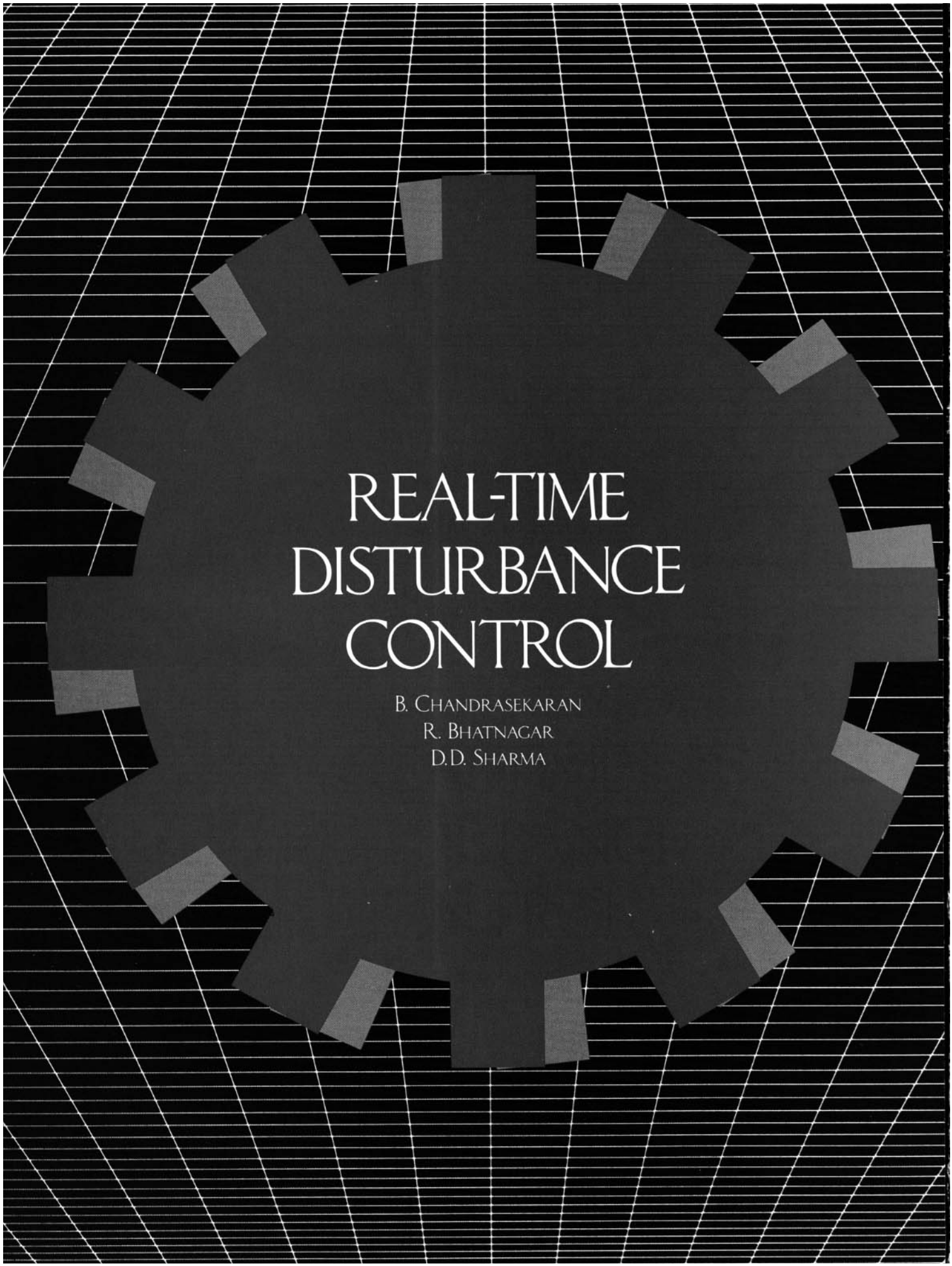
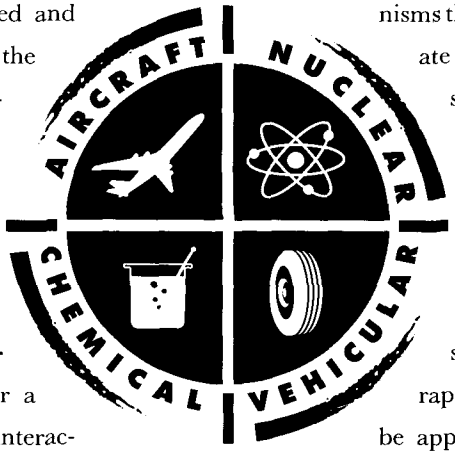

REAL-TIME
DISTURBANCE
CONTROL
B. CHANDRASEKARAN
R. BHATNAGAR
D.D. SHARMA

For a number of years we have been investigating a family of real-time problems that we call *real-time disturbance control problems.* Operating a complex process such as a power plant or a chemical refinery as an internal or external disturbance develops is an example of this type of real-time problem. Along with our colleagues, we have built a series of knowledge-based advisory programs for assisting operators of such plants [2, 6, 12]. Our experience with the task in these domains suggests to us that success in real-time disturbance control arises less from complex problem-solving methods specific to the real-time problem than from the way the process system itself is designed, the task is decomposed, procedures are compiled so the subtasks can be readily solved, and goals are judiciously abandoned and replaced. In this article, we study the nature of the task, propose some general architectural principles for this class of problems, and describe a real-time knowledge-based control system called Operator Advisor which is a partial embodiment of the principles. ✲ **Real-Time Disturbance Control Problems** Consider a controller-operator (or just operator) interacting with a dynamic physical process system. A description, or model, of the process system will be in terms of a set of time-varying parameters of interest, which together constitute a state vector (e.g., output power level, core temperature, *valve A opening*). These parameters can be at differing grain-sizes of time, and will have causal relations among them, i.e., some of the parameters can be thought of as causes of other parameters. The process system model may include an account of these causal relations as well. ✲ The values of a subset of the state parameters are available to the operator as the set of *observations.* The operator is given certain goals regarding the process systems. These goals are defined as predicates over functions of various state variables of the system (e.g., power level to be less than 50 MW, total escaped radiation over the next hour to be less than R units). ✲ The operator has access to a set of action primitives. The state of the process system can be changed as a result of the application of these actions. The control task for the operator is to synthesize a sequence of action primitives so that the process system is maintained in or brought to states that satisfy the goals. Specifically, the disturbance control task arises when events—external or internal to the process system—cause changes in its state that are great enough to threaten the goals or prevent them from being satisfied. ✲ In order to generate the action sequence, the operator needs to have an action-generation scheme by which the observations are mapped into actions. These schemes can range from servo-mechanisms that adjust control parameters or initiate control actions, to problem-solving systems that work with explicit symbolic models of the process system and synthesize suitable plans of action. Of course, a combination of action-generation schemes can be used as well. ✲ The state of the process system may change rapidly, and the generated action may not be appropriate unless it is generated and executed quickly. Secondly, if actions are generated before the goals are actually violated (for example, in response to observations that indicate possible future violation of goals), the process system may be kept in a satisfactory state without any violation of goals. Thus, actions must be generated and executed fast enough for goals to be achieved. This is what makes the control task real time. ✲ What is the relation between time available and the quality of the action generated? Not all action-generation schemes have the property of providing a better solution when increased time is available. For example, servo-mechanisms take a fixed time to respond; they are not structured to trade off decision quality for available time. On the other hand, for action-generation schemes that search for a solution in some space of possi-



bilities, the likelihood of a satisfactory solution increases in proportion to greater time availability, assuming, of course, the problem space and the search strategies are appropriate.

A central property of action-generation schemes is they are based on explicit or implicit models of the system to be controlled. However, models are, at best, only partial models of real physical systems. Further, even with respect to the given model, observations offer only partial information about the state variables included in the model. These two facts mean there is no action-generation scheme for which there is a guarantee that the actions generated will achieve the goal, even if time available to solve the problem is unbounded. To the extent that the goals must be achieved within a time bound, all finite-resource agents making plans to control real-world environments are facing a real-time problem. If the task is to generate actions that are sure to achieve the goals, the achievement of the task simply cannot be ensured, even if we increase the speed of processors involved in computing actions, or find better models of the process system.

Thus in practice, real-time control problems must be conceived in terms of accepting deviations from ideal goals. Notions of goal abandonment and goal substitution are important in this regard. Goals which conflict with higher-priority goals may be abandoned. Goals which have higher preference but are not being achieved can be replaced by goals which are less preferred, but more likely to be achieved in return for higher costs. Goal abandonment and substitution are important means by which graceful degradation of the behavior of real-time systems can be achieved. The control task is now to generate actions and adopt goals in order to minimize the costs to the extent possible.

