A Decision Support System for Real-Time Monitoring and Control of Dynamical Processes*

Steven R. Nann† and Asok Ray
Mechanical Engineering Department, The Pennsylvania State University, University Park, Pennsylvania 16802

Soundar Kumara
Industrial and Systems Engineering Department, The Pennsylvania State University, University Park, Pennsylvania 16802

This article presents the concept and development of a prototype diagnostic decision support system for real-time control and monitoring of dynamical processes. This decision support system, known as Diagnostic Evaluation and Corrective Action (DECA), employs qualitative reasoning, in conjunction with quantitative models, for monitoring and diagnosis of malfunctions in dynamical processes under routine operations and emergency situations. DECA is especially suited for application to time-constrained environments where an immediate action is needed to avoid catastrophic failure(s). DECA is written in common Lisp and has been implemented on a Symbolics 3670 machine; its efficacy has been verified using the data from the Three Mile Island No. 2 Nuclear Reactor Accident.

I. INTRODUCTION

Decision support systems for on-line monitoring and control of complex and large-scale dynamical processes require timely handling of data and execution of diverse functions. In general, the quantitative approach to decision making relies on analytical techniques, such as dynamic programming, hypotheses testing, and optimal filtering, to find an optimal or a sub-optimal solution. However, as the complexity of the plant increases, the number of plant vari-

*This work was supported in part by the Space and Naval Warfare Systems Command under a research contract to the Applied Research Laboratory of the Pennsylvania State University.

†Currently with E.I. Du Pont De Nemours, Wilmington, DE 19880–0840.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 6, 739–758 (1991)

CCC 0884-8173/91/070739-20$04.00

ables and the program execution time increase exponentially. This often renders the use of quantitative techniques unpractical for real-time decision-making. On the other hand, qualitative procedures using knowledge-based techniques have been successfully used for off-line decision making in diverse disciplines[1–4] but their applications to real-time operations have been rather limited due to the constraints of knowledge-base size and complexity of interactions within the currently available computational power. Examples of the existing knowledge-based systems in the real-time control domain are ECESIS,[5] PICON[6], and FALCON.[7] Whereas ECESIS and PICON are designed for the Space Station, FALCON serves as an automated fault detector and analyzer for chemical processes.

The role of a human operator in real-time decision support systems of large-scale strategic processes, such as advanced aircraft, spacecraft, and nuclear power plants, can be envisioned as a supervisor assisted by a set of heterogeneous instruments. The major tasks of the operator are monitoring of the processes under automatic control and intervention in the event of catastrophic failures and emergency situations. The human operator's role can be well-defined under the normal operation but could drastically change in the emergency environment when a myriad of unforeseen and hard-to-diagnose events may occur. The human operator, while faced with abundant data and severe time constraints under these circumstances, is required to plan strategies, make decisions, and take actions. There is a need for a decision support system that will systematically prioritize the available information for the human operator in real time as well as assist him/her in making timely and accurate decisions. The research reported here is aimed toward fulfilling this need.

A. Theme of the Research

Timely detection of a malfunction is the first step in real-time control and decision making. The second step is to locate the faults and then prioritize them in the order of their severity. At the final step, a decision is made for appropriate corrective action(s). A procedure encompassing the above three major steps is usually specific to a problem. However, the architecture of a decision support system should be general enough for adaptation to similar domains of problems. These concepts are brought forward to the development and implementation of a decision support system with the following features:

- Development of a framework for on-line diagnosis of malfunctions, that is, disruptions, abnormal operations, and failures of plant components;
- Incorporation of qualitative reasoning into the above framework in conjunction with quantitative models.
- Providing support to human operator(s).
- Ensuring portability of the system software.

B. Development and Implementation of the Decision Support System

Jow[8] used a semi-empirical quantitative algorithm, which is based on the principle of local optimization and is much less complex than dynamic pro-

gramming for on-line monitoring, information prioritization, and decision making. The decision support system, proposed in this article, is built upon the general concept of Jow's work, and is referred to as Diagnostic Evaluation and Corrective Action (DECA). However, instead of being restricted to an algorithmic approach, DECA makes use of qualitative reasoning. The architecture of DECA is built upon the fundamental notions of Milne's theory of diagnosis.[9] A major objective of DECA is to support human operators in the decision making process.

