A Plan Induction System for Monitoring and Interpreting Operator Interventions in Process Control Environments

Running Title: A Plan Induction System

William E. Spangler
College of Business and Economics
West Virginia University

James M. Peters
Heinz School of Public Policy and Management
Carnegie-Mellon University

1 Corresponding Author:

William E. Spangler
College of Business and Economics
P.O. Box 6025
West Virginia University
Morgantown, WV 26506-6025
Phone: 304-293-7933
Fax: 304-293-8905
E-mail: wspangle@wvu.edu
A Plan Induction System for Monitoring and Interpreting Operator Interventions in Process Control Environments

ABSTRACT

This paper describes the architecture and behavior of a prototype intelligent decision support system for monitoring operations in complex process control environments. Development of the underlying model required an examination of the various influences on process outcomes, including not only the causal nature of physical processes themselves, but also the role of human interventions and the associated impact of operating procedures on human behavior. The empirical study of nuclear power plant operations used in this research indicates that procedures are an important, but not necessarily deterministic, influence on the intervening behavior of an operator. Operators will deviate from procedures when the requirements of a situation render a procedure inadequate or counterproductive.

Goal- and plan-based knowledge structures were derived from physical processes, operating procedures, and human operators. These structures were incorporated into the model's knowledge base, which serves as the basis for interpretation and prediction of operator interventions in a series of emergency scenarios in simulated real-time. The eventual goal of this research is to enhance management oversight and control of complex, dynamic task environments by providing both management and operators with advice that is informed by an understanding of the constituent influences on process outcomes.
1. INTRODUCTION

This article describes the computational model underlying a prototype process control decision support system, and explores the issues of decision support in human-mediated process control environments. A complex process control system, such as the nuclear power plant environment under study here, distributes responsibility for decision making and action across the individual constituents that comprise the system (see Figure 1). The behavior of the system is influenced by each of these distributed constituents, which include:

1. the physical components of the system, which react to inputs and produce outputs based on the causal nature of the processes. The fluid level in a tank rises, for example, when there is a net in-flow of fluid into the tank.

2. automated interventions, which react when a specified set of system parameters crosses a pre-determined threshold. A safety valve closes, for example, when the pressure and/or fluid level inside of a tank falls below some value. In engineered systems, control processes generally reflect the presumptions and rationale of system designers.

3. human operators, who respond to observed events in the context of a) their own goal structure, which might be facilitated or threatened by system events, and b) organizational programs or procedures, which direct certain actions based on system conditions and events.

Insert Figure 1 about here
Decision support in this type of environment must consider a) who in the organization is a decision maker and therefore in need of decision support, and b) the types of information required for effective decision making. In a process control environment, decision making typically is distributed between the organization's management and the operators who act on behalf of management's goals and directives -- as indicated in policy and procedure manuals.

The purpose of the plan induction model is to provide decision support to both parties. By constructing a situation assessment and set of possible response plans in real-time, the DSS can help to guide operators through problems for which static, written procedures may be inadequate. Management, in turn, is provided a type of control by proxy, whereby organizational goals and intentions can be communicated to operators through the computational model. The model's situation assessment and associated planning framework also provides a context within which the DSS can anticipate or predict operator actions and then explain subsequently-observed actions within the same context. In real-time, on-site management could use the information about anticipated actions to communicate organizational preferences and preempt undesired actions. A post-mortem explanation of operator actions, in turn, could help management a) to adjust control mechanisms as needed -- i.e., to facilitate organizational learning -- and b) to educate operators regarding appropriate actions under observed conditions.

With respect to the second decision support consideration listed above, the distributed nature of the domain requires a monitoring DSS to include certain types of knowledge and functionality. To perform a realistic situation assessment of a complex system, the model underlying the DSS should include not only the individual components
of the system, but also the goal-based interactions between components, which generate
the behavior of the process control system. This entails the construction of a physical as
well as a cognitive model, the latter being important because of the critical role of the
human operator in the overall generation of system behavior. The operator influences
system behavior through various manual interventions, at times interrupting or modifying
a causal chain of events occurring as a result of physical processes. For example, an
increase in pressure in a vessel above a certain threshold might be expected to cause a
cascading series of system failures -- unless the operator takes a compensatory action,
such as opening a relief valve. A cognitive model of the operator, including his or her
goal and planning structures, allows a DSS to anticipate and understand these types of
interventions, and to provide an interface that facilitates effective communication with
the operator.

Notably, operator interventions are influenced, but not necessarily determined, by
the directives of applicable operating procedures. The relationship is non-deterministic
because procedures at times can be incorrect, and experienced operators often are capable
of recognizing those situations and acting in an ad hoc manner to achieve or maintain
important system goals. This study focuses on the interaction between the action plans of
human operators and operating procedures, and on the system behavior that results from
the interaction. The computational model is intended to observe, interpret and predict
events and activities occurring in the process control system -- from both physical and
human sources.

With respect to human interventions, the model interprets two observed aspects of
program-constrained behavior: 1) actions taken during program execution that might be
construed as deviating from the constraints of the program, and 2) arguments or rationalizations constructed by the decision maker to justify and defend such actions. The empirical data indicate that, during program execution, operators tend to deviate from programs when organizational or management goals will not be achieved through rote execution of the current program. Because a program generally cannot anticipate every situation that might arise during its execution, it will proceed based on default, and potentially incorrect, assumptions concerning the state of the world at execution time. Acting on incorrect expectations creates a paradoxical incongruity between the higher-level goals that the program intends to achieve and the goals that will actually be achieved as a result of its execution in real-time. This in turn can motivate an enlightened program executor to deviate from the program in order to avoid the expected adverse consequences that might be incurred by the organization if the program were to be executed as-written.