Some Principles for Good Real-Time Disturbance Control

There are many domains in the world in which real-time disturbance control is being achieved more or less successfully by using a combination of automated systems and well-trained human operators. Power plants and chemical refineries are operated and airplanes are flown on a regular basis. We found it instructive to investigate how such control is achieved in these domains in spite of the limited processing capabilities of human controllers. The most explicit and public documentation of control procedures is available in the area of operation of nuclear power plants, but in our view the principles that underlie these procedures are more generic in character, and underlie control of many other dynamic processes.

The general principles we have extracted from our study are:

Sensor system design. Sensor systems should be designed to allow a relatively direct mapping from sensor readings to internal states that are related to threats to important goals. Actions can now be indexed over these internal states, reducing the complexity of the observations-to-action mapping. However, not all potentially relevant states can be detected in this manner without a proliferation of sensors.

Action system design. Action primitives design should have a direct or almost-direct relation to achieving or maintaining the more important goals. For example, since keeping the reactor temperature below limits is a very important goal, the action system in nuclear power plants is capable of injecting cooling water. As with sensors, accomplishing all the desired goals directly with very few actions will result in the proliferation of action primitives.

Compiled goal-ordering knowledge. The goals are designed with explicit compilation and priority and preference relations among them; thus if any of the goals fail, decisions can be made immediately about which goals should be substituted.

Compiled corrective procedures. There must be explicit compilation of procedures (or plans, or sequences of actions) to follow for numerous clearly identifiable internal states which can threaten goals. That is, much of the planning is reactive.

Throughout the rest of this article, we assume the sensor and action systems of the process systems to be controlled are well designed in the sense described here.

Relations Among Goals

It is useful to distinguish among the following possible relations between the goals in the operation of a process system.

Priority relations. Some goals have higher priority than other goals. For example, in the operation of complex equipment with potential to cause damage, a clear distinction is often made between safety goals and operating goals—the former having higher priority over the latter. But such priority relations may exist between operating goals as well. Given two goals with such a priority relationship, both goals are to be pursued under normal conditions, but if the pursuit of a higher-priority goal conflicts with meeting a lower priority goal, the latter goal is to be abandoned. Consider driving an automobile as an everyday example of real-time control. The goal of "keep engine from overheating" generally has higher priority than the goal of "reach destination by time T."

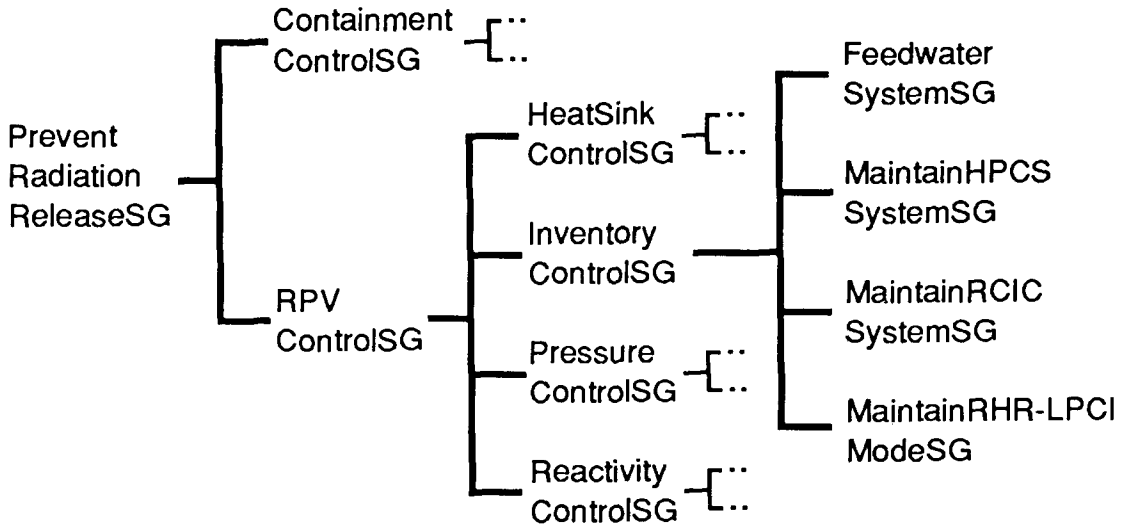
Preference relations. There may be preference ordering among goals in subsets of the set of goals. At any time only one goal from this set—the one with the highest preference that is thought to be achievable—will be operational, unlike the goals with priority relations. If it appears

that this goal cannot be achieved, it can be substituted by a less preferred but more achievable goal if one is available, or it will be abandoned. To continue with the automobile example, the goal "reach the destination by T + 2 hours" may be achievable and thus may be substituted for the original goal, even though the new goal is less preferred.

Subgoal relations. A goal **g** may be achieved by achieving **g1** and **g2,** which are the subgoals of **g.** But—unlike subgoals in problem solving in general—**g1** and **g2** may not be present in the goal set merely as means to achieve **g,** but may be important in their own right. However, if **g1** or **g2** is unachievable for any reason, it may be abandoned and other methods which do not involve **g1** and **g2** as subgoals may be adopted to achieve **g.** Such subgoal relations are often used to organize the achievement of safety goals. The safety function hierarchy for the nuclear power plant domain is an example of such an organization and a part of it is shown in Figure 1. A safety goal at any level of the hierarchy is achieved if its successor safety goals are achieved. The leaf-level goals are maintained as part of normal operating conditions, thus achieving all the goals in the safety function hierarchy. If a disturbance develops, specific plans to maintain only those safety goals that are directly threatened are invoked. If any of these fail, then alternate plans for achieving their parent safety goals are adopted (i.e., plans that do not depend on the achievement of the subgoal that failed), giving the name "defense-in-depth" to this strategy.

Normally, systems are designed so the safety goals are maintained as part of normal operation (i.e., as operating goals are met, the safety goals are met automatically). When a safety goal is threatened, the plans associated with it typically maintain the safety goal at the cost of some operational goals. As one goes higher in the safety hierarchy, the corresponding plans sacrifice more and more operating goals. This is because the plan is invoked only when the plans for one-or-more children safety goals have already failed, calling for more drastic ways of maintaining the deeper-level safety goal. This generally calls for further sacrifices in operating goals. In this sense, a safety goal is easier to achieve than its child, but at increasing costs in operational goals.

FIGURE 1.
A Partial Safety Function Hierarchy for a Nuclear Power Plant

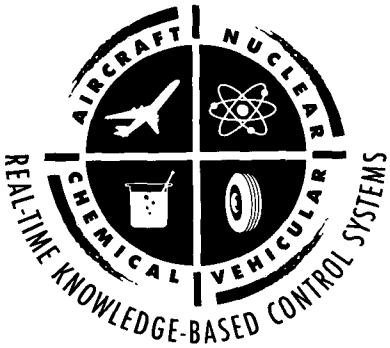


Design-Time vs. Run-Time Aspects of Control

The goals and their ordering, as well as the procedures to be adopted for achieving the goals, are, as indicated, determined at design time. What aspects of the control behavior are then determined dynamically? To see this, it is useful to note that there are three aspects to the trade-offs that lie at the heart of real-time control:

1. The amount of compute time taken to generate and implement actions.
2. As the time in 1 increases, the undesirability of the control goal that can in fact be achieved decreases. That is, it is likely that we will be able to achieve goals closer to the ideal ones.
3. As the time in 1 increases, the undesirability of the plant state that is likely to be reached increases.

Ideally, one would like to allocate just enough computing time that an optimal balance is struck between the undesirabilities in 2 and 3 (intuitively, where the curves in 2 and 3



cross). A main point is that this trade-off assessment need not (and often in this domain should not) be performed at run-time, but that the compiled goal-ordering and procedures should reflect design-time optimization of the trade-off. What is determined dynamically is the choice of the actual goal. For example, once a procedure for a goal has been put into effect, if the process system does not respond as expected within a certain time (both the time limit and the expectation are to be specified in the procedure), that goal will need to be abandoned and replaced with a less desirable, and presumably more achievable, goal.

