On the Requirements of Active Expert Systems

Tim Finin
Computer and Information Science
University of Pennsylvania
Philadelphia, PA USA

phone: 215-898-6783
network: tim@cis.upenn.edu

David Klein
Computer and Information Science
University of Pennsylvania
Philadelphia, PA USA

phone: 215-898-5635
network: klein@cis.upenn.edu

abstract: Most expert systems may be described as consultative advisors that engage in interactive dialog with users in order to provide useful advice in some knowledge-intensive domain. Over the past few years, however, we have witnessed the development of a number of expert systems that operate as components of more encompassing physical environments. Examples include the intelligent control of complex physical systems such as chemical process plants, large computer installations, and nuclear power plants. We may collectively refer to such systems as *active expert systems* (AES's). This paper discusses some general design issues for building AES's that we have identified in our previous and current work.

keywords: expert systems, real-time expert systems, intelligent control, active expert systems.

On the Requirements of Active Expert Systems

Tim Finin[1] and David Klein[2]
Computer and Information Science
University of Pennsylvania
Philadelphia PA 19104
USA

1. Introduction

Most expert systems may be described as consultative advisors that engage in interactive dialog with users in order to provide useful advice in some knowledge-intensive domain. Over the past few years, however, we have witnessed the development of a number of expert systems that operate as components of more encompassing physical environments. Examples include the intelligent control of complex physical systems such as chemical plants, large computer installations and nuclear power plants. This paper discusses some of the general design issues that arise in building such *active expert systems (AES's)* based on our previous work on JESQ [6], an expert system for managing queue space in a large computer system, and our current work on architectures for *intelligent safety systems* [7].

The paper is organized as follows. Section 2 classifies expert systems in terms of how they interact with their users and with the environment, providing a basis for defining, in section 3, the class of active expert systems. Section 4 describes some of the design issues underlying the construction of AES's which distinguish them from consultative expert systems, and outlines associated implementation strategies. Section 5 concludes the paper.

2. Classifying Expert Systems by Interactions

Expert systems vary in how they interact with the world. One way to classify them is with respect to how they gather information and what they do with their conclusions. Figure 1 classifies expert systems as either *dialog driven* if they get their input from a human or *sensor driven* if they get their input directly from sensors. Note that we have in mind a rather abstract notion of a sensor as a logical device which provides information. This could be realized as a physical sensor or as a communication from another computer following some protocol. Similarly, Figure 2 shows that we can classify systems as either *advisors* if they offer their output to a human audience or as *controllers* if they direct their output directly to the physical world in the form of control signals to other devices.

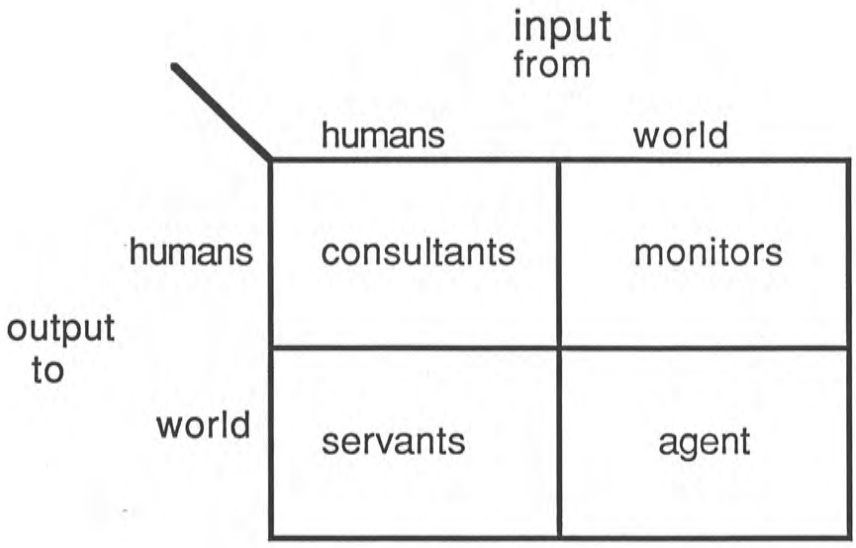
These two dimensions, input source and output destination, define a space of four basic types of systems, as shown in Figure 3: *consultants* (dialog driven advisors), *monitors* (sensor driven advisors), *servants* (dialog driven controllers) and *agents* (sensor driven controllers). In practice, of course, a given system may encompass elements of more than one type.

