A Decision Support System for Maintenance Management of a Boiling-Water Reactor Power Plant[a]

By J. H. Shen,[b] A. Ray,[b] and S. Levine[b]

Abstract: *This article reports the concept and development of a prototype expert system to serve as a decision support tool for maintenance of boiling-water reactor (BWR) nuclear power plants. The code of the expert system makes use of the database derived from the two BWR units operated by the Pennsylvania Power and Light Company in Berwick, Pennsylvania. The operations and maintenance information from a large number of plant equipment and subsystems that must be available for emergency conditions and in the event of an accident is stored in the database of the expert system. The ultimate goal of this decision support tool is to identify the relevant Technical Specifications and management rules for shutting down any one of the plant subsystems or removing a component from service to support maintenance.*

A nuclear power plant must be maintained to satisfy the Technical Specifications and other rules and regulations during all operating phases, including normal and unscheduled operations. Adherence to these requirements leads to handling a large information database involving numerous plant components. Identification of the appropriate component(s) to be maintained is usually time-consuming and is also vulnerable to human errors and omissions. Often the problem is to determine whether taking a plant component out of service for repair or maintenance or reinstating it into service will cause a change in the status of the subsystem to which it belongs. To this effect, an expert system[1,2] is suitable for decision support in maintenance management where the knowledge of experienced operators needs to be assimilated for implementation within the technical information database. Although the specific objective of this expert system code is to support the decisions regarding which components can be safely and reliably removed from the plant operations for maintenance, it may also serve as a query-answering facility for checking the plant system status as well as for training purposes.

This article presents the concept and development of a prototype expert system to serve as a decision support tool for maintenance of boiling-water reactor (BWR) nuclear power plants.[3] The code makes use of the database derived from the two BWR units operated by the Pennsylvania Power and Light Company (PP&L) in Berwick, Pennsylvania. The operating and maintenance information from a large number of equipment and subsystems, which must be available for emergency conditions and in the event of an accident, is stored in the database of the expert system. The database must contain the relevant Technical Specifications and management rules for shutting down any one of the systems or removing a component from service to support maintenance. Because of the

[a]This work was supported in part under a research grant from Pennsylvania Power and Light Co., Berwick, Pennsylvania.

[b]Pennsylvania State University, University Park, Pennsylvania 16802.

complexity and time needed to incorporate a large number of systems and their components, the expert system has been developed in two phases. In the first phase, the prototype code includes only the reactor core isolation coolant (RCIC) system, the high-pressure core injection (HPCI) system, the instrument air system (IAS), the service water system (SWS), and the plant electrical system. The second phase is targeted to include all other systems. This article summarizes the prototype code and the design concept of the complete expert system code for maintenance management of the plant subsystems and components.

PERTINENT DEFINITIONS

The expert system, serving as a decision support tool, must record any changes in the current status of each subsystem and its relationship with others and, at the same time, ensure that the decisions not violate the Technical Specifications and regulations.

For the construction of the expert system in view of the maintenance schedule in BWR power plants, the subsystems are classified in two categories:

- **Type 1:** Subsystems for major operations such as RCIC and HPCI, whose status is either operative (denoted as *op*) or inoperative (denoted as *in_op*)
- **Type 2:** Subsystems for support services such as IAS and SWS, whose status may be full capacity (denoted as *full_cap*) or reduced capacity (denoted as *red_cap*)

The set S denotes the complete plant, the subset A represents the set of all subsystems belonging to type 1, and the subset B represents the set of all subsystems belonging to type 2. Therefore $S = A \cup B$ and $A \cap B = \emptyset$, which implies that the subsets A and B are exhaustive and mutually exclusive. The sets A and B can be further classified into subsets on the basis of their status as follows. The subsets A_1 and A_2 of A are defined as

$$A_1 = \{\alpha \in A \mid \text{status of } \alpha = op\}$$
$$A_2 = \{\alpha \in A \mid \text{status of } \alpha = in\_op\} \quad (1)$$

and subsets B_1 and B_2 of B are defined as

$$B_1 = \{\alpha \in B \mid \text{status of } \alpha = full\_cap\}$$
$$B_2 = \{\alpha \in B \mid \text{status of } \alpha = red\_cap\} \quad (2)$$

The status of each of the preceding subsystems is determined in terms of its *cut sets* defined as follows:

Definition 1: A cut set of a given plant subsystem is defined as a set of its components that will degrade the status of the subsystem (of type 1 or type 2) if each of these components is out of operation. In that case, the cut set is said to be reached.

Remark 1: The notion of a cut set introduced in this article is not identical to the definition of *cut set* or *minimal cut set* used in risk management (e.g., fault tree and event tree).[4] Following Definition 1, a cut set consists of one or more components, each of which performs specific functions for the subsystem. Consequently a subsystem is partitioned by all its cut sets; that is, if C_i, $i = 1, 2,...,n$, are the cut sets of a subsystem S, then

$$\bigcup_{i=1}^{n} C_i = S; \quad C_i \cap C_j = \emptyset; i \neq j; \quad i, j = 1,...,n \quad (3)$$

A change in the system status may alter the relationship of this system with others. These relationships are defined below.