Task Decomposition

The complexity of directly mapping from observations to actions can be high: for even a moderate-sized set of observations, the number of combinations to provide good coverage can be prohibitive. The task is generally decomposed into stages to reduce the complexity. If the problem is not real time, the ideal strategy is to decompose the task into two stages: diagnose the cause of the disturbance; and generate actions to remove, or compensate for the effects of, the causes. But diagnosis can take an open-ended amount of time, and hence is not always possible for real-time control. The strategy that has evolved in a number of domains is to decompose the task as follows:

1. Map from observations to violations of, or threats to goals,
2. Map from threats/violations to action plans,
3. Monitor plan execution, and
4. Modify plan and goals as appropriate.

In subtask 1, if the goals are preenumerated, the process system can be equipped with sensors to detect goal violations in a relatively direct way. However, detecting *threats* to goals is more open-ended, since it involves prediction—a complex problem-solving task in general. In practice, the more immediate threats can be detected directly. As mentioned earlier, the design of sensors for a process system often evolves over time to achieve better detection of goal violations and threats. For example, one of the consequences of the Three-Mile Island accident was the introduction of a new sensor subsystem that could detect the reactor being in a potentially unsafe operating region. This information can be used to predict (much earlier than was previously possible) a threat to the goal relating to reactor temperature.

Subtasks 2, 3, 4 can also be quite complex to solve in an optimal way. The normal strategy is to precompile procedures for countering each threat to goals. Much thought is given offline to the development of procedures having a high likelihood of success. Since availability of diagnostic information cannot be ensured at the time actions are being generated, the procedures and action primitives are generally designed to compensate for the effects of the disturbance rather than to remove the cause. For example, consider a threat to the (safety) goal of keeping the reactor temperature below safe limits. Perhaps the real reason for the disturbance is a stuck valve. Identifying this is a complex diagnostic task; and even if the reason for the disturbance can be identified, the action required is to open the valve, which may be located in a relatively inaccessible place. This disturbance will be handled by turning on pumps to supply additional cooling water, an action that does not require diagnosis.

It should be noted that all the subtasks, even with a good process system design and compiled procedures, are still open-ended (i.e., there is no guarantee that the subtask can be accomplished optimally within a given time). We already indicated how the task of detecting potential violations can involve extensive amounts of predictive problem solving. Generating an action plan to ensure satisfactory control can be equally complex even with compiled procedures. Perhaps a compiled procedure that is recommended will fail because of an additional malfunction that is present but that was not taken into account during the compilation of the procedure. If more than one disturbance is present, perhaps the compiled procedures for them interfere with each others' effectiveness. And so on. Thus the subtasks, and hence the goal, may fail. This is precisely why the abandonment and substitution of goals are essential components of this approach to real-time control. Identification of what goals to substitute may itself be the subject of run-time problem solving; but there are considerable advantages to precompiling the preference and priority relations among goals.

The fact that so much of the knowledge needed is precompiled in many domains is another instance of the dictum "In the knowledge lies the power." However, a task analysis such as the one we have performed is needed to understand what knowledge plays what role and hence what knowledge ought to be compiled.

Knowledge-Based Systems for Assisting Operators

Some of the principles just outlined (even though they arose in the context of human operators with limited processing power) are valid for providing computational assistance in generating control actions as well. For example, well-designed sensor and action systems that help do the mappings as directly as possible will be just as valuable for computer-based planning systems. The fact that even with vast computational resources the actions may still fail to achieve the goals means that goal substitution will still be

essential. But how about compiled procedures for threat identification and plan generation? Would it not be better to use additional computing power to perform a more thorough analysis of the problem and to explore the plan space?

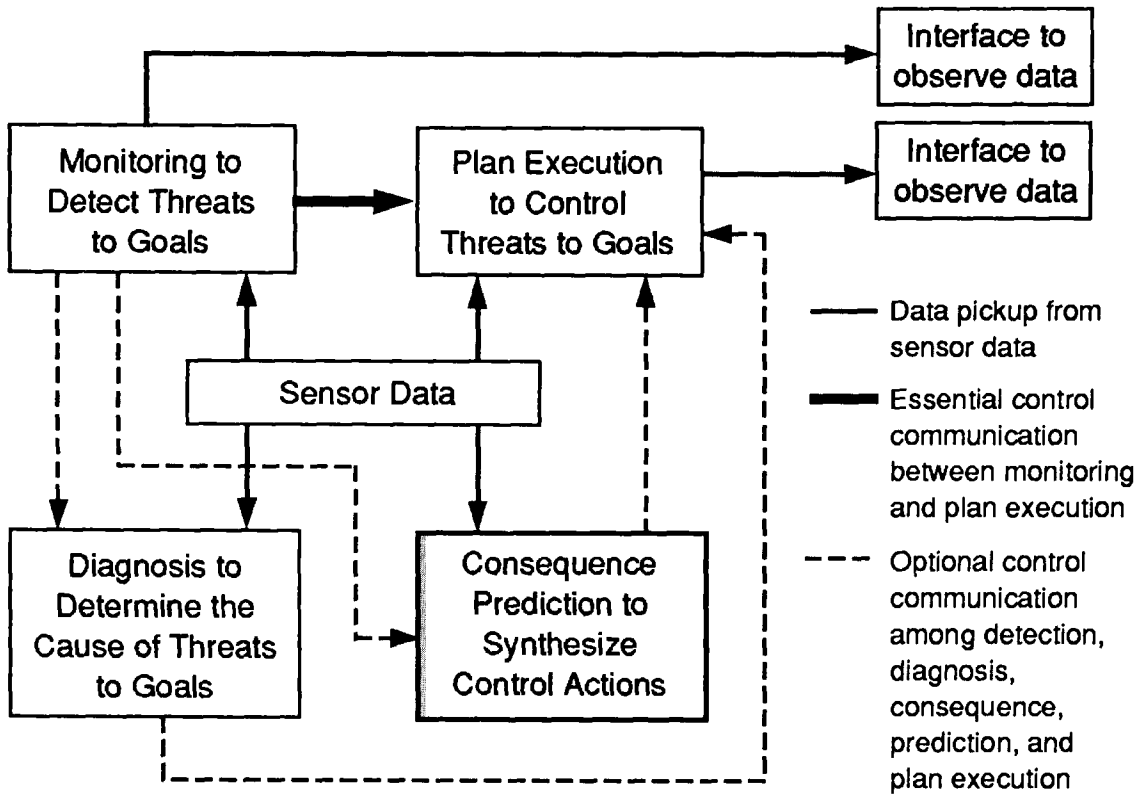
Certainly additional computational power can be devoted to implementing the compiled mappings (from sensors to threats and from threats to plans) more effectively. Even with well-designed sensor and action systems and well-defined procedures and goal hierarchies, the human operator's short-term memory is severely stressed, and the procedures are often executed suboptimally. The likelihood that the goal is achieved is decreased, leading to premature resort to less preferred goals. A knowledge-based assistant can implement more complex compiled procedures than a human operator and can keep track of a larger number of parameters at one time and their interrelations. For example, such a system can perform a more reliable detection of conflicts among plans in case of a multiple disturbance. Thus there is an increased likelihood that an advisory system can help achieve a given goal.

Additional problem solving can also be performed in parallel for a "deeper" analysis of the disturbance and more sophisticated planning, without these capabilities being on the critical path to the performance to any of the subtasks. Success in these parallel problem-solving activities can improve the performance on the subtasks—that is, increase the likelihood that the current goal is met and an important goal is not abandoned or a less preferred goal substituted. Specifically, the following open-ended problem-solving activities can be undertaken in parallel:

Diagnosis of the disturbance. This information, when available, may be used by the plan evaluation subtask. If the plan assumes the availability of certain subsystems, and it is known by diagnostic analysis that these subsystems are malfunctioning, that plan can be abandoned, and an alternate one may be invoked. Diagnosis may also help in the detection, at an earlier stage, of additional threats to goals. Finally, the results of a diagnosis effort can be used for fixing the problem off line. For example, if a loss-of-cooling event is due to an open valve, even though the immediate control action may be to inject cooling water, having the diagnostic result makes it possible to take actions to fix the valve.

Prediction of system state. Using observations and any diagnostic information, simulations can go on

FIGURE 2.
The Overall Architecture of a Disturbance Control System



in parallel to detect additional threats to goals beyond those that can be detected by direct mapping from observations.

Plan evaluation and conflict detection. The compiled knowledge approach can, at best, do minimal checking for conflicts. The task really involves complex simulation. The results of this task done in parallel can be used to abort plans and to start other plans, and for goal abandonment/substitution decisions.

Figure 2 illustrates the architecture that is proposed as a way of using knowledge-based reasoning to significantly enhance the quality of real-time control. We will now present the design and performance of a system called the Operator Advisor that has been constructed using these principles and the architecture. We describe mostly those aspects of the Operator Advisor that use compiled knowledge in the performance of its various subtasks. A minimal diagnostic capability has been implemented and will be described briefly.

