A slightly edited version of this paper appears in
Proc. 1998 Conf on Foundations of Computer-Aided Process Operations [FOCAPO]

Applying a Procedural and Reactive Approach to Abnormal Situations in Refinery Control

Kurt D. Krebsbach and David J. Musliner
Automated Reasoning Group
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
{krebsbac,musliner}@htc.honeywell.com

Abstract

Oil refineries literally provide the lifeblood for global economic health, and disruptions to their operations have major worldwide impact. We are developing a large-scale intelligent refinery control system to assist human operators in controlling refineries during abnormal situations. Based primarily on reactive and procedural approaches to intelligent behavior, the Abnormal Event Guidance and Information System (AEGIS) will interact with multiple users and thousands of refinery components to diagnose and compensate for unanticipated plant disruptions. Through intelligent autonomous behavior and improved human situation awareness, the AEGIS project is expected to have a multi-billion dollar annual impact on refinery productivity. This paper discusses lessons learned during the initial prototyping efforts of the goal-setting, planning, and plan execution components of AEGIS.

Introduction

One of the largest industrial disasters in U.S. history was a $1.6 billion explosion at a petrochemical plant in 1989. This accident represents an extreme case within the spectrum of major process disruptions, collectively referred to as *abnormal situations*. While most abnormal situations do not result in explosions, they can be extremely costly, resulting in poor product quality, schedule delays, equipment damage, reduced occupational safety, and environmental hazards. The inability of automated control systems and plant operations personnel to control abnormal situations has an economic impact of at least $20 billion annually in the petrochemical industry alone.

At the Honeywell Technology Center, we are building an intelligent, mixed-initiative refinery control system designed to dramatically reduce the frequency, severity, duration, and cost of abnormal situations. The Abnormal Event Guidance and Information System (AEGIS) is a large-scale distributed intelligent system specifically designed both to assist operations personnel (e.g., by displaying the most useful information) and to take diagnostic and compensatory actions autonomously.

This paper describes a portion of the goal-setting, planning, and execution (GPE) components of AEGIS. Although a detailed description of the entire AEGIS system is beyond the scope of this paper, we consider the requirements and constraints that guided our approach, and evaluate the current prototype with respect to them. In particular, we report on the benefits and the challenges raised in our attempt to satisfy the often conflicting requirements inherent in the enormously complex domain of oil refining.

In the next section, we briefly describe the current state of refinery control and the associated problems. We then overview the AEGIS architecture, focus on the goal-setting, planning, and execution components, and discuss the lessons learned in prototyping those functions.

Background: Refineries and Control

Petrochemical refining is one of the largest industrial enterprises worldwide. The functional heart of a refinery, and the most economically critical component, is the Fluidized Catalytic Cracking Unit (FCCU). As illustrated in Figure 1, the FCCU is primarily responsible for converting crude oil (feed) into more useful products such as gasoline, kerosene, and butane (Leffler 1985). The FCCU *cracks* the crude's long hydrocarbon molecular chains into shorter chains by combining the feed with a catalyst at carefully controlled temperatures and pressures in the riser and reactor vessels. The resulting shorter chains are then sent downstream for separation into products in the fractionator (not shown). The catalyst is sent through the stripper and regenerator to burn off excess coke, and is used over again.

Figure 2 illustrates how a typical state-of-the-art refinery is controlled. The Distributed Control System (DCS) is a large-scale programmable controller tied to plant sensors (e.g., flow sensors, temperature sensors), plant actuators (e.g., valves), and a graphical user interface. The DCS implements thousands of simple control loops (e.g., PID loops) to make control moves based on discrepancies between setpoints (SPs) and present values (PVs). For example, as depicted in Figure 1, the dotted line connecting the temperature sensor and the riser slide valve denotes that the position of the slide valve is dependent on the temperature being sensed in the riser. As the temperature drops, the slide valve will be opened to increase the flow of hot catalyst. A typical FCCU will have on the order of one thousand readable "points," and a few hundred writable "points." In addition to PID control loops, the DCS can be programmed