Definition 2: If a change in the status of a subsystem α restricts any further change in the status of another subsystem β, then α and β are defined to be *interacting* systems relative to each other; this relation is denoted as interact_sys(α,β).

Definition 3: Let α and β be subsystems of type 1 such that they are interacting systems relative to each other. Then β is defined to be *high risk* relative to α if the status of β has to be *op* for the plant to remain in operation whenever the status of α is *in_op* and vice versa. If γ is a subsystem of type 2, then γ is defined to be high risk if its status is *red_cap*.

Remark 2: In the database, the high-risk list of each subsystem is formed in a list structure attached to that particular subsystem and is expressed as follows:

If $\alpha \in A_2$, $\beta \in A_1$ [that is, α is *in_op* and β is *op*, and interact_sys(α,β) holds], then β belongs to the high-risk list of α.

If $\alpha \in B_2$ (that is, α is *red_cap*), then α is in its own high-risk list.

For example, given the subsystem status that RCIC is *op*, HPCI is *in_op*, and interact_sys (RCIC, HPCI), then HPCI is in the high-risk list of RCIC.

Definition 4: Let α be a subsystem of type 1, β be a subsystem of type 2, and the relationship interact_sys(α,β) hold. Then β is defined to be contingent relative to α if β is *full_cap* and α is *in_op*.

Similarly, α is defined to be contingent relative to β if the status of β is *red_cap* and that of α is *op*.

Remark 3: In the database, the contingency list of each subsystem is formed in another list structure attached to that particular subsystem. Therefore the contingency list can be expressed in terms of sets as follows:

If $\alpha \in A_2$, $\beta \in B_1$, and interact_sys(α,β), then β is in the contingency list of α.

If $\alpha \in A_1$, $\beta \in B_2$, and interact_sys(α,β), then α is in the contingency list of β.

For example, given the subsystem status that RCIC is *in_op*, IAS and SWS are both *red_cap*, and interact_sys (RCIC, IAS), interact_sys (RCIC, SWS), then IAS and SWS are both in the contingency list of RCIC.

ARCHITECTURE OF THE EXPERT SYSTEM

The expert system code is structured with three major interacting modules: (1) database, (2) rulebase, and (3) user interface. The computer language PROLOG was chosen to program the code because of the following features: (1) backtracking for searching databases, (2) use of factual data to modify the system status, and (3) generation of relevant technical information.[5] The expert system is operated at two different hierarchical levels: component level and system level. The set of rules at each level is executed with the use of the individual database. The overall structure of the expert system in Fig. 1 illustrates interactions between the rulebases, databases, and the user interface. The arrows in the figure indicate the direction of information flow of the program.

The major decisions at the component level include the determination of whether a subsystem will change its status if a specific component is taken out of operation. Obviously, this depends on the operational status of other components of the subsystem. At the system level, the rules are used to find the relationships of a subsystem with others. These operations depend only on the data related with the operating status of each subsystem. Accordingly, the databases are set up separately into two parts—component database and system database. In addition to the system-level and component-level modules, there is a separated module for the electrical system because it has some special properties that are unique from other subsystems. Both the database and rulebase of electric components are accessible to system-level modules and component-level modules through an electric-rule module because a change of the electrical system may affect all other major plant subsystems and their components. The system database is accessible to the system rulebase, electric component rulebase, and the user interface. Following the execution of component rules, the system rules are activated to decide upon the new relations between the individual subsystems in the event of any change in the status of their components. The details of the operation of each module are introduced in the following sections.

Two Levels of Databases

Component Database. This database is built from two different tables for one particular subsystem as shown in Figs. 2 and 3. In the component table of Fig. 2, the pertinent information of each component is first represented as a tuple:[6]

component(system_number, cut_number,
comp_name, description, ...)

The second tuple in the component table is the information of cut sets stored in a list structure:

cut_set(system_number, cut_number,
[comp_name1, comp_name2, comp_name3, ...])

In the maintenance table of Fig. 3, the information about the maintained components is recorded as two different tuples:

in_repair(system_number, cut_set,
[comp_name1, comp_name2, ...])

repair_comp(system_number, comp_name,
maintained_type, started_date)

A component name is a key factor for database operations in each table, which is labeled by a unique system number to enhance the efficiency of the search procedure within the database of each subsystem.

System Database. The pertinent data at the subsystem level are structured as tuples, and all such data are contained in one table. The subsystem information is expressed as a collection of tuples; for example, the first tuple in the system database is as follows:

system(system_number, system_name,
system_type, system_status, description)