Action generation by relating actions to states of the world (rather than plan generation by abstract problem solving with complex world models) is an important emerging area of study in AI. Agre and Chapman [1], Rosenschein and Kaelbling [11], Schoppers [12] and Hayes-Roth [8] provide distinct approaches and perspectives on this subject. The use of compiled procedures for real-time disturbance control is related to the ideas on reaction (or universal) plans [11]. However, a number of distinctions ought to be noted as well. Hayes-Roth describes a complex architecture in which perception and deliberative planning are integrated. Agre and Chapman use perceptual markings in the world by the agent as a way of reducing plan complexity. Rosenschein and Kaelbling describe automatic compilation of reactive or situated machines from a set of formal specifications of behavior in such a manner that the machine's behavior can be proved to be correct. The use of compiled procedures here can no doubt be improved considerably by incorporating the insights that are developing as a result of this family of investigations.

Kim and Modarres [10] describe the use of a type of goal tree in controlling process systems. The goal tree in this work captures the traditional goal-subgoal relations (i.e., the subgoals have to be achieved in order to achieve the goals). In our work on the other hand, the safety goal hierarchy represents the goal-subgoal relations under normal operating conditions, but is also used to represent the ordering of goals so decisions about goal abandonment can be made as well, in case goals in the lower levels of the hierarchy cannot be achieved for any reason.

Operator Advisor: A Disturbance Control System for the NPP Domain

Here we describe the knowledge, design, and implementation of an Integrated Operator Advisor (OA) system. The OA is the latest in the series of prototype knowledge-based systems we have built [2, 7, 13] for the operation and safety maintenance of the Nuclear Power Plant (NPP) domain. The NPP domain is a highly regulated domain with a considerable body of compiled prescriptive procedures, and OA exploits the established structures of knowledge in the industry. The distinctions that have been made by the industry make considerable sense from the viewpoint of real-time problem solving, even though they were developed independently of any possible application in an automated advisory system.

Task of Mapping from Sensor Values to Threats to Goals

In the nuclear industry the direct threats to safety goals are called safety threats and threats to operational goals are called abnormal events or, simply, events. The selection and deployment of sensors for the process system itself are driven by the need to identify *events* and *safety threats* simply and directly.

In theory, plant safety can be achieved through the maintenance of certain plant conditions, quite independent of the plant operation. For example, consider the safety goal of keeping the radiation level at less than a certain value. In an abnormal situation the radiation level can be maintained below this value by lowering the power level, or in the extreme case by shutting down the plant, without expending any effort in identifying the cause of the increased radiation level.

The safety threats are organized in the Safety Function Hierarchy (Figure 1). Each node in the Safety Function Hierarchy represents a safety goal that would be lost if the corresponding safety function (or goal) cannot be maintained. Each node in the hierarchy contains the sensor conditions that identify the threat to that particular safety goal.

While in principle not all possible events can be precompiled or detected by simple pattern matching on sensors, a set of events have been chosen by the regulatory agencies either on the basis of their potential seriousness or the frequency of their occurrence. For example, the so-called Loss-of-Coolant Accident (LOCA) is one such event. The occurrence of the event indicates a threat to the operational goal of maintaining power because of lack of sufficient coolant for normal power generation. If not controlled, the event can develop into a threat to the safety goal of maintaining water level within safety limits. During a LOCA, it is required from the operational perspective that some immediate actions be taken to restore the loss of coolant to maintain power. In the

NPP domain there are more than 20 such events.

The main difference between safety threats and events is that in the case of events the concern is about restoring as many of the operational goals as possible while maintaining safety goals.

Detection of Events and Threats to Safety Goals

In the OA safety threats are detected by monitoring their identifying conditions stored in the nodes of the Safety Function Hierarchy. A top-down Establish-Reject strategy is used to go through the hierarchy.

The Establish-Reject strategy consists of pattern matching identifying conditions in a parent node, representing a safety threat, with the current sensor values. If there is a match, the corresponding threat is established. After establishing the node, the monitoring process moves to the sibling node, rather than going down the established node. This is both possible and required because of the relationship among the plans for safety threats. It is possible because if a parent and its child threats are established, then only the plan for the parent threats needs to be executed, since the plan for an established parent node is designed to take care of all the safety goals below it. It is required because once a parent-level threat is established, attention should be given to threats at the same level, rather than trying to identify less important threats.

The Establish-Reject strategy of going through the Safety Function Hierarchy is a conservative (priority is given to major threats) and efficient way of detecting threats to safety goals. Detection of the events is done by a pattern matching of the conditions identifying the events, with the current sensor values. The events are monitored according to the total ordering given in the list of events.

Procedure Execution to Control Threats

In the nuclear industry, what we have been calling plans are called procedures—a term we will adopt in the remainder of the discussion on OA. The procedure execution task has the following components:

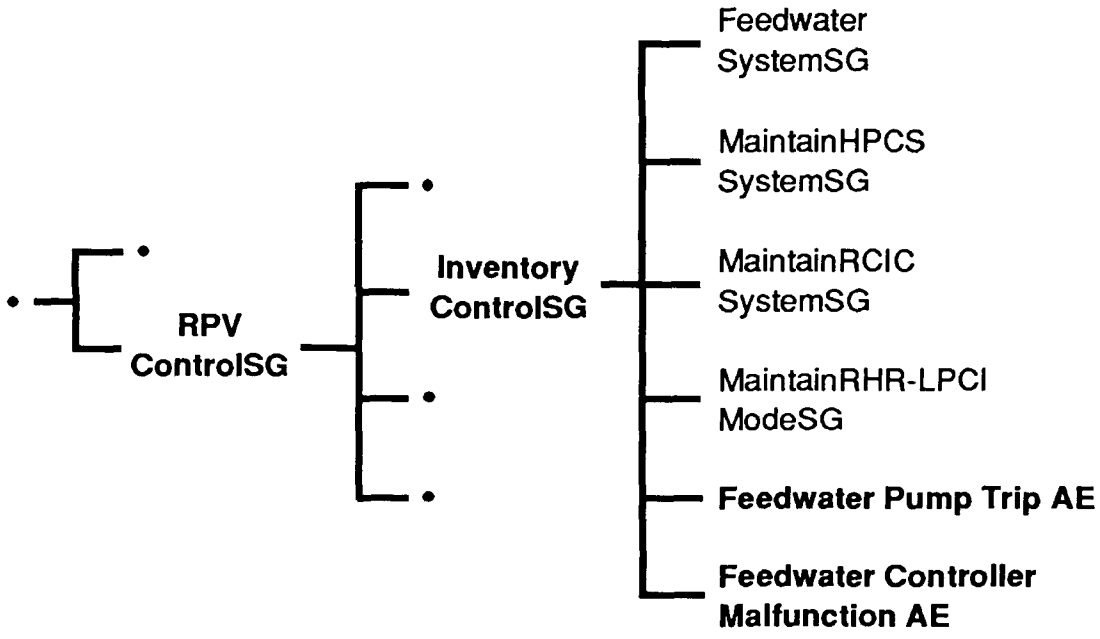
(a) Retrieve procedures for all the threats and detect conflicts
(b) Select achievable, nonconflicting procedures
(c) Initiate and monitor procedures and go to (a) as needed
(d) If a chosen procedure fails, abandon/substitute goal and modify procedure or go to (a) for retrieving new procedure.

The event procedures, while restoring the operational goals, are also designed to prevent the development of safety threats. However, the procedures for safety threats may involve loss of operational goals.

Relations Among Procedures

As discussed in the section on relations among goals, the goals can have priority, preference and subgoal relations among them. In OA, the representation and organization of procedures reflect the relationship of the associated goals. Priority of safety over operational goals means that when both a safety and an event procedure are retrieved, OA will choose a safety procedure first. Similarly, associ-

FIGURE 3.
Example of a Part of the Integrated Procedure Hierarchy



ated with each event procedure in OA are pointers to safety goals that would be threatened if the event procedure fails. This enables OA to abandon the event procedure (i.e., the operational goal) and initiate the safety procedures (after additional conflict resolution among them, as appropriate). The subgoal relation between safety goals means that OA will abandon a safety goal if it cannot be achieved; this means that its parent goal will be threatened, an OA will retrieve and attempt to execute the independent safety procedure for the parent goal. Similarly, if two safety goals, one a descendant of another (*à la* the safety function hierarchy, Figure 1), are both threatened, OA will first execute the safety procedure for the "deeper" safety threat (i.e., the one closer to the root), and the shallower safety procedure will be considered later.