The majority of expert system applications to date would best be categorized as consultants. Consultants typically operate with a static description of the world which is discarded after the session has ended, and present advice or conclusions through a friendly interface. This description fits "classic" expert systems such as MYCIN [11], PROSPECTOR [2], and CASNET [13] as well as the majority of recent systems (see

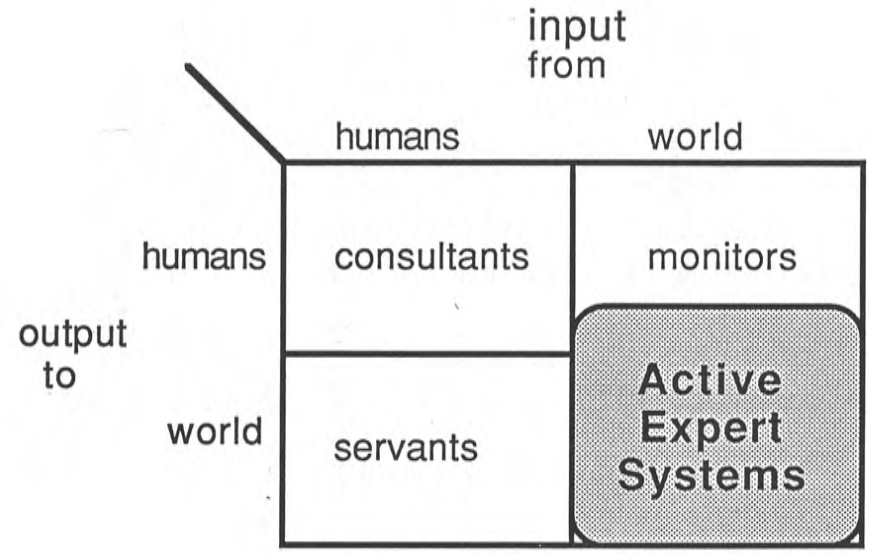
---

[1]Partial support for this research is from a grant from the Army Research Office

[2]David Klein is supported under the NASA Graduate Student Researchers Program



Expert Systems Categories
figure 3



Active Expert Systems
figure 4

[12] for a review).

During the eighties, a number of expert systems that fit into the sensor driven category were developed. These systems can be seen as components of more encompassing physical environments. Such systems typically receive information from the environment in real time and are designed to operate continuously. The original VM system [4] and the REACTOR system [9] are two examples. Most of these systems can be viewed as monitors rather than controllers which exercise any real time control over the aspects of the world being monitored.

There has also been recent interest in agents, expert systems that gather information directly from sensors in the physical world and exercise some control functions over it [8, 10, 1]. The combination of direct sensing with control closes the loop with the environment and introduces many new issues. For a variety of reasons (e.g., the experimental nature of expert systems), many expert system agents are designed so that some or all of the control functions are exercised via explicit instructions to a human. If such a system is designed to monitor for the results of the actions carried out by the human intermediaries who execute the system's recommendations with little or no modification, then it has all of the important characteristics of an agent system (even though a narrower interpretation of our classification system would describe it as a monitor system).

3. Active Expert Systems

An *active expert system (AES)* is an expert system that receives input from sensors in the environment and exerts some control functions over that environment. Control may be accomplished via direct connections to the environment or through a human intermediary who executes the expert system's advice without modification. Figure 4 depicts AES's in terms of our classification scheme.

The construction of an AES encompasses requirements not normally addressed in more typical consultative systems. While several experimental systems of both types have been constructed, there has been little work done to characterize their fundamental differences. In the following section, we identify the design issues that distinguish AES's from consultative systems, and outline strategies for accommodating some of these differences.