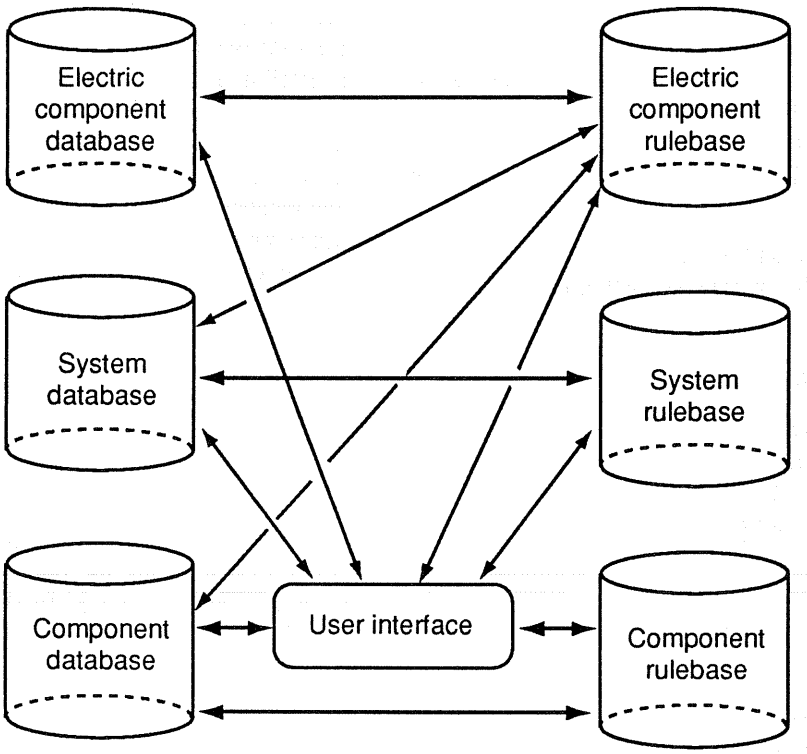

Fig. 1 Overall structure of the expert system.

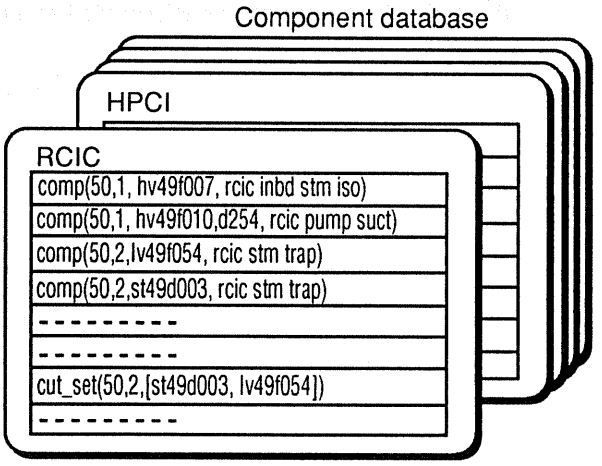

Fig. 2 Component database. [The pertinent information of each component is represented by tuples: component (system_number, cut_number, comp_name, description, ...) and cut_set (system_number, cut_number [comp_name1, comp_name2, ...]).]

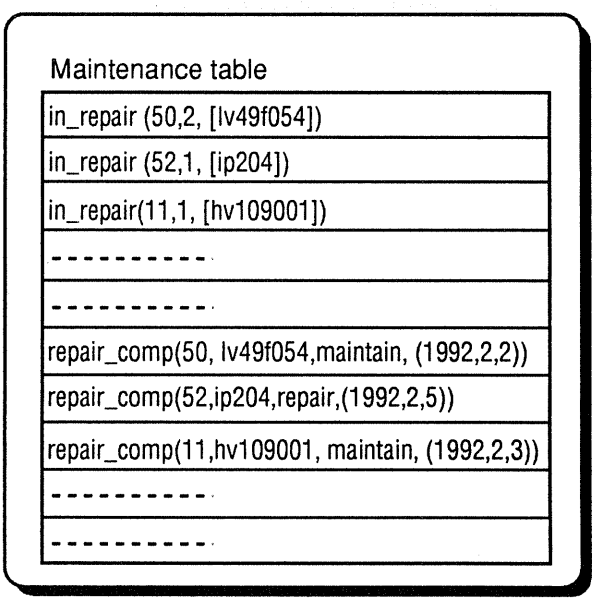

Fig. 3 Maintenance table in component database. [Information about the maintained components is recorded as two different tuples: in_repair (system_number, cut_set, [comp_name1, comp_name2, ...]) and repair_comp (system_number, comp_name, maintained_type, started_date).]

and the next two tuples represent relationships between the subsystems as follows:

high_risk_sys(system_number, system_type, [system_number1, system_number2, ...])

contige_sys(system_number, system_category, [system_number1, system_number2])

A list of system numbers in the preceding example is entered in highrisk_sys() and contingency_sys() because a given subsystem could simultaneously belong to the categories of high risk and contingency relative to another subsystem.

Similar to the formation of the databases, the rulebases are structured into two separate groups for the component level and the system level.

Two Levels of Rulebases

The rulebases for the expert system are structured in two levels—component level and system level—as delineated below.

Component-Level Rulebase. The rulebase at the component level consists of several major rules. If a component is scheduled for maintenance or reinstated into operation, then the expert system determines whether there is a change in the system status. The rules are stated as follows:

Let S be a subsystem and C_i be the cut sets of S, $i = 1, ..., n$, as set forth in Definition 1.

Rule 1 [Taking the component(s) out of operation]: Let A be the set of components currently under maintenance or repair and B be the set of components to be maintained or repaired, implying that $A \cap B = \varnothing$. Partitioning of A and B into subsets of the cut sets C_i yields

$$\bigcup_{i-1}^{n} A_i = A, A_i \subset C_i;\ \bigcup_{i-1}^{n} B_i = B, B_i \subset C_i; \quad \text{and } A_i \cap B_i = \varnothing\ i = 1,...,n \qquad (4)$$

where some of A_i's and B_i's could be empty sets.