In OA, there is no separate representation of goals and their relations. The event and safety procedures are organized into a single hierarchy which fully compiles the partial order among the goals, and hence the procedures, into an integrated procedure hierarchy (IPH).

Figure 3 illustrates a portion of the Integrated Procedures Hierarchy, which is an elaboration of a fragment of Figure 1, the safety function hierarchy. Each node in the IPH is named after a safety goal or an event, and it stands for the corresponding procedure. The interpretation of the link from a child to the parent is as follows: If the procedure for the child fails, invoke the procedure for the parent. We see that two events have been added as children of the InventoryControl SG node. The procedures for these events have the property that if they fail, the InventoryControl safety goal will be threatened. The integrated procedure hierarchy explicitly records this relationship between an event procedure and the safety goal that the procedure's failure endangers. Thus if the children safety goal procedures or event procedures fail, the integrated procedure hierarchy directly points to the procedures to be invoked, without any additional reasoning.

TABLE 1

Knowledge Representation of the Control Procedure

```
(Control-Procedure-For      FeedwaterPumpTripAE)
(Procedure-Name      FeedwaterPumpTripAE-Planner)
(Procedure-For-Safety-Goal      InventoryControlSG-Planner)
```

TABLE 2

Knowledge Representation of the Contents of a Procedure

```
(Procedure-Name-FeedwaterPumpTripAE-Planner)
(Prerequisites None)
(Criteria      (AND(Is the RPVLevel < 197 inches)
               (Is The RPVLevel > 2 inches)
     (IsTheNumericalValueOf ReactorPower < 4 percent))
(Sub-Procedures ONI-N27)
(Relation-Among-Sub-Procedure Sequential)
```

Knowledge Representation of Procedures

A task-specific language was developed for procedure representation [3]. (For the advantages of task-specific languages, see [5]). The procedures in OA are based on the corresponding ones compiled in the domain. The procedure contains several types of information. At the top level, the following information is represented: the event or safety threat for which the procedure is meant, and any safety procedures that should be invoked if the procedure fails (implicitly representing the safety goal that will be threatened by the failure). Table 1 gives an example of this.

At the next level the contents of the procedure are represented as prerequisites for its execution; criteria for its success; subprocedures and relations among them.

Table 2 is an example of this. Most of the information in this example is self-explanatory, but the representation of relations among subprocedures may need some explanation. Possible relations are Sequential (the subprocedures must be executed in sequence and all must succeed); BackUP (the subprocedures are ordered alternatives, an alternative is executed only if the main subprocedure fails); Parallel (subprocedures are to be executed concurrently, this feature is currently under implementation); ManyOf (as many of the subprocedures as possible need to be executed to increase chances of success); Monitor (all the subprocedures must be executed within a specified amount of time or within a specified number of repetitions). While we have been able to implement the procedures of interest using these, no claim of completeness is made for them.

Additional representational primitives for procedures include actual procedure steps. Since representation of these steps is fairly straightforward, we omit a discussion of them.

Temporal Information in Procedures

General-purpose planning systems need to have a fairly sophisticated temporal reasoning capability (e.g., for reasoning about plan conflicts and interactions). There is no general-purpose temporal representation in the procedures that we just presented. Instead, the needed information is explicitly compiled in the procedures. For example, the procedures explicitly contain information about how soon after the initiation of the procedure the monitoring should be done for evidence of success or failure, and about potential plan conflicts. If the information is incomplete, as is likely to be the case in compiled procedures, there is a strong chance the procedure will fail. A more achievable goal will then be chosen.

Execution of Procedures

Conflict resolution is the first step, after all the procedures corresponding to events and safety threats are retrieved. If P1 and P2 are a pair of procedures in a parent-child relationship in the integrated procedure hierarchy (see Figure 3), OA makes the following conflict resolution choices:

- If P1 and P2 are both retrieved, only P1 needs to be executed.
- If during execution of P2, a situation arises in which P1 is needed, P2 is suspended to initiate P1.
- If P2 is not successful, P1 is initiated.

The conflict resolution performed in this step is actually only a part of reasoning about plan conflicts. Other kinds of conflicts may exist, being more or less difficult to detect. For example, a step in Plan A might turn a valve on, while a step for Plan B might turn the same valve off. Detecting this would require running through the plan steps to make sure they do not have undesirable interactions. A more complicated example is one in which the future effects of a step in Plan A undo the conditions for success of Plan B. Arbitrarily complex simulation capabilities will be needed to ensure that no plan conflicts or deadlocks remain. While OA's conflict detection capabilities can be improved, solving this subtask in a guaranteed way within a fixed time is in principle not possible.

Plan Execution, Monitoring and Modification

As the name of the task suggests, this task itself consists of three subtasks. The first subtask is Plan Execution and its objective is to guide the operator through the steps of the identified plan. The second subtask is Plan Monitoring, and its objective is to continuously monitor the success of the executed steps. The third subtask is Plan Modification, and its objective is to provide safety maintenance when the executing procedure fails completely (i.e., the primary and the alternative actions are not successful). The objective of the overall task is to recover from the abnormal situation by maintaining plant operation and safety, with the priority being on safety. That is, maintain safety even if normal operation is not possible. Plan Execution is performed by displaying the actions given in the procedures and Plan Monitoring is performed by verifying the expected effects' executed actions. The expected effects are also available in the procedures.

Plan Modification is required only when a procedure fails completely (i.e., the primary and alternative actions are not available or successful). In this situation, the executing procedure is modified to maintain safety by initiating the procedure for the safety goal associated with the failed procedure. Modification of the unsuccessful procedures essentially provides graceful degradation by giving up on the more preferable goals that cannot be pursued because of time limitations, and pursuing goals that can be achieved within the time constraints.

While we have not discussed procedure abandonment, it is a subtask that requires at least some mention. When procedures are abandoned, clearly a number of activities are called for to shut down the procedure as gracefully as possible; that is, the effects of actions already taken are noted, any changes that are no longer useful are reversed if possible, and so on. Again, this is a task that could use substantial problem solving in its own right, but OA merely uses whatever explicit steps are suggested by the domain procedures for this. In this task, as in all others, OA tries to use compiled knowledge if it is explicitly available—but simply depends on the fact that goal substitution and abandonment will help to reduce functionality gracefully rather than radically, in case of failure to perform the subtask.

Optional Tasks

We discussed the possible roles that can be played by additional problem solving in parallel. The current implementation of OA includes only a simple diagnostic component that works in parallel. The diagnostic task is initiated only after a threat has been identified.

The diagnostic task, after finding the cause(s) of the detected event or safety threat, can interrupt the task of Plan Execution to Control Threats. The information provided by the diagnostic task can help in recovering some of the operation goals earlier, while executing the control actions.

As an example of how the information from the diagnosis module can be useful, consider the following:

The event FeedwaterPumpTrip is identified and the procedure to control it is initiated. The diagnostic task also is initiated, in parallel, to determine the cause of Feed-

waterPumpTrip. The diagnosis determines the cause of the FeedwaterPumpTrip to be Low Net Positive Suction Head (NPSH). Once the cause Low NPSH is known, the pump suction head can be increased by adjusting the pump speed. With this action the tripped pump will start functioning again. This will lead to earlier recovery of feedwater flow and may not require the adjustment of recirculation flow, which is the independent and alternate way of increasing the flow without knowing the cause, but this also affects the power generation.

After the correction of the cause, some actions may, in fact, have to be undone, and certain other actions may not be required. The procedures contain knowledge that call for certain conditions (given in the slot Prerequisites) to be checked before a procedure or an action is initiated. These conditions, in case the faults are corrected, will prevent the execution of some unnecessary and less efficient actions.