When a deviant action is contemplated or implemented, the organizational context encompassing the operator requires that s/he justify his/her decision to deviate from the program. As noted, programs are instruments of behavioral control, and to this end they typically impose penalties upon operators who deviate from program constraints. These penalties in turn provide sufficient motivation for an operator to construct an argument in support of any program-constrained behavior that is claimed to be in violation of a prescribed program. The argument constructed by an operator in response to an expected or realized criticism of his actions thus is an integral component of his program-constrained planning process, and an important, explicit indication of his reasoning.
2. PROGRAMS AND PROGRAMMED DECISION MAKING

The approach to organizational programs and programmed decision making taken in this research is based in large part on the organizational theoretic work of March and Simon (1958), who laid out the basic structure of organizational procedures and the circumstances underlying their execution. Programs are important to the extent that they are capable, through their execution, of achieving or maintaining specified organizational goals and subgoals. Therefore goals, along with pre-defined environmental triggering conditions, serve as the basis for organizational selection and execution of appropriate programs.

When faced with a problem, people tend to search for solutions that satisfy the minimal requirements established for a task; i.e., they *satisfice* (Simon, 1947). The tendency toward satisficing in turn predisposes people toward program-directed behavior. By retaining previous solutions in the form of organizational procedures, the processing involved in the decision-making process can be significantly reduced. Simon has outlined a three-stage process of decision-making, which encompasses *intelligence*, *design* and *choice* (Simon, 1960). Simon uses the military connotation of the term ‘intelligence’ to describe the initial phase, an environmental scanning of events or circumstances that might require action. Developing and evaluating possible plans of action comprises the design phase, while selecting a particular plan from candidates developed in the design phase constitutes the choice phase. The availability of pre-formulated programs eliminates much, but not all, of the design activity. A program that is somewhat too general, or is not relevant in all respects, can still be selected -- and then modified as required to satisfy the task requirements.
This hybrid nature of the task begins to hint at the complexity of the problem-solving activities comprising it. While most behavior is programmed to some extent, most programmed behavior is program-influenced rather than program-determined. The degree to which a decision is programmed affects the amount of discretion afforded to the decision-maker. Highly programmed tasks allow little or no discretion, while non-programmed tasks allow unconstrained discretion. Much of the discretion in turn depends on the structure of the program itself. As Simon notes, if a program specifies a goal to be attained, but does not indicate the means for its attainment, then the person executing the program has discretion in designing and choosing the means.

While program constraints and influences on behavior vary with the specificity of the program, the influence also varies with the actual or perceived relevance of the program in solving the problem at hand. The relevance of the program at the time of its execution is dependent on the ability of the program writer to anticipate a priori all of the situational factors to be confronted by the program. If unanticipated events occur during execution, the program executor may have to modify portions of the program to conform to the changed situation, thus exercising what might be called implicit or non-program-directed discretion.

2.1 Procedure-based explanation

If behavior can be guided and constrained by organizational programs and procedures, then those same procedures can be used as a partial basis for explanation of that behavior. Allison (Allison, 1971) maintains that many observed behaviors can be explained in terms of organizational programs, specifically in terms of the output of organizational programs. An organization has defined goals, and programs are
constructed and implemented in order to coordinate activities so that the goals are achieved. Knowledge of an organization’s goals allows inference of the types of programs that might be invoked to achieve those goals. Conversely, inference of organizational programs from observed actions provides some indication of the organization’s goals.

March and Simon (1958) in turn offer insights into the explanatory capabilities of programs that are congruent with the aims of this research: 1) organizational programs can be used to explain and predict behavior, and 2) the quality of the explanation and prediction is correlated with the how heavily programmed the behavior is. If a task is highly repetitive and routine, the behavior associated with that task will be heavily programmed, and thus easily explained in terms of the active program. As noted, most tasks have aspects of routine and novelty, and thus are partially-programmed. Partially-programmed tasks require additional explanatory mechanisms, derived from a model of the human procedure executor, that can encompass the types of ad-hoc methods occurring outside of procedural constraints.

2.2 Empirical study and modeling of Nuclear Power Plant operations

The management of a nuclear power plant requires human operators to monitor and respond to the behavior of hundreds of plant parameters -- levels, pressures, valve positions, flow rates, and so on. Because of this, much of the empirical study of NPP operations focuses on the general cognitive abilities and limitations of operators, with an eye toward limiting the contribution of human error to critical incidents and emergencies through human factors engineering and improved ergonomics (Shankhar, 1991; Staveland, 1991; Cacciabue et al, 1992; Corker and Smith, 1993). Other studies have
focused more closely on the detailed cognitive processes of operators, with the intention of reducing human error through cognitive modeling and improved computer-based decision support (Woods et al, 1990; Roth et al, 1992; Roth et al, 1993; Roth et al, 1994). These latter studies provide an empirical and conceptual basis for this research, since they investigate the same issues of situation assessment, planning, and procedure navigation necessary for the development of an intelligent DSS in this domain. The sometimes paradoxical influence of procedures on operator behavior, caused primarily by inherent flaws in the procedures themselves, is of particular interest and is reviewed below.