DECA is designed to implement a decision making strategy for dynamical processes during routine operations and time-constrained, emergency situations. DECA has been developed as a general-purpose shell which is versatile and sufficiently autonomous in the sense that it would be capable of handling the computer operational details and execute the processes in real time while the human user would concentrate on setting up the knowledge base for the particular application at hand. DECA has a hierarchical decision structure such that it functions less autonomously at higher levels.

The general framework of DECA and its efficacy have been tested using data of the Three Mile Island No. 2 (TMI-2) nuclear power plant accident. (This particular example was chosen due to the availability of relevant data.[10,11]) DECA has been implemented on a Symbolics 3670 machine using common LISP in conjunction with Flavors,[12] and can be interfaced with real-time simulators or with the plant under control.

C. Organization of the Article

The article is oganized in four sections including the introduction and two appendices. Section II describes the DECA architecture in detail. Implementation of DECA on a Symbolics 3670 machine is described, and the results of the test runs that utilize the actual data from the TMI-2 accident are presented and discussed in Section III. Summary, conclusions, and goals of future research are given in Section IV. Milne's theory of diagnosis[9] which provides the notion of qualitative reasoning in DECA is succinctly described in Appendix A. Appendix B briefly describes the accident of TMI-2 which is the plant under consideration for elucidating the architecture of DECA.

II. ARCHITECTURE OF DIAGNOSTIC EVALUATION AND CORRECTIVE ACTION (DECA)

The architecture in DECA is formulated to meet the objectives stated in Section I-A. Specifically the goal of DECA is as follows:

- Simultaneous monitoring of multiple plant variables in real time;
- Feature extraction, that is, ability to identify relevant data from extraneous information;
- Diagnoses of malfunctions; and
- Prioritization of relevant information, thus drawing the operators' attention to the part of the plant where the problem emanates from.

A. Overview of DECA

The architecture of DECA is formed on the basis of Milne's theory of diagnosis[9] which consists of four interconnected levels: (i) Structural; (ii) Behavioral; (iii) Functional; and (iv) Pattern Matching. The functions of these four levels are described in Appendix A.

The concepts of the first three of the above four levels of Milne have been used in DECA, and their correlation is as follows:

- DECA's first level, called as Classifier, is similar to Milne's Structural level;
- DECA's second level, called the Prioritizer, correlates to Milne's Behavioral level; and
- The Corrective Action segment of the DECA's inference engine resembles Milne's Functional level.

The Classifier performs the following tasks: (i) Read the input data to determine what plant variables are beyond the normal operating range and to assess their severity; (ii) determine the source of problem(s); and (iii) select feasible scenarios. This information is then fed to the Prioritizer whose major functions are: (i) identification of critical plant variables under the current operations, and (ii) determination of the priority of these plant variables on the basis of a number of scenarios. At the third level, the Corrective Action assimilates the information on the plant variable priority and scenario likelihood to determine the cause of the emergency. The duty of the Corrective Action is to decide which of the selected scenarios are most closely related to the problem under investigation. Once the likely scenario(s) are identified, DECA searches the knowledge base to identify appropriate action(s) that are sugested to the operator or autonomously implemented. In the event that DECA is unable to identify any action (possibly because of uncertainties associated with the selection of scenarios), the Corrective Action will provide the operator with a list of plant variable priorities and subsystem(s) where the problem is likely to be located.

The major factors in making prioritized decisions for any action are categorized as follows:[8,13]

Importance is attributed to each pertinent variable of the plant. The level of importance bears a direct relationship to potential consequences that may follow if this plant variable is ignored. More catastrophic the potential consequence would be, more important is the variable.

Stability is a measure of the impact of deviations of a plant variable from its desired response. More stable is the variable during a given time interval, safer is it to be set aside by the operator for any later action(s).

Urgency is a measure of the proximity of a plant variable to its nearest alarm level or safety limit.

A trade-off between the above three attributes is necessary to arrive at the decision of taking any specific action(s). Such decisions must be made in real

Table I. Critical plant variables in TMI–2.

Symbol	#	Description
PZR–P	1	Pressurizer pressure
PZR–L	2	Pressurizer level
HL1–T	3	Hot leg temperature
CL1–T	4	Cold leg temperature
SG1–P	5	Steam generator #1 pressure
SG1–L	6	Steam generator #1 level
SG2–P	7	Steam generator #2 pressure
SG2–L	8	Steam generator #2 level
QNT–P	9	Drain tank pressure

time because of the time-constraints (e.g., sampling time interval in digital control, and time-out periods in scheduled operations) in the plant under consideration.