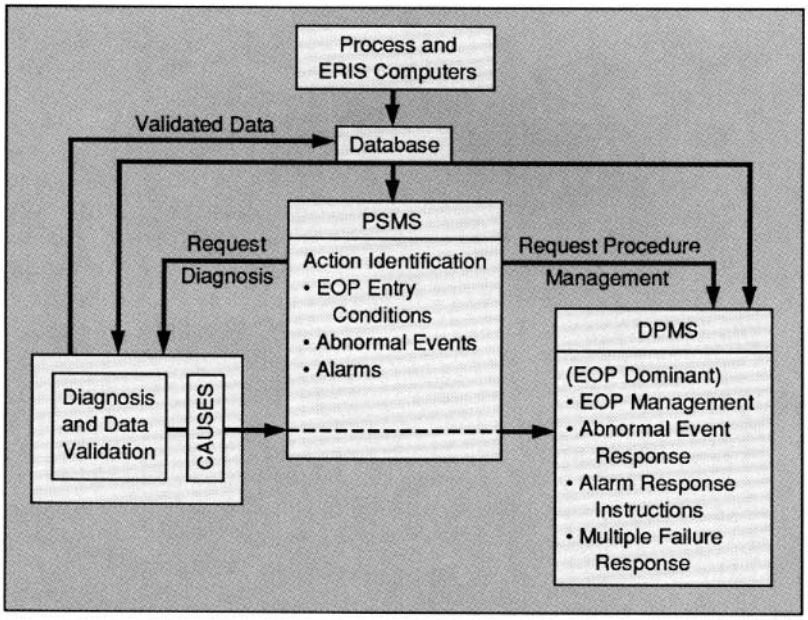
The diagnostic task is implemented in the language *Conceptual Structures Representation Language* (CSRL) developed for the task of hierarchical classification [4, 5] and is reported in detail in [7].

Present Implementation of the Integrated Operator Advisor System

The present implementation of the Integrated Operator Advisor System [2] consists of the following four modules: an "Intelligent" (Knowledge-Based) Database; a Plant Status Monitoring System; a Dynamic Procedure Management System; and a Diagnostic and Data Validation System.

The Intelligent Database contains the real-time plant data and the knowledge to interpret it. Interpretation mainly consists of *data abstraction* (raising the abstraction level from that of the raw sensors to one that can be directly used by the various procedures). This is a knowledge-based system, based on the architecture proposed in [9] and is described in detail in [2]. The database is updated continuously to reflect the current plant conditions. The other three modules address the database to monitor plant conditions and evaluate the success of the action.

FIGURE 4.
The Overall Architecture of the Operator Advisor System



The Plant Status Monitoring System (PSMS) performs the task of detecting threats to goals in terms of events and safety threats. The Dynamic Procedure Management System (DPMS) performs the task of plan execution to control threats (i.e., both the subtasks of Conflict Resolution and Plan Execution, Monitoring, and Modification). Finally, the Diagnosis and Data Validation System (DVS) performs the optional task of diagnosis to determine the cause of the detected threats to goals (i.e., events and safety threats). DVS performs its task in parallel with the task of DPMS. The overall architecture of the current implementation is shown in Figure 4.

Following are some of the details required to relate the terminology in Figure 4 to the general terminology used in the earlier sections:

- EOP refers to *E*mergency *O*perating *P*rocedures corresponding to the procedures for safety threats
- Entry conditions refer to the conditions identifying the events and safety threats
- Certain alarms also indicate threats to safety goals
- Abnormal Events refer to what have been called events
- Causes in the Diagnosis and Data Validation module refer to the causes detected by the optional activity of diagnosis.
- EOP dominant refers to the priority given to the procedures for removing safety threats.

The OA has been tested on a set of realistic simulated scenarios from the subdomain that has been represented in its knowledge base. Initial success in simulations has led to a more detailed evaluation and test under a U.S. Department of Energy project in conjunction with a full-scale nuclear power plant

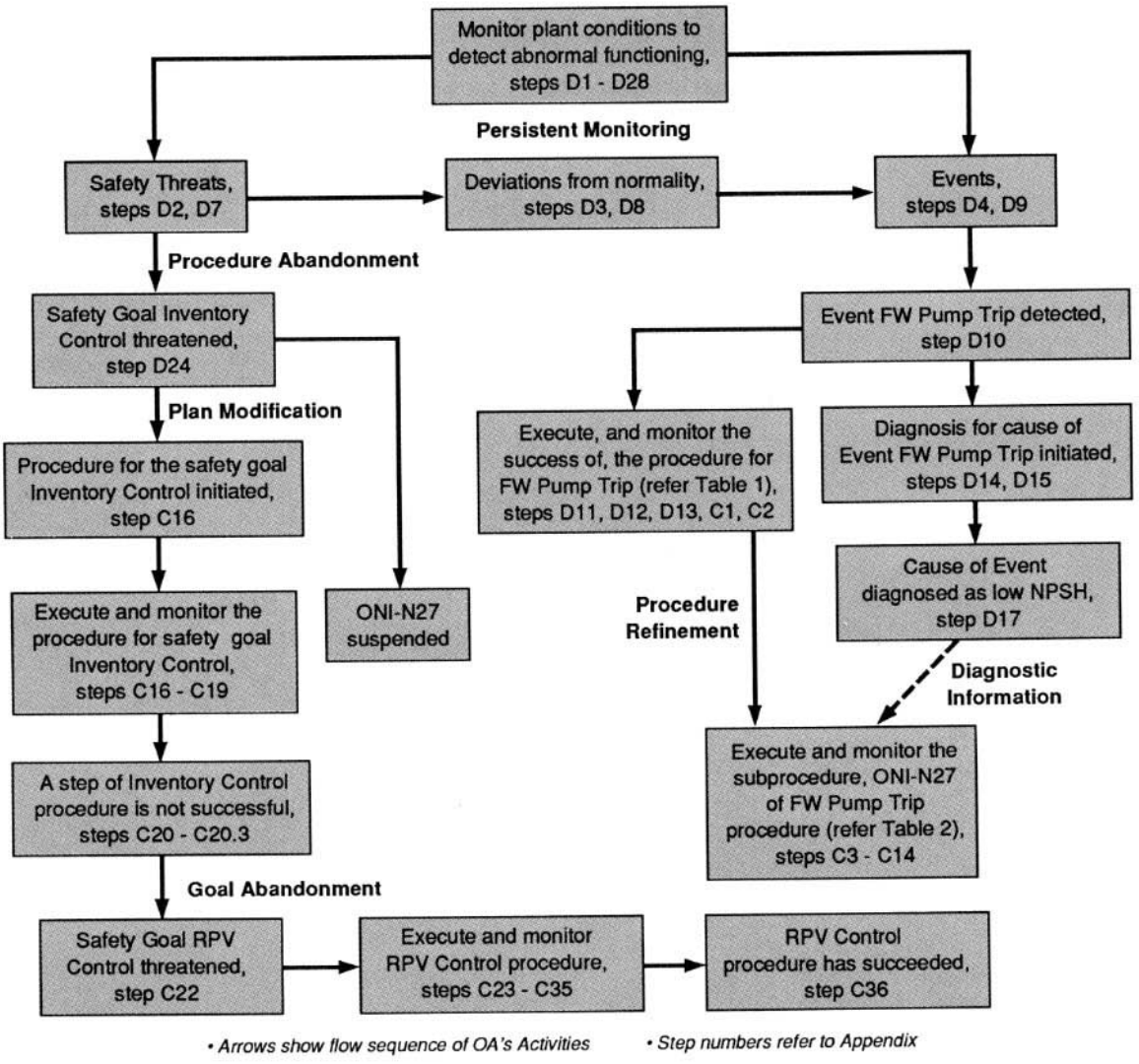
simulator which is typically used for training and certifying nuclear power plant operators.

A Disturbance Scenario and the Response of Operator Advisor

The OA is implemented on two machines. The first machine hosts PSMS and DVS (namely, monitoring and diagnosis systems), and the second machine hosts DPMS (namely, the plan execution and monitoring system). Both the machines can receive updated plant data either interactively or through the database. A disturbance scenario was simulated by providing plant data interactively to the OA. The hardcopy output (see Appendix and Figure 5) shows how the OA responded to the scenario. The scenario illustrates the activities performed by the OA. Some of the significant activities include the following:

1. Detection of safety threats (Steps D2–D9) and events (Steps D4–D9) through persistent monitoring. The event FW Pump Trip is detected in the first cycle (Step D10).
2. Communication of the detected event to DPMS (Steps D11–D13). Initiation of the procedure to control the FW Pump Trip event (Steps C1, C2).
3. Execution and monitoring of control procedure (Steps C3–C14). In this case the procedure consists of only one subprocedure, i.e., ONI-N27 (refer to Figure 5).
4. Initiation of the optional diagnosis (Steps D14–D17), concurrent with the previous steps 2 and 3, to find the causes of the event. Cause of the event (Low NPSH) detected in Step D17.
5. Interruption of the ongoing actions (C3–C14) to inform the

FIGURE 5.
Simulated Disturbance Scenario



operator about the diagnosed cause (Steps D18 and C8).

