Appears in Working Notes of the AAAI Spring Symposium on Adjustable Autonomy
Stanford University, March 22-24, 1999

Adjustable Autonomy in Procedural Control for Refineries

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

Abstract

Oil refineries 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. Adjusting the autonomy of AEGIS's behavior is a key requirement for success in the dynamic, highly-unpredictable refinery environment. This paper discusses our procedural and reactive approach to the goal-setting, planning, and plan execution components of AEGIS, and the adjustable autonomy features they support.

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 the goal-setting, planning, and execution (GPE) components of AEGIS, focusing on the adjustable autonomy aspects of the GPE design. 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 adjustable autonomy features we have currently implemented and are planning to develop. We conclude with a few lessons learned while prototyping GPE.

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

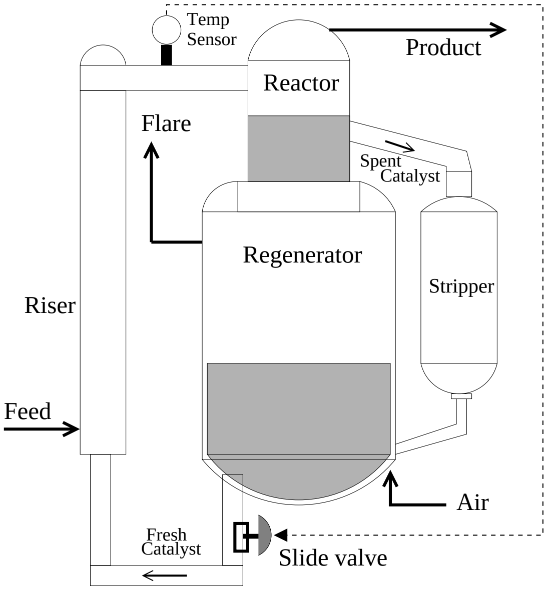

Figure 1: A Fluidized Catalytic Cracking Unit.

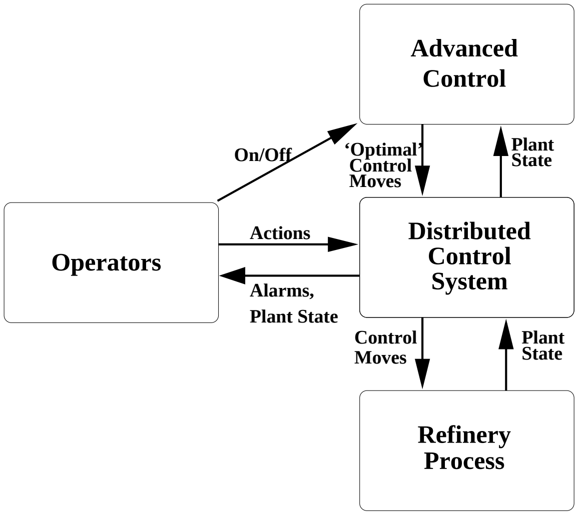

Figure 2: Refinery Control without AEGIS.

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 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.

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).

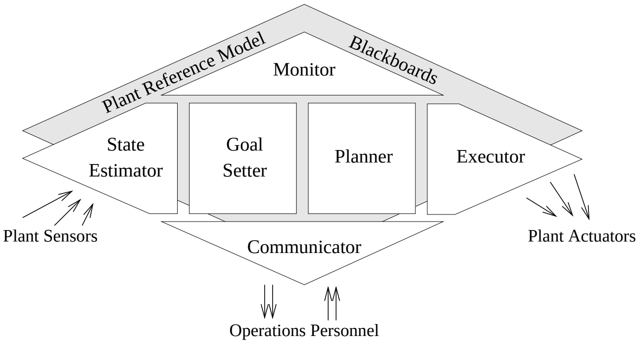

Figure 3: The AEGIS architecture.

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.

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.

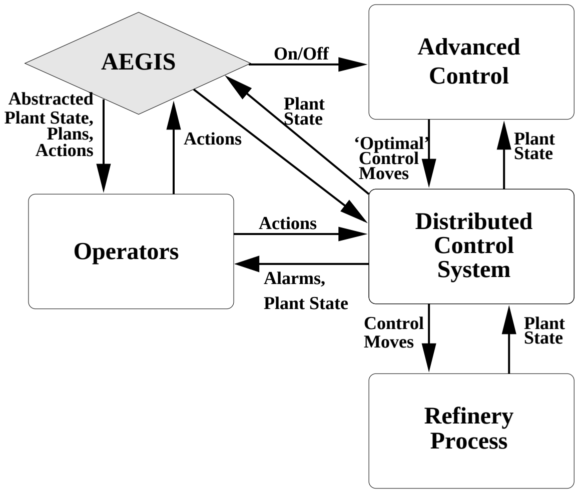

Figure 4: Refinery Control with AEGIS.

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.
- 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.