B. Detailed Description of DECA

The features of DECA will now be discussed in detail by using the TMI–2 nuclear power plant accident as a prototypical example. Appendix B provides a brief functional description of the TMI–2 plant and explains how the accident happened. The nine plant variables of the TMI–2 plant that are pertinent to this specific accident[8] are listed in Table I. In view of the physical process that the TMI–2 plant had undergone[8,10,11] during this accident, twelve possible scenarios have been identified as listed in Table II. These scenarios are considered to be adequate for describing different events of the accident.

The databases in DECA are structured in an *a priori* known order. It is assumed that built-in test procedures (e.g., limit check and rate check) in the plant instrumentation routinely test the accuracy and validity of the collected data before their delivery to DECA. For the normal operation, nominal values

Table II. Description of possible accident scenarios in TMI–2.

Scenario #	Description
1	Pressurizer leak
2	Block valve leak
3	Drain tank disk rupture
4	Drain tank discharge
5	Pipe rupture PCS hot
6	Pipe rupture PCS cold
7	Reactor pump fail
8	Steam generator PCS
9	Steam generator SCS
10	Pipe rupture SCS
11	SCS feedwater pump fail
12	SCS turbine trip

of the critical variables are defined *a priori*. The ranges on the deviations from the nominal values are divided into a number of categories and subcategories to classify the severity of out-of-bound (*oob*) variables. As the sensor data is read into DECA, it is stored in an on-line information (OLI) relation. This history of transient data facilitates deduction of importance, stability, and urgency of individual plant variables.

1. Evaluation of Plant Variables

After processing a sensor record from the OLI database, the plant variables are computed for comparison with the respective data in the nominal-point-record (NPR) database. In this way, DECA determines whether a plant variable is within or beyond the normal operating range. Flags for the out-of-bounds (*oob*) plant variables are set, and their severity indices such as high (H), very high (HH), extremely high (HHH), low (L), very low (LL), and extremely low (LLL) are identified. If such information is unavailable or inadequate or found to be erroneous, DECA may consult an appropriate analytical module for computing the required nominal point(s).

There are several ways in which a plant variable and its nominal point can be compared and analyzed. This versatility is necessary to keep a generic format of DECA for use in a variety of dynamical processes. As an example, one such procedure (e.g., estimation of the plant variable) can be described by the following three rules.

Rule 1. Computation of pertinent plant variables using the sensor data from the OLI database. The sensor data are validated using the built-in-test procedures.

Rule 2. Comparison of the plant variables with their respective values in the NPR database. If a plant variable is greater than H or less than L, then it is flagged as *oob*.

Rule 3. Identification of severity index. An *oob* plant variable is compared with its respective values for LLL, LL, HH, and HHH in the NPR database, and the corresponding range where they are located is identified. This range determines the severity index for the *oob* variable.

The above rules and the associated database retrieval may require a hybrid formalism for efficient implementation, that is, different representations such as algorithmic, rule-based, and data-based schemes are preferred instead of using a single representation. For example, Rule 1 can be executed through procedural calls to appropriate algorithmic and/or database modules in a Lisp environment. Once this step is complete, the data from NPR are retrieved using the Sequential Query Language (SQL) for executing Rules 2 and 3. The plant variables can then be compared with respective severity index parameters (e.g., values from LLL to HHH) using an algorithmic approach.

C. Knowledge Base of DECA

The knowledge base and inference mechanism in DECA are developed as separate entities to ensure modularity and portability. The knowledge base is generated by integrating various pieces of information related to potential scenarios of malfunctions of the plant under consideration. In this article the knowledge base has been been constructed using the information from the TMI-2 nuclear power plant.

The submodules in the knowledge base of DECA only represent what are anticipated as possible disasters, and the completeness of the knowledge base largely depends upon the knowledge of the expert, thoroughness of the system design, and completeness of the analysis and coding. In general, this is not enough to cover all possible scenarios. To alleviate this problem, DECA discriminates between the root cause and the side effects by taking advantage of sensor and analytical redundancies[14–17] that are usually provided in large-scale strategic systems. For example, a well-designed failure accommodation plan may often point out a single gradually degrading component; this will prevent propagation of abnormal inputs through cascaded plant components, which might result in a myriad of alarm signals. In essence, this information will help determine the relative importance and urgency of individual plant variables at a given instant. This feature of DECA directs the user toward the source of the problem even if the exact cause is unknown. When DECA fails to clearly diagnose a problem, it lists the plant variables in the order of their (dynamically changing) priorities, and points out where the operators should concentrate their efforts. This is precisely what was needed during the TMI–2 accident because the operators, being overwhelmed with side-effect alarms, failed to notice the block valve that was stuck open until after the reactor core was damaged.