6. Detection of higher-level (safety) threat, in the second monitoring cycle (Step D24), to Inventory Control, while executing and verifying the control actions of ONI-N27. This information is communicated to DPMS (Steps D25 and C15–C16).
7. Suspension of ongoing actions (i.e., abandonment of subprocedure ONI-N27) to attend to the higher-level threat (C16).
8. Modification of the executing procedure to maintain the threatened safety goal, Inventory Control (steps C16–C19).
9. Safety goal Inventory Control abandoned, as it cannot be maintained because a step of the procedure (Step 20) failed to achieve the desired results. The procedure to maintain the next safety goal, RPV Control, initiated, to maintain safety at a deeper level.
10. Plant safety maintained by successfully completing the procedure for RPV Control (Steps C36–C42).

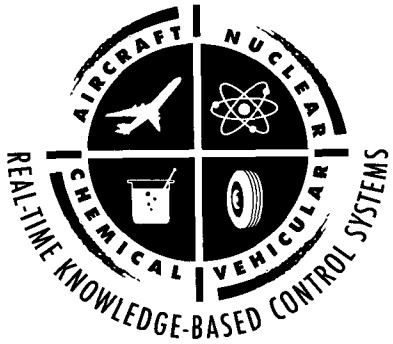
This scenario shows how the OA can help maintain plant safety at a deeper level, in unexpected situations. For example, while the event was being controlled, and another threat developed unexpectedly, the OA shifted its attention to attend to the detected threat. Further, again unexpectedly, the procedure to control the threat was not successful. The OA responded to this by giving up on the current goal to pursue the next goal to maintain safety at a deeper level.

Concluding Remarks

In this article we have analyzed the structure of the class of real-time disturbance control problems. We noted that the two major subtasks needed for an optimal resolution—namely diagnosis and plan synthesis—are intrinsically open-ended, and the completeness or correctness of solutions could never be guaranteed, least of all within a fixed time available for a solution. Generally, this means there is no control procedure that can be guaranteed to achieve arbitrary goals of interest. We showed how this fact makes it imperative to abandon and substitute goals, and has resulted, in domains such as the control of nuclear power plants, in the development of the distinction between operational and safety goals, and the notion of depth in safety goals.

We proposed that real-time considerations place a great premium on using compiled knowledge to solve rapidly, when it can, the many subtasks in procedure selection and execution. Some protection against

APPENDIX
Hardcopy of output from PSMS and DVS
D 1. PSMS New Monitoring Cycle beginning with Safety Function Monitoring
D 2. PSMS No condition threatens any safety goal, looking at general conditions of deviations from normality
D 3. PSMS No general deviations exist, proceeding to look at other events
D 4. PSMS No condition indicates any abnormal event
D 5. PSMS There is no need to do any diagnosis
D 6. PSMS New Monitoring Cycle beginning with Safety Function Monitoring
D 7. PSMS No condition threatens any safety goal, looking at general conditions of deviations from normality
D 8. PSMS No general deviations exist, proceeding to look at other events
D 9. PSMS **********
D 10. PSMS The FeedwaterPumpTrip abnormal event is established
D 11. Message is being sent to DPMS to trigger Plan for FeedwaterPumpTripAE
D 12. PSMS Following is the message from DPMS
D 13. FeedwaterPumpTripAE-Planner triggered by DPMS
D 14. PSMS Do you want to do further diagnosis for cause of FeedwaterPumpTrip (answer y or N no N n)?
D 15. Y
D 16. Diagnosis for cause fo FeedWaterPumpTripAE initiated
D 17. PSMS The cause of FeedWaterPumpTripAE is LowNPSH
D 18. Message is being sent to DPMS to trigger Plan for LowNPSH
D 19. PSMS Following is the message from DPMS
D 20. LowNPSH lower fault plan triggered by DPMS
D 21. PSMS The diagnosis has completed its evaluation.
D 22. PSMS New Monitoring Cycle beginning with Safety Function Monitoring
D 23. PSMS **********
D 24. PSMS The InventoryControlSG safety goal is established
D 25. Message is being sent to DPMS to trigger Plan for InventoryControlSG
D 26. PSMS Following is the message from DPMS
D 28. InventoryControlSG-Planner triggered by DPMS
Hardcopy of output from DPMS
C 1. > > DPMS is ready to receive messages from PSMS
C 2. > > DPMS ********** Message received from PSMS for Threat to Safety or Abnormal Event ******



the fact that this leads to a decreased likelihood of satisfactorily meeting a goal is provided by goal abandonment and substitution, since a more graceful loss of functionality can be achieved in this way. We also proposed an architecture which can additionally make use of more intensive, but more open-ended, problem solving, but which is not in the critical path of executing the more rapid, compiled procedure. If the procedures can be designed to accept any solutions by these parallel modules, in principle we can have the best of both worlds. Of course, achieving this in practice requires careful knowledge engineering of procedures. The approach proposed here is in the spirit of *any-time* algorithms [6].

Real-time control involves making trade-offs between different goals in a reasoned manner. In the approach outlined here, most of the trade-off decisions are made at design time in the way the control system sensors and actions are configured, and the way knowledge is precompiled about response procedures and the goal structure. However, there remains a dynamic aspect to control. The goals that are pursued are determined by the real-time behavior of the process system: how well and within what time the process system responds to the control actions determines if a goal will be abandoned and replaced with a weaker goal.

We discussed a specific implementation of a prototype knowledge-based system for advising operators of a nuclear power plant. OA, the implemented system, uses the general approach outlined here, but also uses a number of features specific to the domain. Most importantly, the NPP domain has a strong tradition of making all aspects of control explicit: the events and threats to be concerned about, the goal hierarchy, the procedures, etc. In domains where such knowledge is not explicitly available, substantial domain analysis will be called for to implement the approach.

Where the trade-offs involved in goal substitution may not be very attractive, "metareasoning" (i.e., reasoning about problem solving resources and how to allocate them) may make sense. Our architecture occupies a niche in the space of possible architectures—a niche where any available computational resource is thrown into the constituent problem-solving tasks (diagnosis and plan synthesis) rather than in metareasoning. Further exploration of this space is clearly called for. Similarly, additional research is needed in how to gracefully abort procedures, and how to integrate parallel tasks into a smoothly functioning control system.

```
C 3. > > DPMS FeedwaterPumpTripAE-Planner is initiated and is in
action now
C 3.1 > > ** Generating operation guidance **
C 3.2 > > Prerequisites of FeedwaterPumpTripAE-Plan being
checked
C 3.4 > > Prerequisites of FeedwaterPumpTripAE-Plan are available
C 3.5 > > Achieve goal of FeedwaterPumpTripAE-Plan being
checked now
C 3.6 > > FeedwaterPumpTripAE-Plan is in action after the
prerequisite and achieve goal checks
C 3.7 > > Prerequisites of ONI-N27 being checked
C 3.8 > > Prerequisites of ONI-N27 are available
C 3.9 > > Achieve goal of ONI-N27 being checked now
C 3.10 > > ONI-N27 is in action after the prerequisite and achieve
goal checks
C 3.11 > > Prerequisites of ONI-N27-3.0 being checked
C 3.12 > > Prerequisites of ONI-N27-3.0 are available
C 3.13 > > Achieve goal of ONI-N27-3.0 being checked now
C 3.14 > > ONI-N27-3.0 is in action after the prerequisite and
achieve goal checks
C 4. > > RECOMMENDED OPERATION IS RUNBACK
RecircFlowContValvePosA1
C 4.1 Did you Succeed in the operation? →
C 4.2 Y
C 4.3 Is the value of reactor recirculation FCV-A position (B33-
K695A) < 22 percent open ? →
C 4.4 Y
C 5. RECOMMENDED OPERATION IS RUNBACK
RecircFlowContValvePosB1
C 5.1 Did you Succeed in the operation? →
C 5.2 Y
C 5.3 Is the value of Reactor recirculation FCV-B Position (B33-
K695B) < 22 percent open ? →
C 6. > > DPMS ********** Message received directly from DVS for
the cause of Abnormal Event FeedWaterPumpTrip
C 7. > > DPMS LowNPSH is initiated and is in action now
C 8. *** TAKE CARE OF LOW NPSH
C 5.4 Y
C 9. > > ONI-N27-3.1.1 is in action after the prerequisite and
achieve goal checks
C 10. *** Reactor power should be maintained BELOW a maximum
of 100 percent by adjusting the recirculation flow
C 11. RECOMMENDED OPERATION IS ADJUST recirculation flow
```

Acknowledgments

The authors wish to thank Don W. Miller and Brian Hajek, of the Nuclear Engineering Program at The Ohio State University, for their contributions to this work; and N.S. Sridharan, of FMC, for reading the initial draft. R. Bhatnagar also wishes to thank Laurence R. Young, of Man-Vehicle Laboratory/Center for Space Research at Massachusetts Institute of Technology for his support. We thank the anonymous referees and Marcel Schoppers for their helpful comments on the first draft of the article.