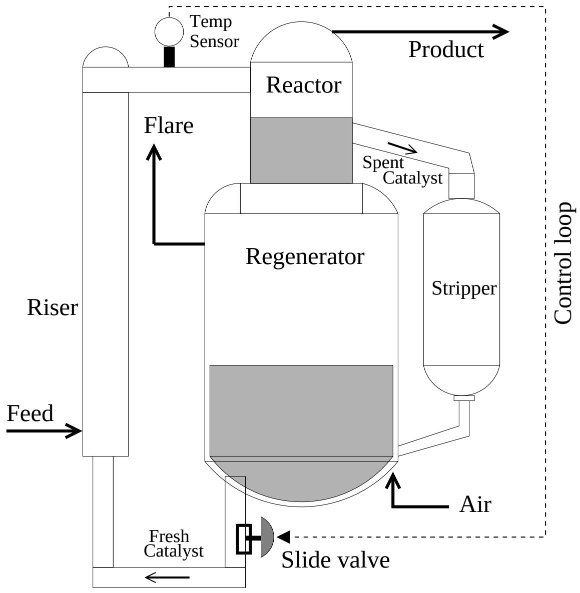

Figure 1: A Fluidized Catalytic Cracking Unit.

with numerous "alarms" that alert the human operator when certain constraints are violated (e.g., min/max values, rate limits). "Advanced control" is the industry term for more powerful mathematical control techniques (e.g., multivariate linear models) used to optimize control parameters during normal operations.

The human operators supervise the operation of the highly-automated plant. This supervisory activity includes monitoring plant status, adjusting control parameters, executing pre-planned operations activities (e.g., shutting down a compressor for maintenance), and detecting, diagnosing, compensating for, and correcting abnormal situations. The operator has a view of the values of all control points, plus any alarms that have been generated. The actions the operator is allowed to take include changing SPs, manually asserting output values for control points, and turning on or off advanced control modules.

Abnormal Situations

During abnormal situations, all hell breaks loose. Minor incidents may cause dozens of alarms to trigger, requiring the operator to perform anywhere from a single action to dozens, or even hundreds, of compensatory actions over an extended period of time. Major incidents may precipitate an *alarm flood*, in which hundreds of alarms trigger in a few seconds, leading to scrolling lists of alarm messages, panels full of red lights, and insistent klaxons. In these situations, the operator is faced with severe information overload, which often leads to incorrect diagnoses, inappropriate actions, and major disruptions to plant operations. If left uncontrolled, abnormal situations can be extremely costly, resulting in poor product quality, schedule delays, equipment damage, reduced occupational safety, and environmental hazards.

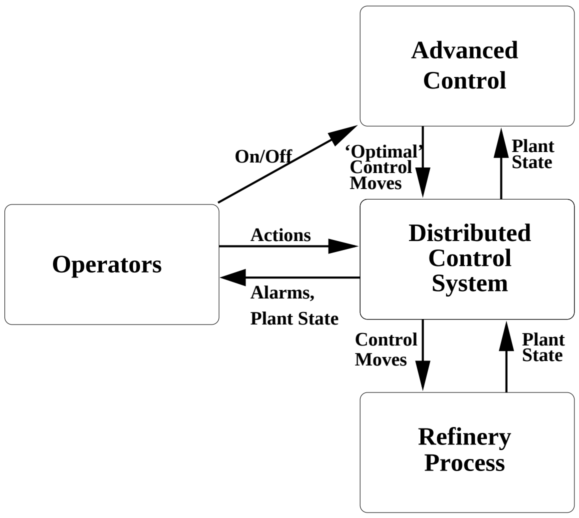

Figure 2: Refinery Control without AEGIS.

