A DISTRIBUTED HIERARCHICAL ARCHITECTURE
OF EXPERT SYSTEMS FOR SUPERVISORY CONTROL
OF MULTIMODULAR NUCLEAR REACTORS

P. J. Otaduy, C. R. Brittain, L. A. Rovere, and N. B. Gove

Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6010

ABSTRACT

A hierarchical supervisory control architecture has been implemented at ORNL to coordinate the controllers of a multimodular nuclear plant. The supervisory controllers form a network of distributed expert systems interfaced with a real-time simulation of the plant, the plant's automatic controllers, and the human operator. The main goal of the supervisory controllers is to maintain the plant operating within safety envelopes while optimizing availability, minimizing stress to components and operators, and facilitating operations. Representative rules implementing strategies for situation dependent reassignment of process goals by embedding diagnostics into the control philosophy are discussed. It should be noted that the control philosophies here described use the ALMR concept for illustration purposes and are not part of the official ALMR design at this time.

I. INTRODUCTION

The development, testing and demonstration of supervisory control concepts at Oak Ridge National Laboratory have been centered around the Advanced Liquid Metal Reactor (ALMR) concept sponsored by the U.S. Department of Energy (DOE). This multimodular ALMR consists of three power blocks, each with three reactors sharing one turbine/generator. The supervisory control architecture implemented (see Fig. 1) mimics that of a fully configured ALMR plant with its nine nuclear reactor cores and three turbine/generator sets.

This paper describes the structure of the expert system architecture, its implementation details, and the communication between the distributed expert systems. The rules in each of the supervisory controller nodes in the hierarchy are also discussed.

II. EXPERT SYSTEM-BASED SUPERVISORY CONTROL ARCHITECTURE

The supervisory control architecture, shown schematically in Fig.1, is of a hierarchical recursive nature. Each node, other than the bottom level actuator nodes, has its own supervisory control module. The overall plant supervisory controller (P-SC) is at the top of the three-level hierarchy. The three power block supervisory controllers (PB-SCs) occupy the middle level, each of them supervising in turn three nuclear steam supply systems (NSSSs) supervisory controllers (N-SCs) and one balance of plant system (BOPS) supervisory controller (BOP-SC). This distributed expert system architecture is fully integrated with ORNL’s advanced controls research facility (ACRF) described in a companion paper.1

The philosophy of operation for each supervisory module is the same: children nodes are viewed as processes to be controlled so that goals specified by the parent node, to which it reports, are achieved. The supervisory modules contain sets of rules that interact with the plant simulator, operator, and automatic controllers and demonstrate the benefits of supervisory control.
control for multimodular systems. Because their rules have access to all data, the supervisory controllers can monitor the performance of plant and controllers and modify operational goals and control laws as needed.

Every supervisory module provides for operator interaction through machine interfaces. The operator is informed of actions being taken by the supervisory module. With regard to control actions, the supervisory modules view the operator as a "manual" optional control generator constrained just like the automatic controllers, to act within operational and safety envelopes set by the parent supervisor. Fig. 1 shows the different levels at which an operator can interact with the plant. Note that, MM-1 and MMI-2 refer to the ACRF's man/machine interfaces described in Ref. 2.

Presently, the five supervisory controllers in a power block are encoded in a combination of Lisp and a modified version of the OPS5 expert system shell, running under LUCID's multiprocessing and Flavors object-oriented extensions.

III. DESCRIPTION OF THE SUPERVISORY CONTROLLERS

The expert system's inference engine is directed towards specific tasks by means of GOAL directives with STATUS, PRIORITY, and LATENCY qualifiers. Rules are triggered by the matching of GOAL and plant state premises. Whenever a goal with status ACTIVE is completed, it is either reactivated or discarded. Reactivation is done by setting the goal status to PENDING. The latency qualifier allows a rule to specify a time lapse for reactivation. A set of goal management rules deals with turning goals from pending to active according to recency and priority.

The P-SC and PB-SCs are normally dormant waiting for messages. A message consists of a string of high level data which may include commands to run the inference engine. When the RUN statement is included in a message, the inference engine takes control until it finishes processing all rules triggered by previously input data patterns.