Throughout the discussion, we refer to a particular AES called JESQ [6] for examples. JESQ is a data-driven rule-based system that mimics the actions of computer operators who manage operating system queue space in a large computer complex. The system takes several protective and corrective actions when queue space begins to diminish, and these may be described in terms of three general goals: (1) protect remaining queue space (e.g. manipulate processors to prevent the additional deposit of data on the queue), (2) free queue space (e.g. manipulate various devices and operating system parameters to purge data from the queue), and (3) diagnose and eliminate the cause(s) of depletion (e.g. reestablish a disconnected network link). JESQ is one of several expert systems that comprise YES/MVS [3, 5], a continuous, realtime expert system for managing large computer systems. A detailed description of JESQ appears in [6].

4. Issues for Active Expert Systems

This section details some of the design considerations which are (1) general issues for AES's and (2) highlight the differences between AES's and consultative expert systems. Although we will present the nine particular considerations as independent items, each is derived from one or more of the following general characterisics of AES's: (a) AES's are inherently dynamic; (b) AES's often contain incomplete models of the systems they are attempting to control; and (c) AES's perform actions which take time to have an effect.

4.1. Maintaining a Timely and Consistent Model of the World

In many consultative expert systems, primitive facts about the world are assumed true until explicitly changed. A richer scheme is required for maintaining a model of the world in an expert system which actively solicits frequently changing information from its environment. Primitive assertions may be rendered inconsistent with the real world at any time by the actions of (1) environmental agents both known and unknown to the AES, and by (2) the AES itself. Periodic verification and reassertion of primitive facts is required in both cases.

Concerning case (1), the AES cannot generally know of changes effected by other environmental agents immediately. Assuming that we do not require all agents to notify the AES when an action is taken (i.e., values for state variables are not volunteered by all relevant environmental components), it is necessary to query the state of world periodically in order to detect such changes. As for case (2), one might assume that the AES could at least reflect the effects of its own actions in its internal model of the world, but periodic reassertion of facts is nonetheless required here as well. There are three reasons for this:

1. The action may fail due to the state of some variable that is missing from the AES's internal model of the world.
2. There may be a significant time lag between the initiation of the action by the AES and the effect of that action on the world.
3. The effects of the action's successful execution cannot be precisely computed in advance, since state variables may have changed since the time of action initiation.

Thus, the AES cannot update its internal model of the environment to reflect the anticipated effects of its actions until such effects are verified by target system responses. An AES may, however, mark those state variables that are expected to change as a result of its own actions as "unreliable". Unreliable state variables may be disqualified from consideration in making operational inferences until fresh values for them are supplied by subsequent target system responses.

In building JESQ, a part of the expert's knowledge that had to be captured was the operator's estimates of the duration over which particular information is reliable. These estimates are used by JESQ to determine when to submit additional queries to ascertain the status of the target computer system. In addition, JESQ submits fresh queries whenever it executes an action that is expected to change the status of resources, and existing information that might change as a result of the action are marked "unreliable".

Note that approximating a timely model of the world is more difficult for a human operator than for an AES. Human limitations render it difficult to keep track of all relevant state variables and to submit queries with the AES's frequency. Similarly, the mechanics of issuing the queries and processing the responses is more burdensome for human operators.

4.2. Retrying Unsuccessful Actions

In JESQ, as in many AES's, we cannot always predict the result of an action with certainty. Some control actions will fail when the system believes they should work. There are two underlying reasons for this. First, an AES must model an environment which is far too complex to represent in its entirety or in complete detail. For example, in the JESQ domain, some commands issued to devices may not successfully reach their destination, but the mechanisms responsible for this behavior are omitted from JESQ's model of the world. Second, AES applications often involve controlling an environment in which other agents are active. In JESQ's domain, for example, a user may submit a large job for printing only to delete it a short time later.

Since the effects of actions are uncertain, an AES must be designed to reflect the assumption that an unsuccessful attempt to perform an action may succeed on the next try. This is the strategy used in JESQ, which assumes that some actions which fail under a given set of conditions may later succeed

under the "same" conditions (as recorded in its internal model of the environment), since the set of conditions considered is incomplete.