Because abnormal situations are so serious, many regulatory and administrative structures are already in place to help manage them. Primarily, operators are trained to respond to abnormal situations based on extensive Standard Operating Procedures (SOPs) that are written down, checked, and updated regularly. The procedures can be quite long (dozens of pages), with lots of logical control structure and contingencies, since the exact state of the plant is almost never known with certainty. Many procedures involve sampling data, confirming other readings, performing diagnostic tests, conferring with other plant personnel, and adjusting DCS control parameters. Some procedures apply to extremely general contexts (e.g., we're losing air pressure from somewhere), while some are less general (air compressor AC-3 has shut down), and some are very specific (the lube oil pump for AC-3 has a broken driveshaft).

Aegis

The Abnormal Event Guidance and Information System (AEGIS) is a large-scale distributed intelligent system designed primarily to improve responses to abnormal situations, both by automating some activities currently performed by operations personnel and by improving human situation awareness. Illustrated in Figure 3, AEGIS is a distributed software architecture based on blackboard-style communications and several distinguished application roles. Multiple application programs, with varying levels of intelligence and abilities, may fill roles including:

State Estimator — Determines the state of the plant, at varying levels of abstraction, by fusing diverse sensor data and other available information (e.g., prior control moves, known malfunctions, human observations).

Goal Setter — Decides which of the currently-threatened operational goals should be addressed.

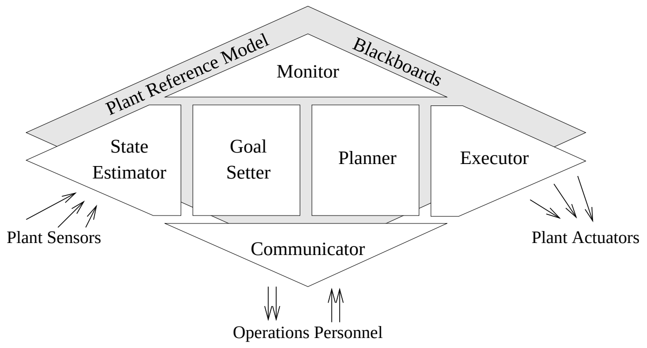

Figure 3: The AEGIS architecture.

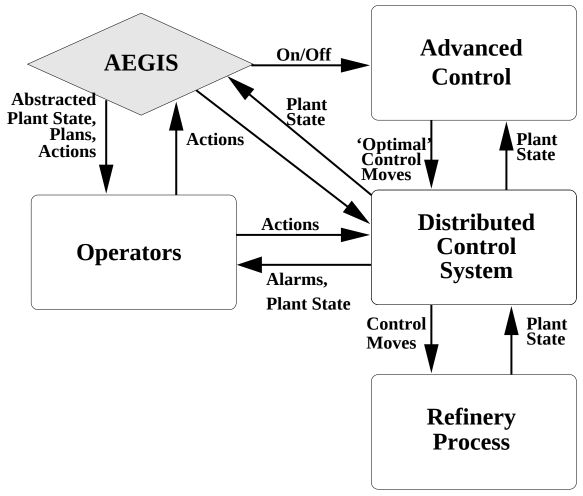

Figure 4: Refinery Control with AEGIS.

Planner — Develops plans to address threatened goals selected by Goal Setter.

Executor — Executes plans, monitoring action outcomes and updating other AEGIS components on progress towards goals.

Communicator — Communicates efficiently and effectively with multiple plant personnel including DCS operators and field personnel located outside the control room.

Monitor — Observes the performance of the AEGIS components and may adjust or adapt the system's behavior in response to observed performance.

These functions interact by exchanging information on shared blackboard data structures. The Plant Reference Model blackboard captures descriptions of the refinery at varying levels of abstraction and from various perspectives, including the plant's physical layout, the logical processing unit layout, the operational goals of each component, and the current state and suspected malfunctions, with associated confidence levels. Figure 3 shows how AEGIS interacts with the existing system.

Advantages of AEGIS

AEGIS emphasizes two main design concepts that form the basis of many of its advantages over the state-of-the-art:

Goal-centric (not Alarm-centric) Information — Raw data interpretation and alarm flood management are enormous tasks currently left to the board operator. In the midst of a plant upset, the operator has neither the time nor the information to properly evaluate what is going on. A hallmark of the AEGIS approach is an abstraction of data and alarms into more useful information such as threatened operational goals, likely malfunctions and their confidence values, relevant symptoms, grouped process data, and trends.