The status of S does not change if either one of the following two conditions holds:

$$\exists i \in \{1,...,n\} \text{ such that } A_i = C_i \text{ and } B_i = \varnothing \qquad (5)$$

or

$$A_i \neq C_i, \text{ and } A_i \cup B_i \neq C_i, \forall i \in \{1,...,n\} \qquad (6)$$

else the status of S changes.

Rule 2 [Putting the component(s) back into operation]: Let D be the set of components to be put back into operation after repair or maintenance and E be the set of components to remain in repair or maintenance, implying that $D \cap E = \varnothing$. Partitioning D and E into subsets of C_i yields

$$\bigcup_{i-1}^{n} D_i = D,\ D_i \subset C_i,\ \bigcup_{i-1}^{n} E_i = E, E_i \subset C_i, \quad \text{and } D_i \cap E_i = \varnothing \quad i = 1,...,n \qquad (7)$$

where some of D_i's and E_i's could be empty sets.

The status of S does not change if either one of the following two conditions holds:

$$D_i \cup E_i \neq C_i; \quad \forall i \in \{1,...,n\} \qquad (8)$$

or

$$\exists i \in \{1,...,n\} \text{ such that } D_i = \varnothing \text{ and } E_i = C_i \qquad (9)$$

else the status of S changes.

Example of Rule 1: Let the subsystem S be of type 1 and be partitioned into three cut sets: C_1 = {cut = 1, hv001}; C_2 = {cut = 2, hv002, hv003}; and C_3 = {cut = 3, fh001, fh002}. Let $A = \varnothing$; A_2 = {in_repair (2, hv002, hv003)}; A_3 = {in_repair(2, fh001)} be currently in the repair or maintenance list. For the cut set, $A_2 = C_2$, the current status of S is *in_op*, and it follows from condition 4 of Rule 1 that $B_2 = \varnothing$. Therefore the condition 5 of Rule 1 is satisfied, and the status of S will remain unchanged no matter how many and what other components are going to be put into repair or maintenance.

Example of Rule 2: Let the subsystem S and its cut sets, described above in the example of Rule 1, have the repair/maintenance list as follows: in_repair (2, hv002, hv003) and in_repair(2, fh001). Since the cut set C_2 is reached, the status of S is *in_op*. If the objective is to put the component fh001 back into operation, then following Rule 2 the sets D and E may have to be partitioned as follows: $D_1 = \varnothing$; $D_2 = \varnothing$; D_3 = {in_repair(2, fh001)}; and $E_1 = \varnothing$; E_2 ={in_repair (2, hv002, hv003)}; $E_3 = \varnothing$. Since $D_2 = \varnothing$ and $E_2 = C_2$, condition 7 of Rule 2 is satisfied, and the status of S will remain unchanged.

System-Level Rulebase. The system-level rulebase determines whether a change in the status of a subsystem will change the high-risk and/or contingency

lists of its own and other interacting systems. The rules are stated as follows:

Let subsets A_1, A_2, B_1, and B_2 be as defined in Eqs. 1 and 2.

High-risk system rules: Following Remark 2, the two conditions for setting up a high-risk list of a subsystem are (1) if $\alpha \in A_2$, $\beta \in A_1$, and interact_sys(α,β), then β is the high-risk list of α; and (2) if $\alpha \in B_2$, then α is in its own high-risk list. The rulebase for updating the high-risk system list for a subsystem is stated below.

Rule 1: If α is moved from A_1 to A_2 and if

$\beta \in A_1$ and interact_sys (α,β), then add β to the high-risk list of α (10)

$\beta \in A_2$ and interact_sys (α,β), then delete α from the high-risk list of β (11)

Rule 2: If α is moved from A_2 to A_1 and if

$\beta \in A_1$ and interact_sys (α,β), then delete β from the high-risk list of α (12)

$\beta \in A_2$ and interact_sys (α,β), then add α to the high-risk list of β (13)

Rule 3: If $\alpha \in A_2$ and $\beta \in A_2$, $\forall$ interact_sys(α,β), then shut down the plant (14)

Rule 4: If α is moved from B_1 to B_2, then add α to its own high-risk list (15)

Rule 5: If α is moved from B_2 to B_1, then delete α from its own high-risk list (16)

Examples of high-risk system rules: For two subsystems of type 1, say RCIC and HPCI, with interact_sys(RCIC, HPCI), if the status of RCIC is *in_op* and that of HPCI is *op*, then, according to Definition 3, HPCI is in the high-risk list of RCIC. There are two possible changes in the status of these two subsystems. First, if RCIC changes its status from *in_op* to *op*, which satisfies condition 12 of Rule 2, then HPCI should be deleted from the high-risk list of RCIC. Second, if HPCI changes its status from *op* to *in_op*, which satisfies condition 11 of Rule 1, then HPCI should be deleted from the high-risk list of RCIC; however, if RCIC and HPCI are the only type 1 subsystems interacting with each other, then, according to Rule 3, the plant should be shut down.