2.2.1 The impact of procedures on operator behavior

In managing the complexity of a nuclear power plant, operators commonly are required to follow pre-defined written procedures designed either for normal operations, or for emergency or malfunction response and recovery. Ironically, the reason for engaging in pre-planning and procedure definition is that by conventional wisdom, human operators cannot be trusted to take appropriate actions when “under fire” as some major accident scenario unfolds (witness the Three Mile Island nuclear power plant incident). Furthermore, without guidance operators might attend only to the particulars of the emergency itself, and in the course of trying to resolve the emergency might actually endanger other, perhaps more important, plant goals.

Although the written procedures are fully specified in advance, the need for real-time adaptation of procedures to meet unexpected situations is seen as one of the main reason for keeping humans in the loop. They are expected to function in those particularly difficult situations - perhaps involving confounding evidence arising from multiple faults - where existing procedures are at best unhelpful and at worst dangerous.
Therein lies the paradox: humans cannot be taken out of the loop because execution of procedures must be carried out on the basis of experience and judgment; yet humans cannot be trusted to make the right decisions in real time without preplanned guidance.

For a complex facility such as the nuclear power plant, emergency procedures are necessarily written to deal with an expected state of the world when, in fact, the choice of the appropriate action to be taken requires knowledge of the real state of the world. As a result, the procedure might constrain the operator and prevent timely action from being taken to resolve the problem. It is not surprising that these inherent limitations in procedures would force additional cognitive and response planning requirements upon operators. Roth et al (1993) identify a number of requirements, including the need to construct an adequate situation assessment and an accurate mental model of the physical plant. Most important to this research, however, they argue that operators must be able to act when procedural guidance is lacking or ambiguous. Operators must have “the ability to balance multiple goals when explicit procedural guidance is not available,” and “the ability to take discretionary action to mitigate or prevent more severe consequences.” (Roth et al, 1993, p. 497)

A model of operating procedures in several domains, including NPP operations, is included within Pople’s EAGOL system (Pople, 1994; Pople et al, 1994). The EAGOL model provides a partial, procedure-goal framework for the explanatory model constructed in this research. EAGOL generates a linearized trace of a procedure as well as the goal hierarchy that is pursued through execution of the procedure. The hierarchy of existing or newly-established plant goals attached to the procedure is retained in a
knowledge structure known as the *synops net*, which in turn is accessible to both the EAGOL program and the operators as the scenario unfolds.

### 3. COMPUTER MODEL OF PROGRAM-CONSTRAINED BEHAVIOR

In this study, a computational model of complex system behavior was constructed from analysis of empirical data gathered in a human-mediated, process control environment (i.e., Nuclear Power Plant (NPP) operations). Although highly proceduralized, the task nevertheless demonstrates the limitations of procedures in determining the behavior of human intermediaries, and therefore, in determining the overall behavior of a complex system. Observations of behavior in simulated scenarios indicate that, despite the ritualized nature of procedure execution and the possibility of sanctions for deviations, operators will depart from the directives of a prescribed procedure when they believe that rote execution of the procedure is not appropriate.

The computer model was developed from an analysis of the relevant procedures and the reasoning of operators, as evidenced by their utterances and actions, during simulated plant emergencies. The goal of the model is to predict, and then explain, the behavior of the complex system based on a description of the organizational processes. The model receives as input a series of process-induced events, and produces as output an explanation of the events and/or a prediction of future actions resulting from the event.

The model's focus on an explanation of operator action is intended to demonstrate that it can understand the contextual features of the process control environment, to the degree that it can predict behavior. From a decision support perspective, while the explanation generated by the current model is directly useful for ex post evaluation by
management, the underlying theory embodied in the model is perhaps more important to
the types of sophisticated decision support that are the eventual goals of this research.
That is, an understanding of complex process behavior, which includes procedure-driven
human intervention, is essential for providing real-time advice both to management and
to human operators. In short, the model described in this section serves as a theoretical
prerequisite for decision support in process control. We return to this issue in the final
section of this article.

3.1 Individual knowledge structures

Explicit in the model are two sets of planning structures made necessary by the
imposition of organizational programs. The first set is comprised of 1) the plans and
actions of the procedure itself, coupled with 2) the ad hoc reactive plans of the procedure
executor (i.e., the operator), which can lead to actions taken outside of the directives of
the procedure. The second set of planning structures is comprised of a decomposition of
the executor's reactive plan into 1) the restoration plan seeking to achieve, maintain or
reestablish a management goal, and 2) the implementation plan, which seeks to carry out
the restoration plan in the context of program constraints. The implementation plan
focuses on the nature of the violation, if any, and the rationale underlying the action(s)
taken in the restoration plan. The goal of the implementation plan is to avoid any adverse
consequences that might arise as a result of actions taken to maintain the important goals
of the system.

The model is comprised of constituents of both procedural and human planning,
which include:
• the *p*-goal hierarchy -- the goals and actions (i.e., steps or directives) represented by a procedure

• the *procedure pathway(s)* -- an extrapolated series of steps through the procedure leading to a specified goal in the hierarchy. Depending on the state of the system, there might be several alternative pathways through a procedure at a particular point in time.