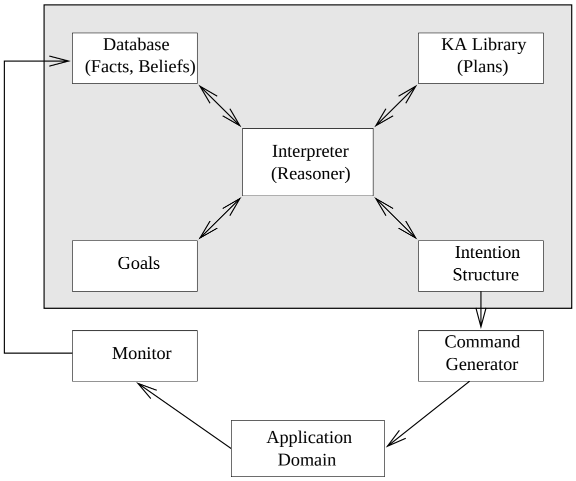

Figure 5: The Procedural Reasoning System Architecture.

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. 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 monitor progress. 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 GPE, given authorization from the user. These actions include actual DCS control moves, communication messages with field personnel, and requests for more data.

Adjustable Autonomy

GPE supports a fairly broad variety of "adjustable autonomy" behaviors by combining two basic features: hierarchical task representations and the simple ability to have either the user or the system execute any of those tasks. In principle, these features allow GPE to support all forms of "combinatorial" adjustable autonomy, in which the potential for adjustment of autonomous functions corresponds to the potential boolean combinations of functions that can be performed by the system rather than a human. In other words, combinatorial adjustable autonomy lets the system execute any combination of the overall set of tasks it is capable of, as selected by whatever is doing the "adjusting" (be it the human user, a learning/monitoring function, etc.).

In practice, GPE supports a somewhat more limited set of adjustable autonomy functions, largely due to domain restrictions. For example, there are functions that *only* GPE can perform, including updating its internal models and communicating with other elements of the AEGIS architecture. Also, there are functions that GPE *cannot* perform, such as closing valves that have no automatic controllers; in these cases, GPE can request actions by human field operators. Finally, there are functions that the plant engineers choose not to put under GPE control.

The combinatorial adjustable autonomy notion also oversimplifies the degree to which the user can be responsible for an action. In practice, there are less important actions that GPE can simply cede entirely to the user, and there are more critical actions that GPE may not want to completely forget about. For those cases, the knowledge engineer who encodes the GPE procedures may add domain-monitoring functions that watch the progress of the system even when the human takes primary responsibility for executing some procedure steps. In that case, the system is remaining attentive to an intended result, but it is largely relying on the human to actively try to achieve that result.

GPE's level of autonomy is controlled online as the human interacts with three aspects of the system:

Responsibility — GPE maintains an active representation of which agent is responsible for every procedure and action that the system believes is currently intended. If a user has responsibility for a procedure or primitive action, GPE may monitor for the expected effects of that action or watch the clock to make sure the action is happening quickly enough, but GPE will not try to execute the action. If GPE has responsibility for a procedure or action, it will proceed immediately *if authorized.* Currently, GPE does not distinguish between different users, but only between itself and users. Eventually it will understand the ongoing tasks of different operators, and may actively manage their load.

Authorization — GPE's ability to execute its procedures is controlled by a uniform *authorization* mechanism. Only when GPE has both responsibility and authorization will it proceed to execute actions or decompose a high-level procedure. The knowledge engineer who encodes the GPE procedures may pre-authorize some procedures (e.g., those that are well-tested and time-critical), but most are left in "user-approval" mode. When GPE arrives at a step for which it has responsibility but not authorization, it prompts the user for approval by displaying an "OK?" icon. The GPEView display system propagates these iconic prompts up the procedure hierarchy, so that even when viewing only the highest-level abstract procedures, the user can see which procedures are waiting for his approval or his actions.

Hierarchy — All GPE procedures are captured in hierarchical form, and the system's level of autonomy can be varied at any level of that hierarchy. For example, a high-level procedure for responding to a failed pump might invoke several lower-level procedures to notify affected personnel, activate a backup pump, adjust numerous valves, etc. The responsibility for any of those lower-level procedures may be assigned to a user or to AEGIS, for immediate autonomous execution or pending human approval.

Trusting the User

One significant problem is that the plant procedures are time-dependent: the correct actions to take are highly dependent on when they are taken, and the most suitable plan may change if actions are not executed promptly. So, if a human user takes responsibility for an action, how can we ensure that GPE's other planned actions remain appropriate? It is often desirable to encode pre-authorized monitoring functions that observe the human and plant, attempting to verify that the human is accomplishing the action at an appropriate rate. If not, the system may remind the user, override the user (if pre-authorized to do so), invoke an entirely new plan as a result of the changing plant state, etc.

Informing the User

A key AEGIS design goal is maintaining user awareness. However, 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. Thus seeing the system's full "plan" is impossible. 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. However, this is insufficient from the operator's perspective, because it provides little insight into what the system is planning globally. 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 (Musliner & Krebsbach 1998). 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 are also investigating a more powerful (but costly) approach similar to that used by Durfee *et al.* (1998) to allow true projection in UM-PRS.

Conclusions

This paper discusses adjustable autonomy in the context 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

Durfee, E. H.; Kenny, P. G.; and Kluge, K. C. 1998. Integrated premission planning and execution for unmanned ground vehicles. In *Working Notes of the AAAI Fall Symp. on Distributed Continual Planning.*

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.

Musliner, D. J., and Krebsbach, K. D. 1998. Applying a procedural and reactive approach to abnormal situations in refinery control. In *Proc. Conf on Foundations of Computer-Aided Process Operations (FOCAPO).*