Contingency system rules: Following Remark 3, there are two conditions for setting up the contingency list of a subsystem: (1) if $\alpha \in A_2$, $\beta \in B_1$, and interact_sys(α,β), then β is in the contingency list of α; and (2) if $\alpha \in A_1$, $\beta \in B_2$, and interact_sys(α,β), then α is in the contingency list of β. The rulebase for updating the contingency lists for each subsystem is stated below.

Rule 1: If α is moved from A_1 to A_2, and if

$\beta \in B_1$ and interact_sys(α,β), then add β to the contingency list of α (17)

$\beta \in B_2$ and interact_sys(α,β), then delete α from the contingency list of β (18)

Rule 2: If α is moved from A_2 to A_1, and if

$\beta \in B_2$ and interact_sys(α,β), then add α to the contingency list of β (19)

$\beta \in B_1$ and interact_sys(α,β), then delete β to the contingency list of α (20)

Rule 3: If β is moved from B_1 to B_2, and if

$\alpha \in A_1$ and interact_sys(α,β), then add α to the contingency list of β (21)

$\alpha \in A_2$ and interact_sys(α,β), then delete β from the contingency list of α (22)

Rule 4: If β is moved from B_2 to B_1, and if

$\alpha \in A_2$ and interact_sys(α,β), then add β to the contingency list of α (23)

$\alpha \in A_1$ and interact_sys(α,β), then delete α from the contingency list of β (24)

Examples of contingency system rules: Let the status of two type 1 subsystems, RCIC and HPCI, be

in_op and *op*, respectively. Let the status of another two type 2 subsystems, IAS and SWS, be *full_cap* and *red_cap*, respectively. If the preceding four subsystems are interacting with each other, then IAS is in the contingency list of RCIC and HPCI is in the contingency list of SWS according to Definition 4. If IAS changes its status from *full_cap* to *red_cap*, which satisfies conditions 21 and 22 of Rule 3, then HPCI should be added in the contingency list of IAS, and IAS itself should be deleted from the contingency list of RCIC. On the other hand, if SWS changes its status from *red_cap* to *full_cap*, which satisfies conditions 23 and 24 of Rule 4, then SWS should be added to the contingency list of RCIC and HPCI should be deleted from the contingency list of SWS.

The process of entering and deleting a component into the maintenance table is illustrated in the program flowcharts in the appendix.

The Electrical System

Database of the Electrical System. The electrical system has a unique property in the sense that the loss of power supply affects every plant component and that the individual subsystems could be electrically interconnected even if they are mechanically isolated. On the basis of this consideration, the electrical components are expressed in a tree structure. This can be easily constructed and operated in PROLOG language and PDC-PROLOG compiler environment.[5] All the electrical components and the components of other subsystems that are electrically connected are constructed in one tree. Each branch of the tree represents a closed loop. Each node of the data structure represents one electrical component where the components of higher voltage are ancestors of that of lower voltage. The last level of the tree (i.e., the leaves of the tree) represents the components of other subsystems connected to the electrical circuits. An example of this tree structure is shown in Fig. 4. By doing so, not only do the operations of component search and execution of the electrical system rules become faster but also user friendliness and visual display are enhanced.

Because a disabled component may affect the operating status of the subsystem to which it belongs and also other subsystems, the components in the repair/maintenance table are then expressed by two lists. The first list contains all the actions that have been taken on the specific component. The second list contains the time corresponding to each action as shown below.

repaired_comp(50, hvf007, [repaired/maintained, disabled], [(28,4,1990), (1,5,1990)])

Any one of the preceding states may cause the subsystem to change its status. If the component is only disabled, however, it could be further scheduled for repair or maintenance.

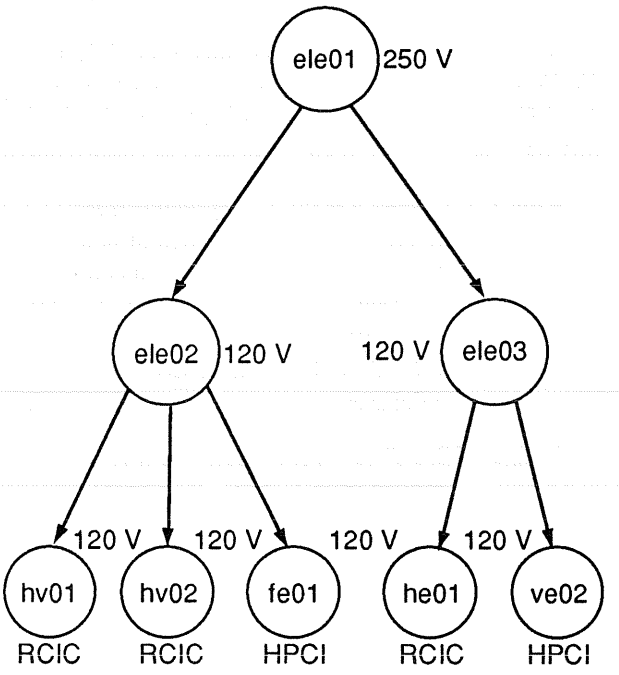

Fig. 4 An example of electric-component tree structure. (hv01, hv02, fe01, he01, and ve02 represent components in the RCIC or HPCI system. ele01, ele02, and ele03 represent electrical components.)