• the *m*-goal hierarchy -- the operational (or management) goals held by individual human operators\(^1\)

• the *r*-goal hierarchy -- the ad hoc reactive goals and actions generated by operators in response to actual or anticipated m-goal violations

The specific goals and actions implemented at any given time are influenced by certain *decision variables*, which collectively characterize the state of the world in the context of the structures described above. The variables are:

1. **the presence or absence of an available procedure pathway**: Pathways are important because they identify alternative programmatic means of attaining procedure goals, and in so doing allow a procedure-constrained decision maker to think strategically about how or whether the procedure is capable of attaining organizational goals.

2. **the conformance of conditions for step execution with current system state**: Each step or directive in a pathway has certain explicitly-defined conditions for its execution -- conditions which the original procedure designers felt were required *in the default*

\(^1\) An important assumption is that, in a process control environment, the operators' goals are to achieve management goals.
circumstances under which the procedure would be executed. If those conditions are not in compliance with the current state of the world, execution of the step -- and traversal of the path -- will result in a violation of procedure constraints.

3. the conformance of step assumptions with current system state: The ‘default circumstances’ described above comprise the implicit assumptions or expectations that, like explicit conditions, constrain the execution of a particular procedure step. Because a pre-formulated procedure cannot anticipate every possible situation that might arise during procedure execution, a procedure might direct or prohibit execution of a step based on possibly incorrect assumptions concerning the state of the world. If so, an actor who decides to violate the procedure might construct a rationale in part by citing the validity, or lack thereof, of inferred assumptions.

In this model, specific procedural assumptions have been abstracted into what is termed goal-based assumptions -- i.e., the perceived need to achieve or not achieve certain procedure goals under specified circumstances. Because of its associated p-goal hierarchy, each procedure step contains inherent assumptions regarding which goals must be achieved as a result of the step's execution. If a step is required, implicit assumptions suggest that achievement of the superordinate p-goals attached to the step is required. If a step is proscribed because current conditions do not allow its execution, the assumptions suggest that achievement of the step's p-goals is not required. An actor in turn might justify execution or avoidance of a procedure step, particularly in the context of a condition violation, by attempting to support or refute the goal-based assumption linked to a procedure step.

4. the source of a management goal violation: This variable presumes that procedure-
deviating human behavior is motivated by the failure of procedures to achieve required goals, thus resulting in goal violations. In the context of this research, management goals can become violated in one of two ways: 1) an external event or series of events in the system causes the goal to fall out of compliance, or 2) execution of a procedure directive violates the goal. If the source is external, the actor might be motivated to execute a specific procedure step identified as a potential means of restoring the goal. However, if the source is itself a procedure directive, the actor might instead choose to avoid the execution of the step, particularly if countermeasures that might mitigate or negate the effect of the step are unavailable.

3.2 Network of knowledge structures

Again, behavior in a program-constrained task environment is defined both in terms of 1) plans constructed in order to achieve, maintain or restore management goals, and 2) plans constructed in order to implement and rationalize m-goal-restoring plans. Consequently, an explanatory model must establish a basis for linking these two aspects of the behavior together within a single network of knowledge structures. Figure 2 illustrates how knowledge of each planning space is represented within the model. Collectively, the structures of the planning dichotomy comprise the Global Expectation Network (GEN), and include each of the knowledge structures described above, including:

1. the violated or threatened m-goal and associated m-goal hierarchy. In the model, the m-goal is linked directly to the central global expectation object shown in Figure 2. The remaining goals in the m-goal hierarchy are not shown in Figure 2, but are similarly represented as objects and linked to the violated m-goal object.
2. the \textit{r-goal} hierarchy, represented as a chain of linked objects and connected to the global expectation object.

3. the current observed action, retained as an intervention object and linked to the associated \textit{r-goal} object. This action corresponds to a possible -- and potentially program-violating -- action contained in a restoration goal hierarchy

4. one or more \textit{path expectation} objects, containing information about which paths, if any, contain the action

5. the values of the decision variables, collected as they are observed and distributed within appropriate structures. The violation source variable (\textit{external} or \textit{directive}) is global in nature and thus are stored directly in the global expectation structure. The \textit{available pathway} variable is applicable to the action itself, and thus is stored in the intervention object. The condition- and assumption-satisfaction variables apply to specific paths and thus are stored in a \textit{path expectation} object.

Insert Figure 2 about here

\section*{3.3 Process model}

The knowledge structures have been incorporated within a computer model written in LISP and implemented on a Sun workstation running the Unix operating system. As shown in Figure 3, the computer model essentially is comprised of two sub-models: 1) a \textit{behavioral process} model, which generates a set of possible operator behaviors, in the form of plans and actions that an operator might execute within some
type of programmatic constraint, and 2) a plan induction model, which uses the planning space constructed by the behavioral model as the basis for explanation and prediction of operator behaviors that might arise.

The behavioral process sub-model is invoked by a system event, which is a change in the status or behavior of some environmental parameter -- e.g., a loss of coolant flow or an endless loop in a procedure. When a system event is observed, the behavioral process model generates restoration plans and implementation strategies that might be invoked by the operator in reaction to the event. Two general sets of plans, each forming an indication of the possible intentions of an operator, are generated by the behavioral process model.

The first is the set of alternative program pathways that might be navigated by an operator in pursuit of either program or management goals. Because a pathway is comprised of both procedural actions and the conditions required for their execution, the pathway defines an operator's program-based options as well as the specific nature of any violation -- i.e., violation of a condition, independent action, and so on -- required in order to achieve a goal.