This is a common human strategy as well. In many domains, it is a standard heuristic to make repeated attempts to apply some operation if it does not work at first. Knowing which operations are worth repeating is, of course, part of the expert's knowledge.

4.3. Knowledge About Temporal Requirements For Execution

The designers of consultative expert systems generally need not be overly concerned about the timeframe in which the system's advice is formulated. Of course, every system incorporates **some** assumptions of this nature (e.g. consultative systems are restricted to employing mechanisms which execute within the temporal limits of a user's patience), but they are not central implementation considerations. In contrast, the design of an expert system that deduces and performs actions in realtime must incorporate approximate knowledge about both its own processing speed and the speed of agents in its environment. Temporal estimates of concern in designing JESQ, for example, included:

- the approximate time required for self-initiated actions to take effect in the real world (e.g. how long it may take to bring a new printer on-line).
- the approximate time required for human operators and other agents in the environment to complete tasks requiring manual intervention (e.g. how long might it take the operator to physically mount a tape).
- the approximate time required for querying the state of the target resource and other resources in the environment which impact it (e.g. how long it takes to submit queries regarding the print queue and to receive responses).

In most AES environments, these speeds cannot be precisely predicted.

4.4. Scheduling the Creation of Primitive Assertions Over Time

Most consultative expert systems need not represent the notion of internally scheduled processing over specific temporal intervals. While these systems dynamically schedule the execution of satisfied rules, such scheduling is usually not performed in the context of an explicit representation of time. In contrast, an AES must schedule its own future events in terms of relative temporal intervals. In JESQ, for example, periodic queries are triggered by the timed creation of primitive assertions (e.g. "it's time for another queue space query"). As another example, JESQ warns users that their datasets may be removed in 10 minutes in the absence of explicit action on their part, and specifies the creation of a primitive assertion that those 10 minutes have passed so that it can remove such datasets.

4.5. Anticipating Problems

"Timeless" systems can employ backward chaining to determine preconditions for goal-oriented actions or conclusions, instantiating subgoals to satisfy preconditions when the principal goals are generated. In contrast, the domains of AES's generally require that preconditions for goal-oriented actions be satisfied in advance of principal goal generation, because setup actions may consume too much time to allow for the achievement of the principal goal. In JESQ's domain, for example, if a printer must be loaded with special forms to print large jobs at a time when the space remaining on the queue is extremely low (say, 3%), queue space may be exhausted by the time the forms have been loaded. Instead, JESQ would reset the forms at, say 10%, in anticipation of the critical condition, possibly before such special form output has even been generated on the queue. Thus, setup actions are performed by JESQ before the target resource reaches a critical level, even though the associated goals may never be generated.

4.6. Monitoring Expert System Support Facilities

Consultative expert systems need not reason about or account for the disappearance of the user. Since such systems do not explicitly reason about their own execution time or the timeliness of their facts, they may wait "indefinitely" for input from the user without consequence. In contrast, an AES must monitor not only the resources in its target environment (the application), but also its own support facilities. For example, the system must monitor the status of the target system interface and report its failure to the operator. In the absence of this capability, the human operator would be unaware, for example, that routine monitoring of queue space was not being successfully performed by JESQ.

4.7. Actions That Do Nothing

A realtime expert system must have a notion of "doing nothing" or idling, since there will be times when it simply has nothing to do but wait until it is appropriate to reexamine the state of the target resource. Idling in this context is defined to be an action that changes nothing in the real environment or in the internal model of the target system (except for recording the passage of time). This action must be interruptable by the receipt of data from the target system (e.g. a response to a query about an environmental resource) and by an internally scheduled event (e.g. query the target resource in five minutes).