The module supervisory controllers, N-SCs and BOP-SCs, are normally in a run mode state featuring a perpetually pending goal of WAIT-FOR-DATA. This goal is activated after all other goals have been fulfilled and triggers the input of a new set of plant process data. If no new data are available, the process becomes dormant until a new-data signal is received.

III. A. Module Supervisory Controllers

The NSSS and BOP controllers process plant data asynchronously. Processing a new set of plant data normally involves several inference engine cycles. At the end of each inference cycle, an input stack is checked for supervisory control messages coming from other supervisors. Were messages to be present, they would be processed immediately. After processing plant data and external messages, the computing process goes dormant until a new batch of plant data is obtained from shared memory.

III. A. 1. NSSS Supervisory Controllers (N-SCs)

The N-SCs contain rules to: (1) monitor the approach and actively prevent violation of high and low limits for neutron power, core inlet/outlet temperatures, and steam drum water level; and high core power-to-flow and core power-to-feedwater flow ratios; (2) detect and react...
to degradation of primary loop heat sink, reactor scram, reactor runback, and control rod asymmetry; (3) turn the recirculation pump on/off based on neutron power threshold; (4) monitor control action implementation and diagnose controller failure and automatic controller availability status; and (5) implement operator and PB-SC demands for changes in power and in the assignment of manual, automatic, and heuristic control modes in the NSSS.

The N-SCs have the capability to selectively override the operator and/or the automatic controller when protection system limits (PSLs) are approached. The set of rules implemented to prevent actuation of the protection system rely on the following strategy: if the controller affecting the concerned parameter (1) is in manual or heuristic control mode, and there is a non-failed live automatic controller available, then change the controller to automatic control mode and set its demand to a value inside the allowed operating envelope (normally 5% from the limit); (2) is in automatic control mode and its computing process is alive and performing as demanded, then reset its demand setpoint to a value within the allowed operating envelope; (3) in any other case actuate heuristically the affected control device directly.

Each N-SC communicates only with its PB-SC, its controllers and the MMIs.

III. A.2. BOPS Supervisory Controller (BOP-SC)

The BOP-SC contains rules to: (1) monitor the approach to and actively prevent violation of high and low limits for steam header pressure; (2) diagnose feedwater pump failures; (3) turn one of the two feedwater pumps on or off, based on feedwater flow thresholds; (4) monitor control action implementation and diagnose controller failure and automatic controller availability status; and (5) implement operator and PB-SC demands for power and turbine-lead to reactor-lead control objective changes, and for reacting to step demands generated by the turbine regulator.

The BOP-SCs have the same capabilities as the N-SCs to selectively override the operator and/or the automatic controllers when protection system limits (PSLs) are approached based on the same philosophy.

The BOP-SC communicates only with its PB-SC, its controllers and the MMIs.

III. B. Power Block Supervisory Controller (PB-SC)

The Power Block supervisory rules are primarily related to the allocation of power loads and steam drum inventory setpoints for the NSSS modules in response to both, requests from the P-SC and unscheduled events in the NSSSs and BOPS. The PB-SC keeps track of the status of each NSSS and divides new power requests from the P-SC into up to three power requests, one for each NSSS available for automatic maneuvering.

Because of practical implementation issues related to scarcity of computational resources only one of the three power blocks has been fully simulated in detail down to the simulation of the process dynamics. Two of the power blocks run as "phantoms". These phantom power blocks are Lisp objects which send, receive, and respond to messages from the P-SC in a way that is indistinguishable from the fully implemented power block object.

There are four load allocation strategies available during normal operating conditions: (1) equal load in which the same load demand is sent to each NSSS available; (2) equal change in
which the requested change is divided into as many parts as NSSSs are available so that each available reactor is directed to change power the same amount; (3) extremes first in which the NSSS operating at the lowest/highest power level is the first to be requested to increase/decrease its power during a power-up/down maneuver, until the desired target is reached; and (4) preset values in which the loads are set to predefined values. Note that when the NSSSs available for automatic maneuvering are at the same power level, the first three strategies are equivalent.