The second set of plans, which correspond to possible maneuvers outside of the program, are derived from 1) the restoration plans formulated by the operator in pursuit of a violated or threatened management goal, and 2) the implementation plans or strategies invoked by the operator in order to execute the restoration plan.
Implementation plans consider the values of the various decision variables, and are carried out within the constraints imposed by the constraining program. Thus, depending on the status and relative impact of the conditions and assumptions attached to specific program steps, an operator might choose one of several alternative strategies in achieving or restoring a goal - i.e., program step execution, condition violation, condition satisfaction, independent action, and so on.

In the second sub-model, the empirically-derived set of plans and methods from each source becomes input to a theory-based plan recognition/induction process. This sub-model in turn is invoked by a response event, which is some type of action invoked by the operator. When a response event is observed, the model attempts to match the event to actions contained in each of two planning repositories: 1) the model’s set of generated restoration and implementation plans, and 2) its knowledge of program directives and pathways. The output resulting from this second event type defines the essential purpose of the model: i.e., it explains the operator’s behavior in the context of the program/restoration planning space.

4. TESTING THE MODEL AGAINST EMPIRICAL DATA

In the empirical studies described in (Roth et al, 1992; Roth et al, 1993), several teams of operators were presented with scenarios related to two specific types of plant emergency, each of which was created using a high-fidelity plant simulator mirroring an actual plant control room. One of the scenarios involved a type of incident termed a loss of coolant accident, or LOCA. A generic LOCA refers to any situation in which coolant escapes from a plant subsystem, primarily through a leak or break in the piping. A LOCA is a serious development within the plant, and if unresolved could lead to an
inability to maintain proper temperature and pressure, and possibly to a core meltdown. Because of this, and because LOCAs often are difficult to recognize, locate and correct, NPP operators are provided with a series of written LOCA procedures intended to guide them through the resolution process.

The monitoring, interpretation and prediction capabilities of the model were tested against a number of simulated emergency scenarios presented to operators, including the LOCA scenario described above. This section describes both the operators' and the model's behavior against a particular incident in the LOCA scenario: i.e., a component failure that prompted operators to violate a procedure condition in order to restore an important m-goal.

4.1 Operator behavior in the LOCA scenario

NPP procedures typically are comprised of a series of steps that include a default action to be performed if a condition is true, and (perhaps) an alternative action if the condition is not true (termed a ‘response not obtained’, or ‘RNO’ action). If an observed action does not conform to the system state indicated by the condition, then the action will have violated the condition, and therefore the procedure as well. Figure 4 is an excerpt of a transcript of operators who are engaged in condition-violating behavior intended to restore a previously-observed m-goal violation. In this case, the LOCA has caused a compensatory increase in the rate of charging (i.e., coolant) flow, to the point where charging flow is now at its maximum level. This in turn is a violation of an established m-goal, which is to maintain charging flow at some level below its maximum so that operators will have some flexibility in increasing the flow when that becomes necessary. Consequently, the operators are seeking some means of addressing the
management goal violation. When an operator indicates in line 6 that he will ‘Realign through the BIT’ (*Boron Injection Tank*), he is referring to an auxiliary source of coolant that presumably will allow a reduction in the rate of charging flow to a point below its maximum. In the context of the model, this would correspond to a restoration method (*r*-method) invoked in order to restore the violated charging flow management goal.

Notably, the procedure technically does not permit the operators to take this action. The procedural action causing realignment of flow through the BIT is associated with the RNO directive of the applicable procedure. However, plant conditions at the time required execution of the *default* action rather than the RNO action. The default action did not allow use of the BIT, and therefore did not, in the operator’s view, address the current goal violation. Therefore, he chose, in what he termed a ‘judgement call’, to violate the procedure by executing the RNO action, despite its lack of conformance with current conditions.

### 4.2 Model behavior in the LOCA scenario

The model observes the series of system events and operator actions described above, and constructs a situation assessment from which it can interpret observations and anticipate future actions. The events are presented to the model as a series of Lisp statements, each of which is read from a text file and converted into internal event objects. Figure 5 shows an excerpt of the text file that served as input to the model.
Analysis takes place in a basic processing loop in which the model reads an event, generates a situation assessment, and then displays its output. The output is generated from a series of text-based templates, each of which is linked to the internal knowledge structures comprising the GEN. When the model forms an interpretation, it retrieves the associated template, fills the slots with the required knowledge elements, and then displays the completed template to the user. This results in an English-like output that approximates the assessment of an experienced human commentator. Figures 6 through 10 show excerpts of the commentary of the model in reaction to its observations of the events in Figure 5.

The essential aspects of model behavior are evident in its response to the first and last events shown in Figure 5. The first event is termed a *param-value-change* event, which describes a system event corresponding to the change in value of a particular system parameter. In this case, the charging flow parameter has risen to its maximum level. In the output shown in Figure 6, the model recognizes the change in value, and then indicates that the new value has caused a management goal violation. In the last paragraph of Figure 6, the model makes explicit the superordinate goals in the m-goal hierarchy that depend on this now-violated management goal -- i.e., the compromise of controller flexibility.
The model's process of discovering the management goal violation is as follows. The procedural daemon responsible for monitoring changes in charging flow rate reads the change in its rate, and assesses the impact on any management goals associated with charging flow. The daemon searches through the hierarchy of management goals related to charging flow, and then evaluates, in the current situation, the system states corresponding to each of the m-goals. In this case, a state attached to one of the two m-goals retrieved for charging flow is now false, which causes the daemon to add the m-goal to its list of violated management goals.