4.8. Garbage Collection

The issue of garbage collection can be ignored in consultative systems which do not generate enough garbage in a single session to exceed reasonable space allocations. (In this context, the term **garbage** includes tokens which are no longer needed by the expert system (e.g. goals that have already been satisfied) and excludes low level entities (e.g. pointers) that are created and maintained by underlying interpreters which are transparent to the expert system). In contrast, space requirements are infinite for any AES that does not do garbage collection and generates at least one token on a periodic basis. For example, the following tokens must be deleted in JESQ:

- primitive assertions that have expired due to age;
- primitive assertions that are assumed unreliable following the execution of expert system actions;
- goals pertaining to actions that have already been performed by another resource in the environment;
- goals pertaining to actions that have already been performed by the expert system itself; and
- conflicting primitive assertions collected over different temporal intervals.

Unlike systems which collect garbage as a function of remaining space or accessibility, JESQ must delete tokens on the basis of their semantics, since it is the presence/absence of tokens which trigger inferences in the system.

4.9. Accommodating Predetermined Inputs and Outputs

Consultative expert systems receive input from and submit output (advice) to a human being. With a cognitive component participating in the formulation of input and the interpretation of output, the choice of semantics and grain size for both inputs and outputs is a relatively flexible one. That is, the consultative framework provides the knowledge engineer with the flexibility to distribute the required domain reasoning between the user and the expert system as appropriate.

In contrast, AES's are performers. They are active agents that receive inputs from environmental resources and directly manipulate them. This limits the semantics and grain size of expert system input and output to the those dictated by the resources in the environment. In general, inputs and outputs are

more detailed and precise in JESQ's domain than those typical of consultative systems. They are low-level data (i.e., operating system responses and commands) rather than high-level opinions. Thus, the design of AES's requires building reasoning strategies around the interfaces of existing systems, in contrast to consultative expert systems.

5. Conclusions

The construction of AES's encompasses requirements not normally addressed in building consultative expert systems. We have identified several such requirements in this paper, along with the strategies for meeting them as implemented in JESQ. More experience with AES's will undoubtedly uncover additional design issues, and it is hoped that their treatment in the spirit of this paper will lead to a methodology for building them.

References

1. Astrom, K., Anton,J., Arzen, K. "Expert Control". *Automatica 22*, 3 (May 1986), 1 - 10.

2. Duda, R., Gaschnig, J., Hart, P. Model Design in the PROSPECTOR Consultant System for Mineral Exploration. In *Expert Systems in the Micro-electronic Age*, D. Michie, Ed., Edinburgh University Press, Edinburgh, 1979.

3. Ennis, R. et. al. "A Continuous Realtime Expert System for Computer Operations". *IBM Journal of Research and Development* (January 1986).

4. Fagin, L. *VM: Representing TIme-Dependent Relations in a Medical Setting.* Ph.D. Th., Stanford University, June 1980.

5. Griesmer, J.H., et. al. YES/MVS: A Continuous Real Time Expert System. Proceedings of the National Conference in Artificial Intelligence, AAAI, August, 1984, pp. 130-136.

6. Klein, D. An Expert Systems Approach to Realtime, Active Management of a Target Resource. Master Th., University of Pennsylvania,1985. available as technical report MS-CIS-85-40.

7. Klein, D., Finin, T. What's in a Deep Model? A Characterization of Knowledge Depth in Intelligent Safety Systems. technical report MC-CIS-87-02, Computer and Information Science, University of Pennsylvania, January, 1987.

8. Moore, R.L., Hawkinson, L., Knickerbocker, C., Churchman,L. A Real-time Expert System for Process Control. Proceedings of the First Conference on Artificial Intelligence Applications, IEEE, December, 1984, pp. 569-576.

9. Nelson, W. REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents. Proceedings of the Second Annual National Conference in Artificial Intelligence, AAAI, August, 1982, pp. 296-301.

10. Sachs, P., Patterson, A., Turner, M. "Escort - An Expert System for Complex Operations in Real Time". *Expert Systems 3*, 1 (January 1986), 22 - 29.

11. Shortliffe, E.. *Computer-based Medical Consultations: MYCIN.* Elsevier, New York, 1976.

12. Waterman, D.. *A Guide to Expert Systems.* Addison-Wesley, 1985.

13. Weiss, S., Kulikowski, C., Safir, A. "A Model-based Method for Computer-aided Medical Decision-making". *Artificial Intelligence 11* (1978), 145-172.