Table III shows a list of scenarios and the corresponding plant variables that were identified to be affected in the simulation of the TMI–2 accident. For

Table III. Relationship between scenarios and out-of-bound variables in TMI–2.

Scenario # (Table II)	Plant Variable # (See Table I)								
	1	2	3	4	5	6	7	8	9
1	X	X			X	X	X	X	
2	X	X			X	X	X	X	X
3	X	X							X
4	X	X							X
5				X	X	X	X	X	
6			X		X	X	X	X	
7			X	X	X		X		
8			X	X	X	X	X	X	
9				X	X	X	X	X	
10				X	X	X	X	X	
11				X	X	X	X	X	
12				X	X	X	X	X	

example, the senario 3 represents a pipe rupture in the drain tank. The scenarios 9 and 10 are affected by the sensor data SG1–P, SG1–L, SG2–P, SG2–L, and CL1-T (defined in Table I). These are the submodules which have to be searched for identification of most likely scenarios if the plant variables are *oob*. This is attempted by the inference engine via its look-ahead mechanism from a control perspective. The objective is to see how closely the ensemble of severity indices of the *oob* plant variables correlates to the expected scenarios and draw conclusions from these correlations. This approach may be subjected to combinatorial explosion if multiple scenarios are selected at several consecutive stages. Therefore, it is critical from the point of view of real-time execution that the number of likely scenarios at each stage be kept as small as possible.

1. *Variable–Expect Knowledge*

This data file is used by DECA to determine the rank of the plant variables. DECA checks to see what scenarios match up, and gives a rank according to the template given below. The list contains several sublists patterned in the following manner:

```
( (plant variable (rank (corresponding match list for the rank))
    (rank (number of scenarios needed for rank)))
)
```

For example, for the plant variable QNT–P, the list is as:

```
(QNT–P ((10  (2 3 4))
        (9.5 (2 4))
        (8.5 (2 3))
        (8   (3 4))     )
             (4.3 1)  ))
```

This indicates that, for QNT–P to have a rank of 10, the scenarios 2, 3, and 4 must be considered as possibilities by DECA. If only scenarios 2 and 4 are considered, then QNT–P will have a rank of 9.5. Similarly, for the scenario pairs, 2 and 3, and 3 and 4, the respective ranks will be 8.5 and 8. In the second sublist, if any one of the scenarios (i.e., 2 or 3 or 4) associated with the plant variable QNT–P is a possibility, then QNT–P will be assigned a rank of 4.3. This serves to cut down the number of scenarios associated with an *oob* plant variable, and thereby drastically reduce the probability of a combinatorial explosion.

2. *Scenario–Tendency Knowledge*

This part of the knowledge base contains information regarding the expected tendency of individual plant variables for a given scenario. Better the

expected tendencies match the plant variables, more likely that the scenario has actually occurred. For example, if the occurrence of senario 4 is suspected, DECA would look at the expected tendencies of the plant variables associated with scenario 4.

```
(4 ( (PZR-P lower)
        (PZR-L higher)
        (QNT-P higher)
   )
 )
```

In scenario 4, PZR-P would be lower than the nominal point, and PZR-L and QNT-P would both be running higher than their respective nominal points. If everything matches, then scenario 4, that is, drain tank discharge, would be one of the more likely explanations.

3. *Scenario Knowledge*

This part of the knowledge base contains all the pertinent plant variables of the process which DECA is monitoring. For each plant variable, there follows a list of scenarios which must be evaluated if that plant variable is flagged as *oob*. This list is accessed by DECA's look-ahead mechanism. For example, if the plant variable PZR-P is beyond its nominal point value (and flagged as *oob*), then DECA will access this data and determine that the scenarios 1, 2, 3, and 4 are possible events occurring in the process.

4. *Nominal Point Knowledge*

The first element of the data file indicates the number of plant variables under consideration. Next listed is the plant variable, then its operating mode(s) (e.g., normal, reactor shutdown, refueling). Next listed is the units of measure of the data, and finally a list of the nominal point value(s).