Having inferred the management goal violation, the model constructs its set of possible operator reactive behaviors (i.e., the global expectation network), beginning with the search for restoration/recovery goals (r-goals) that might be invoked by the operator in order to restore the management goal. In Figure 7, the model lists the matching r-goal hierarchy, showing first the top-level r-goal that is linked directly to the violated management goal, then the subordinate r-goals, and finally the r-methods that serve as the basis for possible subsequent operator actions. Note that one of the methods listed is an 'ENGAGE BIT' method, which will be observed subsequently.

The last event is an intervention event, which describes some type of operator action. In this case, the event simply describes the operator's manual engagement of the BIT. This is the pivotal event in this scenario -- the one that reflects the operators'
decision to deviate from the procedure in order to invoke a specific r-method (i.e., 'engage BIT'). The model begins its analysis of the event by relating it to current procedure pathways and steps. As shown in Figure 8, the model has linked the event to the action corresponding to "step 8a~" -- the tilde (~) symbolically representing the RNO component of the step. Although the model is capable of maintaining multiple alternative pathways in its knowledge base, the current sequence of observed steps (previous observations are indicated by '*'; the current event is indicated by '**') correspond to a single path, which serves as the model's procedure-based hypothesis of operator actions and intentions.

The model then tests the condition for the step (i.e., the level of coolant in the pressurizer should be stable -- neither rising nor falling) and determines that it has not been satisfied (see Figure 9, first paragraph). It responds to this condition violation by attempting to relate the event to actions residing in the global expectation network. The model presumes that there are two possible reasons for a condition violation: 1) the actor specifically wanted to execute this step in order to respond to an existing management goal violation, or 2) the actor decided to avoid or circumnavigate the previous step, and the condition violation observed here is simply a side-effect of that decision. In this case, the first reason applies; i.e., operator specifically chose this action. The action corresponding to engagement of the BIT is explicitly resident in the global expectation structure. The model searches through its existing expectation network, retrieves the
applicable expectation by matching the observed action to the expectation-resident action, and notes the result in the second paragraph of Figure 9.

Having retrieved the appropriate expectation structures and action, the task of the model is to explain the condition violation in the context of the retrieved expectation structure, including the various constituents of the associated implementation plan (IP). From the condition violation and retrieval of an associated expectation, the model assigns a standard characterization to the current action -- CONDITION-VIOLATION -- and proceeds on to respond to the condition violation. The model's response entails a disambiguation of the decision variables and identification of the applicable IP. At this point in the analysis, the model knows the values of three of the four variables: 1) condition-satisfied? = no (the condition for step 8A~ is not satisfied), 2) available-pathway = yes (as shown in Figure 8), and 3) source-of-program-violation = external (i.e., a system event, and not a step in the procedure, was the cause of the current m-goal violation).

Determination of the assumption-satisfied? variable requires additional model processing to ascertain the achievement of goals and the implicit assumptions that accompany them. The decision to execute step 8A~ and achieve associated p-goals suggests that the goal-based assumption is true -- i.e., that the achievement of superordinate p-goals via execution of this step is required. The model tests this thesis by searching the superordinate p-goals for one matching an applicable restoration goal. A
match suggests that achievement of the p-goal is required as a means of restoring the violated m-goal. Having found a match, the model searches for any other program steps that might achieve this same goal. When the search is unsuccessful, the model is able to conclude not only that the p-goal achieved by the program step is required, but also that the program step is uniquely capable of achieving the required p-goal. This last conclusion is the essence of the step's assumption.

The model proceeds as follows. First, it searches for a match between the hierarchy of program goals attached to the current step and the hierarchy of restoration goals identified to respond to the violated management goal. If a match is found, the utility of the program goal -- in addition to the restoration goal -- in restoring the management goal is demonstrated. Next, the model attempts to show that there are no alternative means of achieving the goal, which requires establishment of the following: 1) there are no other steps that might achieve the same goal or, if there are, those steps are similarly blocked by non-compliant conditions, and 2) if the current node exists in other pathways, the condition for each of its incarnations must also be false.

Testing these conditions is a search and evaluation process. The model progressively searches through the program goal hierarchy from each step node in each pathway in an attempt to find a match with the restoration goal of interest. If a match is found, the model evaluates the condition attached to the matching node in order to verify that the condition is false. After all other alternative nodes have been analyzed in this fashion, the model searches the other pathways for any other instances of the current node. For each instance found, the model evaluates the attached condition to verify that the condition is false.
The result of the evaluation process is a conclusion that the goal-based assumption underlying the step is true; i.e., that the step is uniquely capable of achieving the superordinate p-goal, and therefore the violated m-goal as well. The condition-satisfied (false) and assumption-satisfied (true) variables then combine with the values of the other decision variables to point to the implementation plan labeled 'DP-5' in Figure 9. In Figure 10, the model summarizes its rationalization of the observed operator behavior.

5. CONCLUSIONS

The eventual goal of this study is enhanced management control of complex, dynamic task environments via a computer-based DSS proxy. Currently, management control in process control environments is governed generally by automated systems and procedure-constrained human behavior. Each of these mechanisms is problematic. Both automated systems and procedures operate based on simplifying, and occasionally incorrect, assumptions about the state of a monitored process. Although the need for human intervention is clear, interventions can be ad hoc, unguided, and occasionally incorrect as well.