References

1. Agre, P. and Chapman, D. Pengi: An implementation of a theory of activity. In *Proceedings of Sixth National Conference on AI* (1987), pp. 168–172.
2. Bhatnagar, R. A knowledge-based operator advisor system for integration of fault detection, control, and diagnosis to enhance the safe and reliable operation of nuclear power plants. Ph.D. dissertation, Ohio State University, 1989.
3. Bhatnagar, R., Miller, D.W., Hajek, B.K., and Chandrasekaran, B. DPRL: A language for representation of operation and safety maintenance procedures of nuclear power plants. Presented at the *Third International Conference on Industrial and Engineering Applications of AI and Expert Systems* (July, Charleston S.C.).
4. Bylander T., and Mittal, S. CSRL: A language for classificatory problem solving and uncertainty handling. *AI Mag. 7*, 2 (1986), 66–77.
5. Chandrasekaran, B. Generic tasks in knowledge-based reasoning: High level building blocks for expert system design. *IEEE Expert* (Fall 1986), 23–30.
6. Dean T., and Boddy, M. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on AI* (1988).
7. Hashemi, S. Expert systems application to nuclear power plant malfunction diagnosis and sensor validation. Ph.D. dissertation, Ohio State University, 1988.
8. Hayes-Roth, Barbara. Architectural foundations for real-time performance in intelligent agents. *J. Real-Time Syst. 2* (1990) 99–125.
9. Johnson, K., Sticklen, J., and Smith, J.W., Jr. IDABLE-application of an intelligent database for medical systems. In *Proceedings of the Mini-Symposium in Medical Expert Systems* (1988). Tech. Rep. Laboratory for AI Research, Ohio State University, Columbus, Oh.
10. Kim, I., and Modarres, M. Application of goal tree-success tree models as the knowledge base of operator advisory systems. *Nuclear Eng. Design 104* (1987), 67–81.
11. Rosenschein, S.J., and Kaelbling, L.P. The synthesis of digital ma-

C 11.1 Did you Succeed in the operation? →
C 11.2 Y
C 11.3 Is the value of TOTAL REACTOR FW FLOW (C34-N002A and 2B) = TotalReactorSteamFlowDB ? →
C 11.3 Y
C 12. > > ONI-N27-3.1.1 has succeeded
C 13. > > ONI-N27-3.0 has succeeded
C 14. > > Prerequisites of ONI-N27-4.0 being checked
C 14.1 > > Prerequisites of ONI-N27-4.0 are available
C 14.2 > > Achieve goal of ONI-N27-4.0 being checked now
C 15. > > DPMS ********** Message received from PSMS for Threat to Safety or Abnormal Event ******
C 16. > > DPMS InventoryControlSG-Planner is initiated and is in action now
C 16.1 > > ** Generating operation guidance **
C 16.2 > > Prerequisites of InventoryControlSG-Plan being checked
C 16.3 > > Prerequisites of InventoryControlSG-Plan are available
C 16.4 > > Achieve goal of InventoryControlSG-Plan being checked now
C 16.5 > > InventoryControlSG-Plan is in action after the prerequisite and achieve goal checks
C 17. *** The reactor water level has decreased BELOW the scram setpoint—entering PEI-B13 RPVControl
C 18. > > PEI-B13 is in action after the prerequisite and achieve goal checks
C 19. > > PEI-B13-3.0.1 is in action after the prerequisite and achieve goal checks
C 20. RECOMMENDED OPERATION IS Check if the reactor scram has initiated
C 20.1 Did you Succeed in the operation? →
C 20.2 N
C 20.3 > > DPMS InventoryControl/Procedure has not succeeded
C 21. > > DPMS InventoryControlSG-Planner has not succeeded, recovery action started
C 22. > > DPMS It is expected that RPVControlSG-Planner will be threatened
C 23. > > DPMS RPVControlSG-Planner is initiated and is in action now
C 24. > > ** Generating operation guidance **
C 24.1 > > Prerequisites of RPVControlSG-Plan being checked
C 25.1 > > Prerequisites of RPVControlSG-Plan are available
C 25.2 > > Achieve goal of RPVControlSG-Plan being checked now
C 25.3 > > RPVControlSG-Plan is in action after the prerequisite

chines with provable epistemic properties. Tech. Note 412, Artificial Intelligence Center, SRI International, Menlo Park, Calif. (1986).

12. Schoppers, MJ. Universal plans for reactive robots in unpredictable environments. In *Proceedings of the 10th International Joint Conference on AI* (Aug. 1987), pp. 1039–1046.

13. Sharma, D.D. A knowledge based framework for procedure synthesis and its application to the emergency response in a nuclear plant. Ph.D. dissertation, Ohio State University, 1986.

CR Categories and Subject Descriptors: C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems—*Real-time Systems*; I.2.1. [**Artificial Intelligence**]: Applications and Expert Systems; J.7. [**Computer Applications**]: Computers in Other Systems—*Process Control.*

General Terms: Human Factors

Additional Key Words and Phrases: Intelligent control, nuclear power plant

```
and achieve goal checks
C 26. > > PEI-B13-3.0.1 is in action after the prerequisite and
achieve goal checks
C 27. RECOMMENDED OPERATION IS Check if the reactor scram
has initiated
C 27.1 Did you Succeed in the operation? →
C 27.2 Y
C 28. > > Prerequisites of PEI-B13-3.1.1-MONITORA being checked
C 28.1 Are all control rod positions < = 2 ? →
C 28.2 Y
C 29. RECOMMENDED OPERATION IS SET ReactorModeSwitch1 to
Shutdown
C 29.1 Did you Succeed in the operation? →
C 29.2 Y
C 30. > > RPVPowerControl is not required as its achieve goal is
already true
C 31. > > RPVLevelControl is in action after the prerequisite and
achieve goal checks
C 32. > > PEI-B13-3.3.1 is in action after the prerequisite and
achieve goal checks
C 33. RECOMMENDED OPERATION IS MANUALLY-ISOLATE
MSIVOutboardVlvDB
C 33.1 Did you Succeed in the operation? →
C 33.2 Y
C 33.3 Are all of the main stream line outboard isolation valves
Isolated ? →
C 33.3 Y
C 34. > > PEI-B13-3.3.2-MONITOR is in action after the
prerequisite and achieve goal checks
C 35. RECOMMENDED OPERATION IS MAINTAIN RCIC Turbine
speed below the RCIC Turbine Speed Limit but above 2000 RPM
C 35.1 Did you Succeed in the operation? →
C 35.2 Y
C 36. > > RPVLevelControl has succeeded
C 37. > > RPVPressureControl is not required as its achieve goal is
already true
C 38. > > DPMS RPVControlSG-Plan has succeeded
C 39. > > DPMS PRVControlSG-Planner has succeeded
C 40. Is the value of RPV wide range level instruments > 178
inches ? →
C 40.1 Y
C 41. > > DPMS InventoryControlSG-Planner has succeeded
C 42. > > DPMS No more active plans to be pursued in NPPGoals
```

About the authors

B. CHANDRASEKARAN is professor of computer and information science and director of the Laboratory for AI Research at Ohio State University. His current research interests are cognitive architectures, knowledge-based reasoning in diagnosis and design, device understanding, and visual reasoning. **Author's Present Address:** Laboratory for AI Research, Department of Computer and Information Science, Ohio State University, Columbus, OH 43210, chandra@cis.ohio-state.edu.

RAJIV BHATNAGAR, formerly with MIT's Man-Vehicle Laboratory/Center for Space Research, is currently at GTE Laboratories. His research interests include applications of knowledge systems to a wide range of problems. **Author's Present Address:** GTE Laboratories, Inc., 40 Sylvan Rd., Waltham, MA 02254, rajiv@ctc.contel.com.

D.D. SHARMA is a senior research engineer and a project leader at Advanced Decision System. His research interests include real-time AI systems, knowledge-based reasoning in dynamic planning and plan execution, and application of knowledge-based approaches to engineering problems in general. **Author's Present Address:** Advanced Decisions System, 1500 Plymouth St., Mountain View, CA 94043, dsharma @ads.com.

This work was partially supported by the National Science Foundation Grant No. ECS 8616254 and the Department of Energy Contract No. DE-AC02-86NE37965. B. Chandrasekaran's contributions were supported by the AFOSR grants 87-0090 and 89-0250 and DARPA contract RADC F30602-85-0010.