Rulebase of the Electrical System. If an electrical component is scheduled for maintenance or repair, the subsystems that are served by the circuit of this electrical component may need to be disabled. This fact will have a tendency to make the subsystem change its status, depending on whether the disabled components form a cut set. Therefore additional rules need to be executed for handling electrical systems.

Two groups of separated rules are executed in the electrical system rulebase. The first group works on the change of the component and subsystem status that are all related to the action of an electrical component

scheduled for repair or maintenance. The second group modifies the rulebase for other subsystems in the event of interactions with the electrical system. The rulebase of the electrical system is described as follows.

Rules for changing the operating conditions of electrical components: Let all components in the electrical system be expressed by the set F, which is further partitioned into the subsets F_1 and F_2 on the basis of the operating conditions of the components.

$$F_1 = \{\alpha \in F \mid \alpha \text{ is in operation}\};$$
$$F_2 = \{\alpha \in F \mid \alpha \text{ is in repair/maintenance}\} \quad (25)$$

Let $\alpha \in F$; $\beta_1, \beta_2, \ldots \beta_k$ be the ancestors of α, and let $\gamma_1, \gamma_2, \ldots \gamma_{m_1}, \gamma_{m_1+1}, \gamma_{m_1+2}, \ldots \gamma_{m_2}, \ldots \gamma_{m_s}$ be the leaves of α. Let $G_1, G_2, \ldots G_s$ be the subsystems such that $\{\gamma_1, \gamma_2, \ldots \gamma_{m_1}\} \subset G_1, \ldots, \{\gamma_{m_{s-1}+1}, \gamma_{m_{s-1}+2}, \ldots \gamma_{m_s}\} \subset G_s$.

Rule 1: If $\beta_i \notin F_2$, $i = \{1, 2, \ldots k\}$, and if α is moved from F_1 to F_2, then disable γ_j, $j = \{1, 2, \ldots m_1, \ldots, m_s\}$ and change subsystem status of G_t, $t = \{1, 2, \ldots, s\}$ and their relations according to the rules described in the section on rulebases (26)

Rule 2: If $\beta_i \notin F_2$, $i = \{1, 2, \ldots k\}$, and if α is moved from F_2 to F_1, then enable γ_j, $j = \{1, 2, \ldots m_1, \ldots, m_s\}$ and change subsystem status of G_t, $t = \{1, 2, \ldots, s\}$ and their relations according to the rules described in the section on rulebases (27)

Example of Rule 1: Considering the electric component tree given in Fig. 4, if ele02 is taken out for repair and if ele01 is in operation, then each of the components hv01, hv02, and fe01 should be disabled. Because hv01 and hv02 belong to RCIC and fe01 belongs to HPCI, the rules described in the section on rulebases continue to be used to change the status of RCIC and HPCI along with the high-risk and contingency lists as applicable.

Rules for maintenance of other subsystems: Let the components of a given subsystem be expressed by the set S, which is further partitioned into subsets S_1 and S_2 on the basis of the operating conditions of their components.

$$S_1 = \{\alpha \in S \mid \alpha \text{ is in operation}\}$$
$$S_2 = \{\alpha \in S \mid \alpha \text{ is in repair/maintenance}\} \quad (28)$$

Because the disconnection of an electrical circuit may disable some of the components in S, the following rules should be executed prior to execution of the rules described in the section on rulebases.

Rule 1: If α is scheduled to move from S_1 to S_2, then check if α is in S_2, which could have three possible answers:

No, α is not in S_2; continue to execute the rules described in the section on rulebases (29)

Yes, α is in S_2 but only disabled; schedule α for repair/maintenance without executing the rules described in the section on rulebases (30)

Yes, α is in S_2 and under repair or maintenance; stop scheduling of α (31)

Rule 2: If α is scheduled to move from S_2 to S_1, then check if α is in S_2, which could have four possible answers:

No, α is not in S_2; stop scheduling of α (32)

Yes, α is in S_2 but only disabled; stop the process (33)

Yes, α is in S_2 but not disabled; execute the rules described in the section on rulebases (34)

Yes, α is in S_2 but disabled and under repair or maintenance; take off the repair or maintenance status of α without executing the rules described in the section on rulebases (35)

Examples of Rule 1: In the last example shown in Fig. 4, the components hv01, hv02, and fe01 are disabled because of the repair status of ele02. The components he01 and ve01 are still operational. Let hv01 be under repair also, then the following items are found in the repair/maintenance table: (1) repaired_comp(50, hv01, [repaired, disabled], [(28,4,1990), (1,5,1990)]) and (2) repaired_comp(50, hv02, [disabled], [(1,5,1990)]). Now a component of RCIC is scheduled for repair. First, if the component he01 is scheduled, the process

can be carried out according to the rules described in the section on rulebases following instruction 29 of Rule 1. Second, if the component hv02 is scheduled, instruction 30 of Rule 1 should be followed. Finally, if the component hv01 is scheduled, the program should stop and generate an error message following instruction 31 of Rule 1.