During off-normal operating conditions the PB-SC responds to messages generated by (1) the N-SCs as follows:

• on EXTERNAL-SCRAM and FAST-RUNBACK, it reassigns as much of the power generated by the reporting NSSS to the other NSSS modules and the balance is reported to the P-SC for reallocation to the other power blocks.
• on NSSS-SETPOINTS-CHANGED, it reassigns power and changes the BOPS setpoints as needed.
• on LOW-WATER-LEVEL, it forces feedwater redistribution by setting the water level setpoints of the other two NSSSs steam drums to a 5% over their minimum limits.
• on LOW-WATER-LEVEL-RECOVERY, it waits until the total steam and feedwater flows balance, and then, resets the water level setpoints of all of the NSSSs steam drums to their nominal values.

and by (2) the BOP-SC as follows:

• on ABNORMAL-FEEDWATER-HEADER-PRESSURE, it first performs an independent diagnosis by examining the behavior of the three NSSSs feedwater valves and their feedwater flows and responds with the appropriate PB-Confirms or PB-Rejects diagnostics to the original message sender. The rejection indicates to the BOP-SC that its loss-of-feedwater-pump diagnosis was incorrect probably due to a failed feedwater flow or pressure sensor. When the diagnosis is confirmed, the PB-SC stores the present power levels and control setpoints, assesses the mass inventory margins in the three steam drums, computes the rate of power descent for each of the NSSSs to achieve a 50% power level in the power block, and directs the automatic controllers to achieve it.
• on FEEDWATER-PRESSURE-BACK-TO-NORMAL, it optionally either maintains the plant at the present conditions or directs the automatic controls to bring the plant to the conditions at the time the feedwater pump tripped.

III. B. 1. Phantom Power Block Implementation

Each phantom power block maintains and displays information on block power, block power target, allowed power range, rate of change and operating status. When a new target request is received from the P-SC, the phantom simply changes its power level in time steps according to the specified rate, providing no limit is exceeded.

The operator can induce power changes in a phantom block by means of menus thus emulating event-driven reallocations. Events can be triggered at a specified or random time or at a specified power level. When triggered, the P-SC is notified and the block power will be updated at the specified rate to the specified power level. All parameters related to power changes may be changed by the operator by means of pop-up menus. The P-SC in turn will attempt to compensate for the change by requesting complementary changes of power from the other power blocks (one of them being a phantom also).
High-Power-Limit and Check-If-Running safeguards have been implemented in the PB-SC. The High-Power-Limit function ascertains that the power generated by the block does not exceed its upper limit by checking for compliance at fixed time intervals. If the limit is exceeded, a request is made to the reactor and turbine supervisors to reduce their power to the limit. The limit itself is automatically adjusted upon diagnosis of events, such as feedwater pump trip, that affect the modules' generating capacity.

The Check-If-Running function is part of a watchdog-timer mechanism to ascertain that the underlying module supervisory controllers are alive and able to communicate. The four module level supervisory controllers update a time marker each time they perform a routine safety check. The PB-SC checks that these time markers are updated. If any module fails to perform a check during a specified time, currently set at 15 seconds, a message is sent to the operator console.

III. C. Plant Supervisory Controller (P-SC)

The P-SC rules are primarily related to power load allocation to the power blocks. Their nature is mostly administrative and embodies strategic decision making for economic and burnup policies.

Upon receipt of a request for change in power generation from the grid dispatcher, the P-SC generates requests for power variations to each of the three PB-SCs. Any of the PB-SCs can send messages to the P-SC indicating that it is not able to meet fully the requested change in power, or that it can no longer supply the same amount of power it was generating. The P-SC may then request another power block to make up for the deficit.

The P-SC maintains and displays information on the plant's present and target power, allowed power range, rate of change, and operating status. Also, if a requested target is too high, it is saved so that a request may be reissued later if conditions change, e.g., if a power block recovers from limited status to full status.

