request the user to provide them thorugh the user interface.
- Information Exchange Handler *(3 rules)* A set of rules that identify the EKS that can answer a request for information and monitor the status of all such requests. In the current system this set of rules is not very significant. However, in a distributed environment an information exchane handler can be used effectively to process information requests between EKSs and the focus of attention in parallel by maintaining information on location of EKSs, the problem they solve and completion status of the request
- Other Domain-Specific EKSs: All domain-specific knowledge sources have two basic types of rules:
  - o task-based rules: Rules that are invoked based on tasks assigned to the EKS, e.g., rules for posting the schedule for a product based on the current demand and partial plan.
  - o information response rules: rules that are responsible for answering information requests from the EKS, e.g., given the current state of the schedule plan on the blackboard, respond with the earliest time at which the product can be scheduled for production.

  Another type of rule that specifies conditions under which tasks performed by the EKS may request information from other knowledge sources is not relevant in our test problem. However, rules of this type may be required in other problem solving situations (e.g. the trip planning problem [5]).

  In our prototype, there exists an EKS *(11 rules)* for each of the products and EKS (3 *rules)* for checking the time-horizon constraint.

In the context of a specific machine shop working in a flexible manufacturing environment the scheduling problem requires detailed knowledge of the alternate production sequences and manufacturing times. Scheduling has to be done periodically to reflect the current state of the machine shop in terms of machine availabilty, current orders for products and time constraints. In the prototype system, the problem of scheduling is handled by generating appropriate goals and the

tasks required to achieve those goals. The current system thus differs from a diagnostic system in the sense that the solution space is generated dynamically as opposed to a priori as in a diagnostic system. Systems that solve problems by generating goals and tasks dynamically in an opportunistic fashion have been termed "generative problem-solving systems" [5].

The prototype system was built using OPS5. Since the size of the problem is small, no attempt was made to store product manufacturing sequence and time data in a separate data base. However, in a larger problem it may be necessary to interface the system to an SRL data-base.

3. Distinguishing Features

A number of features distinguish our blackboard approach from past work [1,5,8,6]:

- Expert KSs work on a wider variety of tasks, in seeking the main goal / design objective, while in Hearsay-II, for instance, there is only 1 task, to understand the utterance, with KSs focusing on various levels of sub-tasks in restricted areas of the utterance;
- Our tasks are dynamically generated, not known in advance & not necessarily embedded in a master planner module;
- Focus of attention is potentially much less in control, since the user can adapt it to meet particular problem demands; there may be domains where distributed, decentralized control is best; the focus of attention can easily incorporate an expert's control heuristics and design strategies;
- We seek to maximize the distribution / specialization / decentralization of expertise, and to maximize flexibility of assembly, flexibility in configuration of KSs, and growth of the set of EKSs;
- Multiple blackboards are possible: an expert KS can itself be a blackboard system, on a remote processor, but using the same communication mechanisms as other KSs;
- We aim to assemble and coordinate existing engineering software, running on mainframes, and to use a networked personal workstation as our host machine for the blackboard; (relatively small, cheap workstations can support the use of such languages asOPS5andOPS83).

4. Principles

In conclusion, we make the following observations and offer them as principles to be considered in many domains where expert system techniques are applicable.

For complex engineering design tasks, it is essential to be able to integrate existing engineering

design and analysis programs. We have found the blackboard model to be suitable for this purpose. Its encoding as a rule-based system is important in making it open to expansion as dictated by the domain.

Within our blackboard model, the grain of interaction is coarser than in models like Hearsay-II. Communications are in terms of tasks and items of information rather than at a more detailed problem level. This enables the diverse expert KSs to interact an important feature. But it orients the system to less frequent, higher-level cooperation. The KSs are more independent and are able to solve significant aspects of problems on their own without close consultation through the blackboard.

Our approach uses a relatively simple kernel, and derives its power in the domain from the complex interactions that develop among the KSs. This is an extension to a larger scale of a basic principle of past expert systems, namely their combination of a simple inference engine with a complex knowledge base.

Finally, we expect our research to support the basic principle of rule-based systems that says that openness and explicitness of the knowledge base is essential to intelligence, i.e., essential to proper and effective utilization. We have extended this to the basic inferences of our problem-solving architecture by making the blackboard mechanisms themselves be rule-based. The reasoning strategies followed by the system are now open to expert system operations such as explanation and interactive revision, enhancing conventional explanations, which have been in terms of expert domain rules.

5. References

[1] Erman, L D., Hayes-Roth, F., Lesser, V. R., and Reddy, D. R.
The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty.
Computing Surveys 12(2):213-253, 1980.

[2] Forgy, C. L.
The OPS83 Report.
Technical Report CMU-CS-84-133, Carnegie-Mellon University, Dept. of Computer Science, May, 1984.

[3] Forgy, C. L.
OPS5 User's Manual.
Technical Report CMU-CS-81-135, Carnegie-Mellon University, Dept. of Computer Science, July, 1981.

[4] Forgy, C. L.
Rete, a fast algorithm for the many pattern / many object pattern match problem.
Artificial Intelligence 19:17-37, 1982.

5 References

[5] Hayes-Roth, B. and Hayes-Roth, F.
A cognitive model of planning.
Cognitive Science 3:275-310, 1979.

[6] Lesser, V. R. and Corkill, D. D.
The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks.
AI Magazine 4(3):15-33, Fall, 1983.

[7] Newell, A.
Some problems of basic organization in problem-solving programs.
In Yovits, et al (editor), *Self-Organizing Systems*. Spartan Books, 1962.

[8] Nii, H. P. and Aiello, N.
AGE (Attempt to Generalize): A knowledge-based program for building knowledge-based programs.
In *IJCAI6*, pages 645-655. Tokyo, 1979.

[9] Rychener, M. D.
Expert systems for engineering design: problem components, techniques, and prototypes.
Technical Report DRC-05-02-83, Carnegie-Mellon University, Design Research Center, December, 1983.

[10] Wright, J. M. and Fox, M. S.
SRL/1.5 User Manual.
Technical Report, Carnegie-Mellon University, Robotics Institute, December, 1983.