Mixed-Initiative Plan Execution — Currently, besides being responsible for evaluating the plant state, board operators must choose appropriate courses of action, perform each task or delegate tasks to others, and monitor the progress of these tasks, while simultaneously reevaluating the next context. Many of these tasks are easier for AEGIS to perform. For instance, AEGIS can perform any number of tasks as parallel threads, removing the serialization often imposed when the human operator himself becomes a limited resource. Monitoring for the expected effects of actions is also a tedious and error-prone task for an operator, but it is a simple matter to make AEGIS procedures self-monitoring, with little or no loss of attention to concurrent activities.

GPE Requirements

In this paper we focus specifically on the goal-setting, planning, and execution components of the larger AEGIS system. We refer to this aggregate functionality as GPE. The major requirements placed on the GPE functions include:

Semi-autonomy — GPE is semi-autonomous and mixed-initiative: many of the actions it is designed to take can be performed either by AEGIS or by the human operator.

Procedural Orientation — As discussed above, responses to abnormal situations are dictated by formal procedures, many of which are already recorded in plant documentation.

Reactivity — While not hard real-time, the refinery domain requires rapid responses (no more than a few seconds) to rapidly changing environmental conditions; GPE must be able to quickly change its focus of attention *and its plans* at any time.

Lack of Models — While some partial analytical and simulation models exist for elements of refineries, these models are not tremendously useful for GPE's task for several reasons, including:

- Abnormal situations, the focus of AEGIS, are precisely the times when the plant is behaving outside of its normal, modeled modes.

Figure 5: The Procedural Reasoning System Architecture.

- Existing models are not sufficiently detailed for first-principles generation of actions spanning large upsets.

GPE: A Procedural Approach

We have chosen to prototype the core reasoning engine of AEGIS in C-PRS, the C-based version of the Procedural Reasoning System (Ingrand 1994; Ingrand, Georgeff, & Rao 1992; Georgeff & Lansky 1986). As shown in Figure 5, knowledge in PRS is represented as a declarative set of facts about the world, together with a library of user-defined *knowledge areas* (KAs) that represent procedural knowledge about how to accomplish goals in various situations. Goals represent persistent desires that trigger KAs until they are satisfied or removed. The *intention structure* represents currently-selected KAs that are in the process of executing or awaiting execution, in pursuit of current goals. The PRS *interpreter* chooses KAs appropriate for current goals, selects one or more to put onto the intention structure, and executes one step from the current intention.

We chose to use an integrated approach to goal setting, planning, and execution based on the AI community's past experiences with autonomous systems applied to real-world domains (e.g., robotics). That experience has shown that choosing a goal to pursue, planning a course of action, and executing the steps of the plan are inevitably intertwined by the unpredictable and dynamic nature of real-world domains. Execution failures, changing goals, difficult planning problems, and environmental changes all disrupt the ideal of simple forward information flow. If the GPE functions were separated into distinct programs, the amount of information constantly passing back and forth due to the changing domain, plans, and goals would be overwhelming. In our integrated GPE approach, in contrast, those changes are kept largely local to GPE, so the C-PRS interpreter can be efficient about managing that information.

Other features of PRS which have proven to be extremely useful for this domain include the following:

- The hierarchical, subgoaling nature of the procedural representation allows PRS to combine pieces of plans in novel ways, which is important for flexible plan execution and goal refinement.
- Its ability to pursue multiple, **goal-directed** tasks while at the same time being **responsive** to changing patterns of events in bounded time.
- Its ability to construct and act on **partial** (rather than complete) **plans**.
- Its **meta-level** (or reflective) reasoning capabilities, an important feature for controlling the allocation of processing resources, planning attention, and alternative goal achievement strategies.
- Its knowledge representation assumptions, which encourage **incremental refinement** of the plan (procedure) library, an enormous advantage for large-scale applications.