5. *Sensor Knowledge*

The sensor knowledge file contains the on-line information from actual measurements from the TMI-2 accident.[10,11] Each line represents information gathered during one time step. The first element is the time instant, and the sublist contains the sensor data readings for each of the pertinent plant variables being monitored. In this case, the time is in seconds after the turbine trip during the accident. The data is always read in the same order: PZR-P, PZR-L, HL1-T, SG1-P, SG1-L, QNT-P, SG2-P, SG2-L, CL1-T.

D. *Inference Mechanism*

DECA employs a data-driven forward chaining qualitative scheme. The basic modules for deriving inferences are as follows:

- Module for reading the sensor data and setting the severity indices of the *oob* plant *variables*.
- Module for looking-ahead to prioritize and predict the possible disaster scenarios and ranking them.
- Module for generating explanation and corrective actions.

A flow chart of the inference mechanism is shown in Figure 1. Initially the sensor data are read. Upon detection of a malfunction, the inference mechanism looks ahead at the possible scenarios and determines how well the criteria for their occurrences are satisfied. In the event of having no close match, the probable cause(s) of the malfunction are predicted, the priority levels of critical plant variables are specified, and further actions for fault identification are suggested. The above scheme could serve as a generic framework for monitoring and control of dynamical processes in real time.

The inference mechanism of DECA is a data-driven forward chaining system, and executes the following major functions:

- Look-ahead mechanism and scenario evaluation.
- Solution Search.
- Context tree generation and scenario ranking.

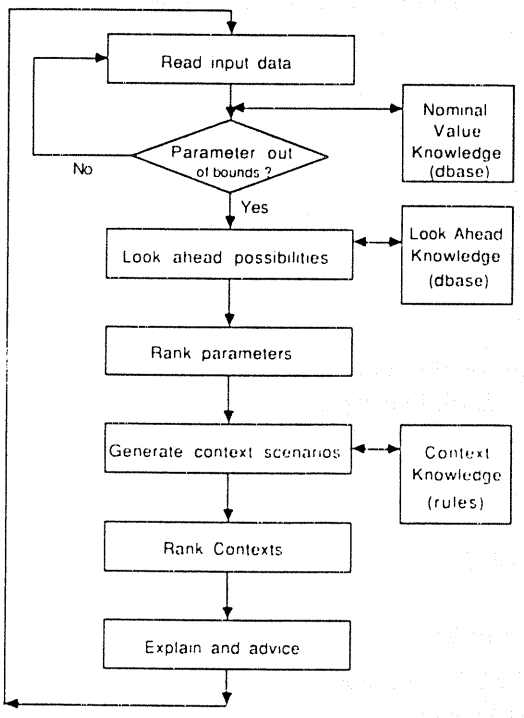

Figure 1. Generic inference scheme for dynamic process control.

The details of the modules of the inference scheme and the above functions are discussed next.

1. *Look-Ahead Mechanism and Scenario Evaluation*

Upon detection of a malfunction, the inference mechanism looks ahead at the possible scenarios and determines how well the criteria for their occurrences are satisfied. The scenario which is "best" satisfied is identified as the malfunction as long as it is above a specified level of confidence. In the event of having no close match, the probable cause(s) of the malfunction are predicted, the priority levels of *oob* plant variables are displayed, and further actions for fault identification are suggested. The look ahead mechanism is similar to MYCIN's control structure.[1] For example, with the pressurizer pressure PZR-P being *oob*, the look ahead mechanism would indicate that the scenario 1 of pressurizer leak (see Table II) is a possibility. In this case the plant variables SG1-P, SG1-L, SG2-P, and SG2-L should also be *oob*.

2. *Solution Search*

Having identified the affected plant variables, forward chaining is employed to ascertain the severity of these plant variables by comparison with the context trees. The confidence levels of individual diagnoses are also generated to assess their relative credibility. If there is a reasonable agreement between a scenario, its expected data, and plant variable criticality, then that scenario will be the present context to be considered. Multiple choices are often available for scenario selection. In that case, plausible scenarios are ranked from highest to lowest confidence levels. An example for possible choices for scenarios in the environment of the TMI–2 accident is given in Table IV.