As noted, the theoretical model developed in this research provides a contextual understanding of process constituents and relationships that is required for decision support. An intelligent monitoring DSS, containing knowledge of the physical plant,
management goals, procedures, and operators, could provide structured, defensible guidance to operators when procedures become inadequate. Operators currently construct restoration plans when faced with threats to management goals. The DSS would help them to construct correct plans, in conformance with the overall goals of the organization -- although not necessarily in conformance with current procedural directives.

Because procedures do not determine behavior -- even in the highly proceduralized domain of NPP operations -- the findings described here are potentially applicable in other procedure-driven organizations and environments. In the legal domain, for example, regulations mandate actions required for various business transactions, and thus are considered types of procedure. Individuals engaged in such transactions generally conform to regulations, but under certain circumstances will deviate from the regulations when individual goals are threatened. In this respect, the model has shown some promise in explaining actions taken in the context of mergers and acquisitions regulations, and in addressing the issues of management fraud. In the medical domain, the model is potentially useful in understanding physician interaction with treatment protocols. As in the other domains, a physician will tend to follow protocols -- except when doing so might endanger the health or life of the patient.

Future research will entail evolving the model from its current 'commentator' role into one that is capable of interacting directly with an operator. Initially, direct decision support to operators can be achieved through relatively minor adjustments in the model's interface. For example, the model can be modified to read trajectory data and share its generated response plans with operators in real-time, in order to provide an indication of
alternatives both within and outside of procedures. Also, because plans are based on various goal hierarchies within the model, a rationale supporting alternative response plans also would be available to operators, by default.

Longer-term enhancements would include enriching the model's knowledge base of goals, plans and causal relations, and allowing the model to assess the relative quality and impact of the various types of response plans. This would permit an operator to propose actions and solicit feedback from the model, in a type of 'what-if' dialogue that is typical of many DSSs. It would also potentially allow the model to infer the reasoning, rationale and goals underlying an operator's proposed action -- even for actions that might be considered 'ill-advised'. This type of user-modeling could allow the DSS to propose alternative actions within the broader context of an inferred operator goal structure.

In conclusion, our research establishes a framework for a general explanatory model of organizational knowledge and behavior that exploits the important role of procedures in understanding and explaining intentional human actions. The theory embodied in the model has general implications for human-computer collaboration in complex systems. In a general sense, by exploring how and why actors sometimes feel compelled to work outside of imposed procedures, this study lays the groundwork for an improved computationally-based method of procedure execution -- one that is capable of engaging in a collaborative problem-solving dialogue with a human decision maker.
REFERENCES


Causal relationships

Automated interventions

Goals

Operator

Procedures

System Output

Decision Support System

• Operator
• Supervisor
• Auditor

Process Control System

Physical Processes (Plant)

Figure 1
Global Expectation Object
violated-mgmt-goal
goal-hierarchy
expected-actions
goal-compatibility?: Yes
violation-source?: external

Management-goal object
goal: (< (rate charging-flow) (max-value charging-flow))
parameter: charging-flow
envisioned?: nil

Restoration-goal object
goal: (< (rate charging-flow) (max-value charging-flow))
parameter: charging-flow
sub-goal/method

Restoration-goal object
goal: attain-additional-flow
parameter: charging-flow

Intervention object
action: engage
patient: BIT
instantiated-implementation-plans
any-available-pathway?: yes
corresponding-m-node: MN-ES-1-STEP-8A-
path-expectations

Implementation-plan object
behavior-characterization: condition-violation
associated-decision-process
encompassing-action

Path expectation object

Intervention object

Step objects

Procedure Goal Objects

Figure 2
monitor events

[ event type? ]

response event

system event

Plan Generation

Generate ad hoc plan(s)

Update expectation framework

Plan Induction

Match behavior to ad hoc action plan?

no

Match behavior to a priori procedural action?

no

yes

Present explanation

yes

system state:
- p-goal hierarchy
- m-goal hierarchy
- r-goal hierarchy
- identified pathways
- decision variables
  available pathway(s)
  condition conformance
  assumption conformance
  goal violation source

Figure 3
1. I’m going to have to make a judgment call.
2. We’re going to [procedure ES-1.2]
3. We’re going to go to [procedure ES-1.2].
4. Doing it on max charging
5. isn’t the way to be going.
6. Realign through the BIT
7. and go to [ES-1.2].
(s param-value-change
  :actor charging-flow
  :patient rate
  :new-value 70)

(s intervention
  :actor operator
  :patient charging-header-flow-control-valve
  :action operate
  :action-status implemented)

(s intervention
  :actor operator
  :patient charging-flow
  :action control
  :action-status implemented)

(s param-value-change
  :actor przr-level
  :patient behavior
  :new-value 'stable)

(s intervention
  :actor operator
  :patient bit
  :action engage
  :action-status implemented)

Figure 5
The value of CHARGING-FLOW has changed. Investigating impact on management goals:

(PRIMARY-CONTROLLER-LIMIT-1 PRIMARY-CONTROLLER-LIMIT-2)

The following goal for CHARGING-FLOW is now in violation:

[Mgmt-goal: (< (RATE CHARGING-FLOW) (MAX-VALUE CHARGING-FLOW))]

Retainment of this management goal is required in order to [ACHIEVE PRIMARY-LEVEL-CONTROLLE-FLEXIBILITY]

Figure 6
From the observed management goal violation, we can establish alternative hypotheses regarding subsequent actor responses. The actor's reactive planning could proceed as follows:

Achieve: (< (RATE CHARGING-FLOW) (MAX-VALUE CHARGING-FLOW))
  Achieve: ATTAIN-ADDITIONAL-FLOW
    ENGAGE BIT
  Achieve: REDUCE-CONTRARY-FLOW
    REDUCE LETDOWN-FLOW
  Achieve: SET-FLOW-TO-DECREASING
    CLOSE CHARGING-HEADER-FLOW-CONTROL-VALVE

From this hierarchy, the following restorative actions are available:

ENGAGE BIT
REDUCE LETDOWN-FLOW
CLOSE CHARGING-HEADER-FLOW-CONTROL-VALVE

Figure 7
The actions observed thus far can be explained within the constraints of the following single procedural pathway.