PRS and AEGIS

The GPE world model consists of a database of facts and beliefs. The database is populated with fairly static information about the plant's physical layout and logical connections between plant components, as well as data dynamically requested regarding attributes and values of DCS points. The GPE can subscribe and unsubscribe to this data on an as-needed basis, but subscribes to some types of information, such as the status of *operational* goals and malfunction confidence values, on a permanent basis.

As this data changes at run-time, procedures from the plan library are triggered, and new *procedural goals* are established. As procedures are selected to achieve procedural goals, they are represented on PRS' intention structure. A user-viewable representation is also generated, and is available to the user through an interface called GPEView. From GPEView, an operator can view skeletal plans, authorize or cancel those plans prior to execution, assume responsibility for pieces of them, and so on. These plan modifications are then reflected in the PRS database, and are incorporated into the procedure's runtime behavior.

Many actions on the intention structure can be directly executed by the GPE, given authorization from the user. These actions include actual DCS control moves, communication messages with field personnel, and requests for more data.

Benefits

Our PRS-based approach naturally provides several benefits especially pertinent to handling abnormal situations, which we briefly outline in this section.

Standard Operating Procedures

An obvious benefit to our approach is that much of the knowledge we wish to encode is already available in refineries as paper SOPs. While it is clear that translation from SOPs into PRS procedures is not trivial, they have provided us with a great deal of insight into the role of the operator, the culture of the refinery, and the current state-of-the-art. We will discuss our observations and suggested extensions to this basic model in the next section.

Parallel Goal Achievement

During an abnormal situation, a human operator must be focused to properly respond, despite an avalanche of data, cascading effects, and a plethora of pending tasks. This difficult situation can tax even the most experienced operator's time, memory, and communication constraints. The result: a wide variety of errors and inefficiencies in procedure execution.

Because operators represent a scarce resource themselves, SOPs are almost always expressed sequentially, to aid the operator in focusing on one thing at a time. GPE effectively has no such constraint, and can react to multiple goals in parallel, while allowing the operator to focus on the highest priority tasks requiring his attention. An interesting corollary is that task prioritization is only relevant for GPE tasks in cases of resource contention. In fact, in our analysis of plant procedures, instances of tasks requiring serial execution are relatively rare once the operator is no longer the constraining resource, often making GPE procedures of much shorter duration than their manually-executed counterparts.

Context-Sensitive Behavior

PRS provides numerous ways to specify context-sensitive triggering of procedures. This is much more flexible than plant SOPs, in which one procedure is often recommended to achieve a goal regardless of the many other factors comprising the current context. For example, one can specify multiple procedures to accomplish the same goal of replacing lost combustion air: one if the secondary pump is available, one if the air loss is below an important threshold, several if the root cause of the malfunction is not yet known, and so on. While several or all of these procedures might be relevant to the goal, the context we describe can distinguish those that truly apply. Further, using (natively available) priorities, or user-defined metalevel reasoning, the interpreter can intelligently select the most preferred among the resulting set of applicable procedures. Finally, to combine the last two benefits, one can describe in the metalevel that the preferred behavior involves attempting several of the applicable goal-achievement methods in parallel.

Action Effect Monitoring

Many of the hardest tasks for humans to perform reliably involve monitoring the effects of earlier actions. Currently, operators must simply remember to check process data at an appropriate later time to confirm that earlier actions are having their desired effects. Because the delays between actions and their observable effects can vary widely, this presents a difficult, and often ignored, tracking problem.

Fortunately, because PRS is not memory, time, or communication-constrained to any significant degree, GPE procedures can quite easily be set up to be self-monitoring, as long as methods exist for confirming goal achievement. In our domain, these methods involve querying the DCS to confirm temperature trends, pressure differentials, and the like, all of which are trivially available. Other more complex confirmatory information can be obtained directly from the operator or other AEGIS components (e.g., state estimators), and at least provide a safeguard against forgetting the confirmation altogether. Feedback from plant personnel preliminarily indicates that this automatic monitoring functionality is among the most immediately and widely useful aspects of the PRS approach.