If no appropriate match is found, the plant variables that are *oob* would be listed in the order of highest to lowest priority. The precedure for prioritization of a plant variable is dependent upon the confidence levels of the scenarios

Table IV. Confidence level of scenarios under TMI–2 accident simulation.

Scenario #	Confidence Level (0 to 1)
8	5/6
5	4/5
1	2/3
12	3/5
11	3/5
10	3/5
2	4/7
6	2/5
4	1/3
3	1/3

which involve this plant variable in accordance with the look-ahead database. In the above example of TMI–2 accident, if none of the possible scenarios are considered to have sufficient credibility, then the critical plant variables and their respective priorities would typically be:

Variable:	PZR–L	QNT–P	PZR–P	SG1–L	SG2–L	CL1–T	SG1–P
Priority:	10	10	10	9.3	9.3	8.6	8.5

In this case the pressurizer and drain tank are associated with the primary coolant system where, incidentally, the mishap occurred. Nevertheless DECA is able to point out the source of the problem even if it fails to generate a solution.

3. *Context Trees and Scenario Ranking*

The context tree contains rules to check the status of individual plant variables and their severity indices, if necessary. The objective is to judge how well the available data matches with a certain scenario which is encoded into the context trees as a pattern. DECA gives three levels of matching; High, Medium, and Low. These three levels help DECA to further consider the scenario that is being evaluated. High match implies a major consideration, and a medium match results in a minor consideration. For a low match, the scenario will not probably be considered. A finer resolution can be realized in the coding whenever necessary.

III. IMPLEMENTATION AND TESTING OF DECA

DECA has been implemented on a Symbolics 3670 Lisp Machine using the Symbolics Common Lisp Language and its object oriented extension *Flavors*.[12] This organization facilitated rapid prototyping and formulation of dynamic databases. Since the purpose of DECA is real-time monitoring of large plants, all computations must be performed within the time interval between two consecutive samplings of the sensor data. To meet these time constraints, the DECA software must be deployed with fast hardware such as Symbolics computers, Lisp on a chip micro-processors (e.g., Symbolics Ivory, TI Explorer Chip), or 32-bit high speed microprocessors (e.g., Intel 80386, Motorola 68030). From a purely implementation perspective, the Lisp environment offered sufficient expressive power to cover the architecture of DECA, and processing speed to satisfy the constraints of real-time operations. Therefore, we did not experiment with other programming languages such as Prolog and C.

DECA has the ability to use multiple databases for nominal points for different plant operating modes. In TMI–2 test run on DECA the nominal point database represented normal operations. Additional nominal point databases need to be developed for other modes like shutdown and start-up, and could be resident in several computers. Taking this one step further, the nominal point database does not have to be an array of numbers, it could be an analytical model which calculates these nominal points.

This implementation of DECA has the capability to interface with external models and use databases from multiple sources. The provision of external interfaces enhances the versatility of the software as it enables DECA to use other information to arrive at its decisions. Another important feature in DECA's capability to hook into simulation models. For example, DECA could call on a computer simulation to validate its conclusions before it makes a recommendation.

A. Test Results and Discussion

The general framework of DECA and its efficacy have been tested using the real-world problem of the Three Mile Island nuclear power plant number 2 (TMI–2) accident. (See Appendix B for a brief description of the accident.) The test run consisted of nine time steps at 0, 15, 30, 45, 60, 75, 90, 105, and 120 sec after the turbine trip and reactor shutdown. Details of the results are available in Ref. 18.

DECA completed the test run without any problem and consistently directed the operators to scrutinize the subsystem(s) that include the pressurizer, block valve, and drain tank. This is precisely where the problem was located. The stuck-open block valve in the pressurizer was allowing the reactor coolant to drain out of the primary coolant system. The test results consistently indicated that the pressurizer pressure and level (PZR–P, PZR–L) were among the most important plant variables (i.e., highest priority). These two sensors are located adjacent to the block valve. As DECA was only monitoring nine plant variables (listed in Table I) in this test run, it did not have the fine resolution to extract the intricate nuances present in the plant operations. Since the block valve, drain tank, and pressurizer directly affect each other, consistently locating these three plant components displays DECA's ability to identify the area(s) of importance.

The main problem which plagued the TMI accident was information overload. The reactor shut itself down after a turbine tripped, and soon after, a block valve opened and stuck open on the primary cooling system. With the draining of primary coolant and other events going on, the number of alarms that were being tripped in the control room caused the operators to overlook the real problem—the stuck-open block valve. It was not discovered for over two hours; by that time the damage had already occurred. DECA could have averted the TMI accident because it has the capability to keep track of many plant variables, figure out which are the most critical, and take corrective action or give the operator a summary of its findings. With this knowledge, the operators could have identified the problem in time.