IV. IMPLEMENTATION INSIGHTS

IV. A. Creation of Multiple Expert System Processes

The expert system processes are part of the class structures defined for the plant, power-block, reactor, and balance-of-plant objects. In these object-classes, the slot named :ops-process holds the pointer to each object's supervisory control process, while the slot :ops-package holds the pointer to the corresponding package. The need to use the package system arises from both, the impossibility to add dynamically new sets of atoms to previously spawned processes in LUCID's multitasking approach, and from the way OPS5 attaches vital inference engine information to those atoms representing the names of the expert system rules.

In OPS5, not only the name of a rule is arbitrary but rules can only be defined after the OPS5 process has been created. In our case, for instance, the rules for each of the three reactor supervisory controllers are identical. Their operation though, is distinct since each of them reacts according to the state of the particular reactor it controls and the private numerical values of the reactor's limiting parameters.

To create multiple non-interfering OPS5 expert systems one must: (1) encapsulate all the special variables intrinsic to the OPS5 code environment; and (2) define a looping function to
access plant process data from within the encapsulated environment. As shown in Appendix A, we chose to define a *Read-Eval-Print-Loop* function to drive the NSSS and BOP expert systems, and an *Evaluate-When-Data-Present* interrupt-driven loop for the power block expert systems. Note that the looping functions are enclosed within a *Throw* construct to allow for unimpeded exit from the loop. A *Throw* with the flag *ATTEND-NEW-DATA* for instance, will cause the immediate evaluation of the :Ops-Input-Stack contents.

**IV. B. Modifications to the OPS5 Expert System Shell**

Modifications made to OPS5 address process control and real-time needs essential to the task at hand. The OPS5 inference engine follows a three step cycle: Recognize (match new incoming data with the pending requirements of the rule set), Conflict Resolution (select the most appropriate rule among those presently satisfied), and Act (implement the action part of the selected rule according to its bindings) for as long as there is a rule satisfied. To allow external communication among the distributed set of OPS5 based expert systems, the Act step was modified to force the execution of the code associated to the atom *OPS-HOOK-FUNCTION* prior to restarting the Recognize step (see Appendix B).

Thus, once in every inference engine cycle, external commands can be processed within the receiving expert system environment and consequently modify the expert system's operation. This structured message passing mechanism together with the *ATTEND-NEW-DATA* mechanism described above (see section IV. A.), allow the supervisory control expert systems to run asynchronously, communicating with each other, the plant, and low level controllers without unwanted interference.

**V. RESULTS**

For illustration purposes we describe the behavior of the supervisory controllers in the case of a loss of one feedwater pump event. When the pump trips the pressure in the feedwater header decreases causing the feedwater valves to open in order to maintain the steam drum water levels. The BOP-SC diagnoses the event by comparing the feedwater pressure-to-flow ratio to the pump characteristic curves, and reports to the PB-SC an *ABNORMAL-FEEDWATER-HEADER-PRESSURE* event with a *1-PUMP-FAILURE* as probable cause. The PB-SC in turn, confirms the diagnosis by independent analysis of NSSSs data and, if needed, directs the NSSSs to reduce their power to the power level one single feedwater pump can support at a rate fast enough to prevent actuation of the steam drums protection system.

Meanwhile, if an NS-SC detects the opening of its NSSS feedwater valves to their maximum it resets its maximum neutron power limit to match the heat removal capacity provided by the evaporation of the feedwater flow entering its steam drum. If the new limit is below the present neutron power level then, unless there is already an appropriate control action in progress, a control action is generated to reduce the neutron power to a level 5% below the limit. This capability to adjust operating limits dynamically protects the NSSSs in case of failure, delay or inadequate action by the PB-SC.

**VI. CONCLUSIONS**

The expert systems described in this work were made functionally independent from one another by developing OPS5 extensions and conveniently using LUCID LISP's multiprocessing facilities and LISP's object-oriented Flavors. From a practical point of view,
nevertheless, they constitute a single and large computer process, which is currently handled by
a single CPU requiring a large amount of physical memory (32 Mb).