In the following sections we discuss specific challenges with our GPE approach, and some preliminary solutions.

Mixed-Initiative Procedure Execution

As with most systems, GPE has competing requirements. During an upset, it is important for GPE to be constantly sensitive to the rapidly-changing plant state, and to respond quickly. On the other hand, a key AEGIS design goal is maintaining user awareness. Unfettered, the lightning fast responsiveness of AEGIS computers could leave users bewildered about what actions the system intends to perform or has already performed. We have spent significant effort addressing this challenge of effectively supporting mixed-initiative, reactive procedure execution. In the following subsections we elaborate on the difficulties in using PRS for a mixed-initiative system, and describe our current solution.

Lack of Projection

Because PRS is reactive, it does not look ahead to determine which procedure it will select to achieve a given goal until that goal has been reached in the procedure. We believe this is "correct" from an engineering perspective, because the precise method of achieving a goal should not be determined until the full environmental context is available for evaluating the alternatives. This context can only be known when the goal is posted, not before. However, this is insufficient from the operator's perspective, because it provides little insight into what the system is planning globally.

There are three aspects to this problem, within the context of executing a single PRS procedure:

- **Future goal-achieving procedures are not yet selected.** PRS procedures are, in the simplest serial case, executed like a normal computer program[1]. When PRS selects a procedure, it instantiates it, and sets the "program counter" at the first goal. Applicable procedures are determined to achieve that goal, and one is chosen. While this newly-chosen procedure is being executed however, selection of procedures for goals beyond the program counter is deferred.
- **Future goals and actions are known but not available to the interpreter.** Although the names of goals and primitive actions beyond the program counter are available in the procedure source code by inspection, they are not available to the C-PRS interpreter until the program counter arrives.

[1] C-PRS also supports parallel goal achievement, but that capability does not affect this discussion.

- **Future goals and actions are not necessarily meaningful to the user.** Even if future goals and actions could be accessed by the interpreter, some are at the wrong level to be relevant to the user (e.g., binding a local variable), while others are not in a form useful to an operator (code), or easily translatable into such a form. In general, it is not reasonable to expect the PRS procedure author to use names and constructs that correspond to an operator's understanding, and vice versa.

Pseudo-Projection

To work around this problem, we have developed a "pseudo-projection" method that allows GPE to appear partially projective without making any changes to the reactive PRS interpreter. Pseudo-projection allows the operator to see as far into the future, and with as much detail, as is possible given the reactive procedural paradigm.

We implement pseudo-projection using a procedure annotation syntax that allows the author to annotate each procedure with a series of comments that the AEGIS user will see at runtime when the procedure is chosen by PRS. These annotations, called *metacomments*, allow PRS to appear partially projective to the user. As soon as a procedure is selected, the user can see the entire structure and status of the procedure.

This metacomment technique is a temporary approach to the problem of user awareness in a reactive system, and suffers from several serious deficiencies. First, it adds complexity to the process of writing procedures, although the metacomment syntax itself is quite simple. In part, this added complexity is unavoidable if we wish the user to see a representation of the procedure that is somehow simplified, abstracted, or in different terms than the raw procedure code itself.

Other Forms of Projection

While pseudo-projection techniques provide a form of lookahead for the user, other limited forms of model-based projection can be exploited which allow more intelligent control by the reactive system itself. Consider the following simplified procedure segment for responding to a loss of combustion air:

```
Procedure Novice-Air-Loss-Response

  1. Cut riser temperature to 930 degrees F.
  2. Eliminate all residual feed.
  3. Eliminate all slurry pumparound feed.
  4. Cut main feed to 20,000 barrels/day.
  5. Add pure oxygen up to 30% enrichment.
```

This procedure fragment is a typical SOP example. They are characterized by simple instructions, understandable by even the most novice operator by design. They are straightforward, safe, static, and suboptimal. In this example, for instance, all residual and slurry feeds are eliminated to allow the operator to concentrate on cutting and monitoring only the main feed.