The User Interface

For the enhancement of user friendliness, the expert system program is designed to be menu-driven following a tree structure. The main menu, illustrated in Fig. 5, has seven selections to facilitate determination of plant or system status, maintenance of components, entering components back into operation, and disabling or activating electrical components.

A complete expert system should have the property that the user is able to access the database. For this purpose, an additional module was designed and built. First, it is accessible only by some special operators with the correct password to protect the database. Second, to save the computer space, external database structure was used,[6] which also increases the speed by using B-tree structure in searching and operation. The module is written such that the user could open a particular database on the basis of its name, searching for a specific element and replacing it, or creating an entire new database.

Summary

A prototype expert system has been developed for decision support of maintenance management in BWR power plants. The prototype expert system is coded in PROLOG, and its feasibility and efficacy are demonstrated for maintenance of two major operating subsystems and several support subsystems at one of the two nuclear power generating units in Berwick, Pennsylvania, operated by PP&L.

The prototype code is being expanded to include all the pertinent subsystems and incorporate PP&L's related computer program and management techniques. In the future, a more friendly user interface will be developed by using the PROLOG tool provided to access PP&L's programs and increase the program flexibility and reliability.

The program is being developed under the supervision of PP&L plant and engineering personnel to ensure that the terminology and operational characteristics of the expert system are consistent with those used by the plant operating personnel.

ACKNOWLEDGMENT

The authors wish to thank Prof. Robert M. Edwards of Pennsylvania State University for constructive suggestions as well as for editing the manuscript.

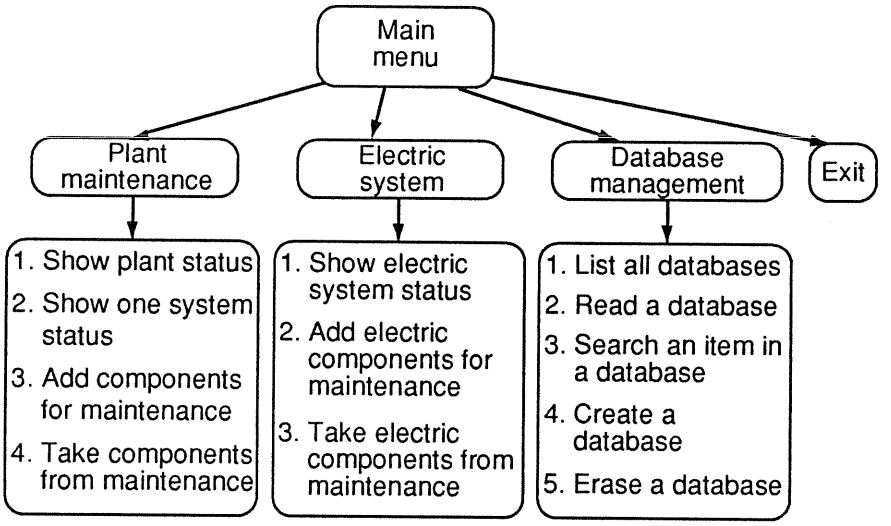

Fig. 5 Structure of the user interface.

REFERENCES

1. J. A. Bernard and T. Washio, *Expert Systems Applications Within the Nuclear Industry*, American Nuclear Society, LaGrange Park, Ill., 1989.
2. D. Merritt, *Building Expert Systems in PROLOG*, Springer-Verlag, New York, 1989.
3. *General Description of a Boiling Water Reactor*, Nuclear Energy Division, General Electric Company, San Jose, Calif., 1974.
4. A. Hoyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods*, p. 88, Wiley-Interscience, New York, 1994.
5. *PDC-PROLOG User's Guide*, Version 3.21, Borland International, Denmark, 1988.
6. C. J. Date, *Database Systems*, Vol. 1, Addison-Wesley, Reading, Mass., 1986.

APPENDIX—FLOWCHARTS OF THE MAINTENANCE PROGRAM

The following two flowcharts describe the logic for scheduling the components for maintenance and repair.