The architecture of the expert systems described can be easily recast to operate in a
network of workstations, each being a truly independent process, in order to achieve faster
speeds of execution maximizing their responsiveness. A multi-CPU distributed configuration
will be a better option as the rule-set size continues to increase due to added functionality. The
present architecture has been proven flexible and convenient for rapid prototyping and testing of
supervisory control strategies.

APPENDIX A

IMPLEMENTATION OF MULTIPLE OPSS SHELLS

```
(defun INIT-MULTIPLE-OPSS ()
  (declare (special ;define globals
                   ; The-list-of-global-variables-in-ops5 augmented with the variable
                   *OPS-HOOK-FUNCTION*
               ))) ;set scheduling parameter
  (setf *scheduling-quantum* 1000)

; in a loop, create the OPSS processes and attach them to the corresponding expert system name
(dolist (NAME *list-of-supervisory-controller-names*)
  (send (symbol-value NAME)
    :set-ops-process (make-process
                       :name (string NAME)
                       :function #'ops5-environment-loop
                       :stack-size 15000.
                       :args (list NAME)))) ;end of INIT-MULTIPLE-OPSS

(defun ops5-environment-Loop (MODULE-NAME)
  (let (
        ; The-list-of-global-variables-in-ops5 augmented with the variable
        *OPS-HOOK-FUNCTION*
              ))
  (declare (special
              ; The-list-of-global-variables-in-ops5 augmented with the variable
              *OPS-HOOK-FUNCTION*
          ))
  (setf *OPS-HOOK-FUNCTION* ;define the communication hook function
        #'(lambda ()
          (let ((OPS-IN (send (symbol-value MODULE-NAME)
                                :ops-input-stack)))
            (when OPS-IN
               (send (symbol-value MODULE-NAME)
                     :set-ops-input-stack NIL)
               (catch 'QUIT-OPS
                       (print (eval OPS-IN)))))
          )))
```
If the expert system is to operate in a Read-Eval-Print-Loop and it has its own i/o window named *PORT* then

(catch 'KILL
  (Loop
    (case (catch MODULE-NAME
          (Read-Eval-Print-Loop *PORT*)) ;Runs the OPS Engine until a Throw
      (KILL
        (format T "~~%Terminated by a KILL throw")
        (throw 'KILL T)) ;Starts a new REPLoop ; waiting for input.
      (RESET
        (format T "~~%Reset by a RESET throw") ) ;Starts a new REPLoop ; waiting for input.
      (ABORT
        (format T "~~%Aborted by an ABORT throw") ) ;Evaluates the stack and ; starts a new REPLoop
        ;NOTE: a (RUN) statement at the end of the :input-stack will start rule execution.
      (ATTEND-NEW-DATA
        (eval (send (symbol-value MODULE-NAME) :ops-input-stack)))
    ))
  )
)

;If the expert system is to operate in an interrupt driven mode, then
(let (OPS-IN)
  (catch MODULE-NAME
    (Loop
      (process-wait "Waiting for OPS-input"#
        when-data-available-in-OPS-slot
        MODULE-NAME)
      (setf OPS-IN
        (send (symbol-value MODULE-name) :ops-input-stack))
      (send (symbol-value MODULE-name) :set-ops-input-stack NIL)
      (catch 'QUIT-OPS
        (eval OPS-IN))
    ))
  )
)

;end of opsS-environment-Loop

(defun When-data-available-in-OPS-slot (MODULE-NAME)
  (send (symbol-value MODULE-NAME) :ops-input-stack))
APPENDIX B

MODIFICATIONS TO OPS5'S INFERENCE ENGINE ACT STEP

The following statement was added to the function MAIN:

\[
\text{(when (and (boundp \,\text{"OPS-HOOK-FUNCTION"})
\quad (functionp \,\text{"OPS-HOOK-FUNCTION"}))
\quad (funcall \,\text{"OPS-HOOK-FUNCTION"}))}
\]

at the end of the Act step.

In addition, to maintain compatibility with old code, the following default function was added at lisp toplevel:

\[
\text{\begin{quote}
*\text{OPS-HOOK-FUNCTION*}
\end{quote}}
\end{quote}
\]

\[
\text{#'(lambda () NIL)}
\]

REFERENCES


DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.