While these procedures provide a starting point for encoding executable procedures, they do not accurately reflect the complexity of most operators' response to an abnormal situation. As operators gain experience, their knowledge of the underlying plant process and DCS response grows, and their response becomes more model-based. For instance, the operators we interviewed noted that they would generally leave in some residual feed to keep the coke component higher, keeping the riser temperature higher. This is an optimization step that, while still safe, maintains a higher level of production, and thus reduces the cost of the disruption.

Mini-Models

In general, the more experienced the operator, the more context-sensitive is his response to an abnormal situation. We view our GPE procedures as evolving in the same way, incorporating more of what we have called *mini-models* directly within PRS procedures. As the authors of the procedures gain a better understanding of the process and control system, we expect the procedures to rely less on static responses, and more on computing over a simplified model to generate a context-sensitive response. For instance, the following is a more model-based version of the same procedure, emulating the expert-operator approach:

```
Procedure Expert-Air-Loss-Response

  1. Compute amount of O2 in lost air.
  2. Add pure O2 to replace lost O2,
        up to 30% enrichment max.
  3. Compute O2 left to replace.
  4. Compute amt of carbon this corresponds to.
  5. For each feed source:
     5a. Cut source accding to carbon factor.
  6. Set riser temperature setpoint
        based on remaining carbon.
```

This procedure concentrates on balancing carbon content of the current feed sources with the amount of oxygen available, while staying within safety limits of 30% enrichment. It is based on a simplified mini-model involving a handful of important factors in the process, and is thus much more tailored to the actual circumstances at the process at the time of its invocation. In this example, GPE can greatly assist the operator by easily and automatically computing parameters to the situation response (e.g., correct riser temperature), as well as providing the option for GPE to take the actions autonomously, and monitor the effects of these actions over time.

Existing Predictive Models

In addition to mini-models directly implemented with PRS procedures, small predictive models exist as black-box applications for very limited pieces of the refinery. While these models are quite small (e.g., ten inputs, four outputs), in certain contexts they can be invoked from within a PRS procedure to provide several valuable types of information. First, in many circumstances, GPE has several possible courses of action. By projecting these models forward in time for each option, GPE can more accurately assess the effectiveness of each alternative and choose the best one. Secondly, the specific results of the projections can often be valuable information to the operator and to GPE. In cases where the results are close, for instance, the operator might prefer one method over another for less tangible reasons than GPE is able to consider. From GPEś perspective, the results form a rank ordering of the options, which can be cached and used in case the first goal-achievement method fails. Finally, the specific expected results can inform GPE in establishing its own monitoring parameters.

Conclusions

This paper reports on the current status of an ambitious project to build an intelligent, mixed-initiative refinery control system. The current GPE prototype includes procedures that are successfully able to handle a variety of failures and disruptions to the air feed system of a simulated FCCU. The simulator is a high-fidelity industrial refinery simulator used to train plant personnel. The level of knowledge in the prototype GPE is not yet equivalent to even a rookie DCS operator, but the approach shows promise and has been successfully demonstrated to enthusiastic industry participants. Current GPE-related efforts are centered around limited field tests of the technology in actual oil refineries, as well as research into user interaction semantics and methods for automating user involvement with the system.

References

Georgeff, M., and Lansky, A. 1986. Procedural knowledge. *IEEE Special Issue on Knowledge Representation* 74:1383–1398.

Ingrand, F.; Georgeff, M.; and Rao, A. 1992. An architecture for real-time reasoning and system control. *IEEE Expert* 7:6:34–44.

Ingrand, F. F. 1994. *C-PRS Development Environment (Version 1.4.0)*. Labege Cedex, France: ACS Technologies.

Leffler, W. L. 1985. *Petroleum Refining for the Non-Technical Person.* Tulsa, OK: PennWell Publishing Company.