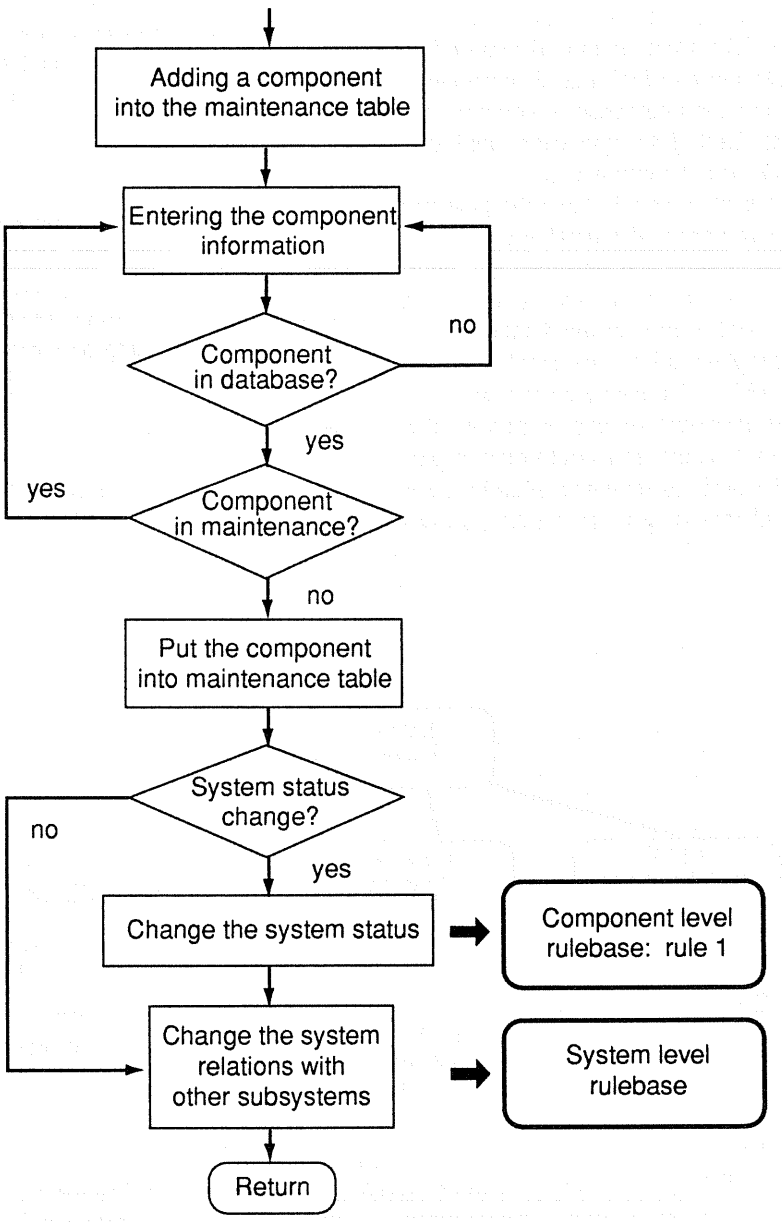

Fig. A.1 Flowchart for entering a component to the maintenance table.

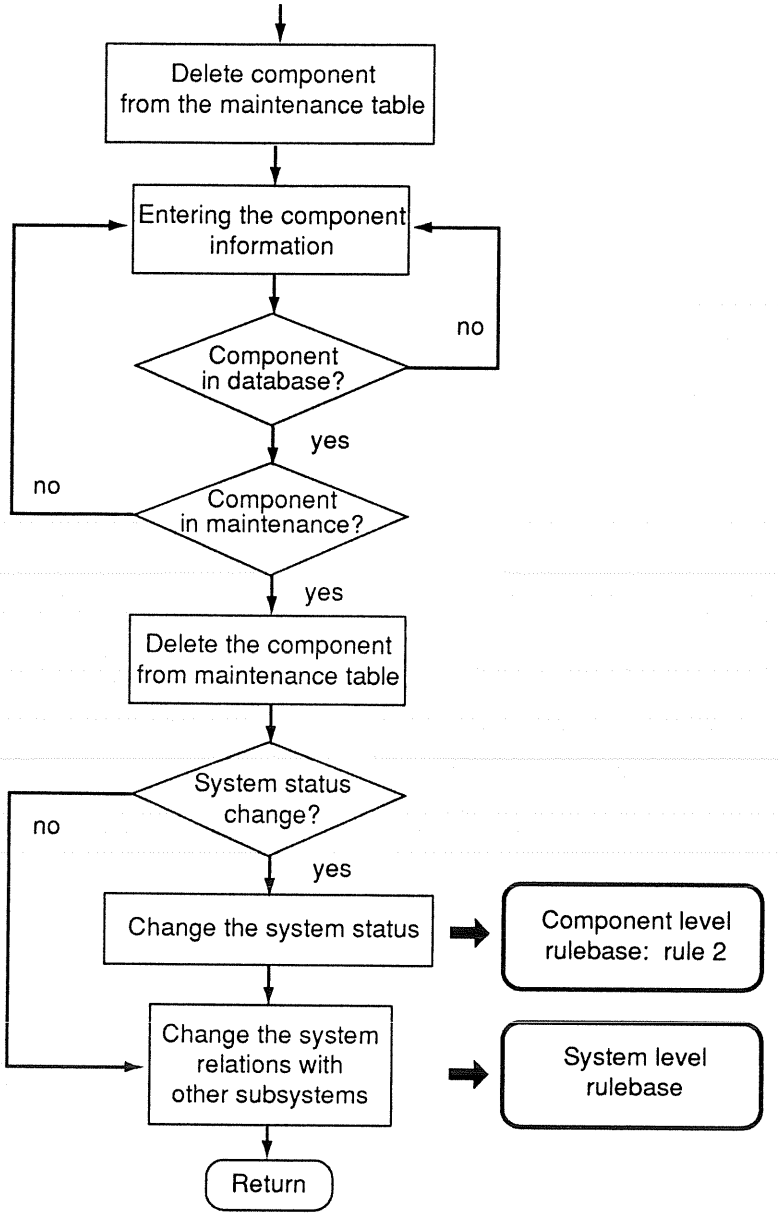

Fig. A.2 Flowchart for deleting a component from the maintenance table.