*** Path 1:  
MN-ES-1.1-STEP-1A  
-> MN-ES-1.1-STEP-2A  
-> MN-ES-1.1-STEP-3A  
-> MN-ES-1.1-STEP-3B  
-> MN-ES-1.1-STEP-4A  
-> MN-ES-1.1-STEP-5A  
-> MN-ES-1.1-STEP-6A *  
-> MN-ES-1.1-STEP-6B *  
-> MN-ES-1.1-STEP-6C  
-> MN-ES-1.1-STEP-6D  
-> MN-ES-1.1-STEP-6E *  
-> MN-ES-1.1-STEP-6F  
-> MN-ES-1.1-STEP-6G *  
-> MN-ES-1.1-STEP-7A  
-> MN-ES-1.1-STEP-7B  
-> MN-ES-1.1-STEP-8A  
-> MN-ES-1.1-STEP-8A~ **  
-> MN-ES-1.2

The observed action corresponds to the prescribed step MN-ES-1.1-STEP-8A~.

Figure 8
The observed action is in violation of the permissive for node MN-ES-1.1-STEP-8A~
[i.e., (NOT (EQUAL (BEHAVIOR PRZR-LEVEL) 'STABLE))]
which means that the actor is operating outside of procedural constraints.

The latest observation conforms to a previously cited behavioral expectation.

Argumentation in defense of the actor's behavior is constrained by the
values of the applicable decision variables and the associated implementation
plan / decision process, which is:

---------------------------------------------------------------
DP-5:
Attempt to conform system state to CONDITION
  If conformance is successful, EXECUTE-PROGRAM-STEP
  Else
    If CONDITION dominant:  ACT-INDEPENDENTLY-OF-PROGRAM
    If ASSUMPTION dominant:  EXECUTE-PROGRAM-STEP
    If BOTH dominant:  ACT-INDEPENDENTLY-OF-PROGRAM
    If NEITHER dominant:  EXECUTE-PROGRAM-STEP
---------------------------------------------------------------

The values of the applicable decision variables, as defined by this
implementation plan, are:

  Source of program violation: EXTERNAL
  Available pathway? YES
  Condition satisfied? NO
  Assumption satisfied? YES

In the context of the implementation plan, the following abstract action
is consistent with observed behavior: EXECUTE-PROGRAM-STEP

Figure 9
In the framework of this implementation plan and chosen action, the actor's expected argument is formed as follows:

Goals are compatible. Therefore, a condition violation can be defended by appealing to the primacy of the mutually-desired management goal:

\[ \text{Mgmt-goal: } (\lt \text{(RATE CHARGING-FLOW)} \lt \text{(MAX-VALUE CHARGING-FLOW)}) \]\n
Although the condition for step [MN-ES-1.1-STEP-8A~] has been violated, an assumption governing execution of the step is valid in the current situation.

The assumption is derived as follows:

Execution of step MN-ES-1.1-STEP-8A~ contributes to achievement of a program goal [ACHIEVE ATTAIN-ADDITIONAL-FLOW] that corresponds to an identical restoration goal. In violating the condition, the actor may be trying to restore the currently violated management goal by using the program.

No other matrix node serves to achieve the restoration goal [ACHIEVE ATTAIN-ADDITIONAL-FLOW].

An implicit assumption of step MN-ES-1.1-STEP-8A~ suggests that attainment of the program goal associated with MN-ES-1.1-STEP-8A~ is required.

Because of the need to restore the violated management goal \[ [[\text{Mgmt-goal: } (\lt \text{(RATE CHARGING-FLOW)} \lt \text{(MAX-VALUE CHARGING-FLOW)})]] \], that assumption is valid in the current environment.

Figure 10
FIGURE LEGENDS

Figure 1: Distributed decision making and behavior in a complex, process-control environment, comprising the basis of the underlying model in the DSS.

Figure 2: Knowledge structures comprising the Global Expectation Network.

Figure 3: Process model of plan generation and induction

Figure 4: Transcript segment showing operator violation of procedure condition.

Figure 5: Sequence of ISLOCA events entered into the model, represented as objects in Lisp. The first and last events (circled) are most important, because they invoke the model's plan generation and induction processes, respectively.

Figure 6: Model evaluation of the impact of the initial event on management goals (m-goals).

Figure 7: Model retrieval of potential reactive plans formed by the operator in response to the m-goal violation, and actions associated with those plans.

Figure 8: Initial model interpretation of the event within the context of a procedure pathway and step.

Figure 9: Model recognition that the condition for execution of the step has not been satisfied, confirmation of previously-recorded expectation, and explanation of the violation action in the context of the associated implementation plan.

Figure 10: Model inference of the rationale or argument supporting operator violation of the procedure.