B. Real-Time Processing Requirements

The DECA program execution consisted of loading in the knowledge base, sensor data, evaluating the plant performance at each time step, and writing the variable log and derived conclusions on the disk. For this, DECA required, on the average, 13.839 sec whereas the sampling time for sensor data collection was 15.000 sec for all nine readings in the test run described above.

The processing delays at different steps of the program execution were obtained by use of a built-in function which also kept track of the time spent in waiting for i/o as well as the amount and type of lists manipulated internally. DECA was tested under a variety of I/O loads with the same set of data in order to determine whee the bottlenecks occur during program execution.

The first test was a single run. The knowledge base data files were all loaded, and then the sensor data were evaluated and the results were displayed for a single time step. The elapsed time was 14.673 sec. Then this test was repeated with no information display at the terminal. The elapsed time was 6.178 sec. This shows that the I/O for the information display takes about 8.500 sec. The results were consistent with repetitive runs due to the fact that the Symbolics is a single user system and one does not have to wait for other jobs unlike a time-sharing system. The next test run consisted of a single time step evaluation without any I/O to the disk or terminal to determine the computational time for execution of the algorithm. The data processing time was evaluated to be 0.120383 sec, that is, about 8.3 time steps per second. The time spent in loading the TMI–2 data files was approximately 6 sec which can be eliminated by use of the active memory prior to the program execution. This will make DECA practical for monitoring and control of processes that require a turnaround time in the range of 0.5 to 1.0 sec per cycle.

IV. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS FOR FUTURE RESEARCH

A diagnostic decision support system has been developed for real-time control and monitoring of dynamical processes with the major objective of supporting human operators in the process of decision making. This system, known as Diagnostic Evaluation and Corrective Action (DECA), makes use of qualitative reasoning following the fundamental notions of Milne's theory of diagnosis.[9] DECA takes advantage of the existing quantitative techniques and integrates them with qualitative reasoning for monitoring, diagnosis and control of dynamical processes during both routine operations and time-constrained, emergency situations where an immediate action is necessary to avoid catastrophic failure(s).

DECA can be viewed as a multi-stage system. Given the current state, qualitative reasoning is used to diagnose the causes for any malfunction(s). For a given diagnosis, relevant variables are identified as functions of the current state so that any impending disaster can be avoided by effectively implementing a set of corrective actions.

At the first level, the plant data are compared with the nominal point database to identify the out-of-bound (*oob*) variables and determine their severity indices. At the second level, the look-ahead database is searched for matching various scenarios with the *oob* plant variables. At the third level, the plant variable priorities and confidence levels of identified scenarios are evaluated in consultation with the respective databases. DECA would attempt to identify a specific scenario that most closely represents the malfunction(s). If unable to

do so, DECA provides the operator with a list of the plant variable priorities and suggests which part of the plant is likely to be affected.

From the perspectives of control and monitoring of dynamical processes in real time, the contributions of DECA are as follows:

- Development and implementation of a novel architecture of a decision support system.
- Development of diagnostic capabilities to utilize qualitative reasoning.
- Integration of analytically derived models along with qualitative models.
- Modularity and portability of the system software.

A. Implementation of DECA

The software of DECA has been developed within a general purpose shell which is capable of handling computer operational details while the human user would concentrate on setting up a knowledge base for the particular application at hand. In the hierarchical decision structure, DECA functions are less autonomous at higher levels. DECA has a modular structure in the sense that it can be applied to a large class of dynamical processes without any major reprogramming, and integrated with existing software modules for performance enhancement. DECA can also be interfaced with the real-time operational environments of simulators and plants under control. DECA has been implemented on a Symbolics 3670 using common LISP in conjunction with *Flavors*.[12]

B. Application of DECA

DECA is particularly suitable for dynamical processes where many plant variables need to be monitored continuously. An example is the Space Station[13] which will need constant monitoring of its vibrational characteristics and stability. Another example is control of advanced fighter aircraft. While in a combat situation, the pilot not only has to fly the plane, but also has to keep track of the weapons systems and targets. DECA can be used to monitor all the rudimentary lower level controls, allowing the pilot to concentrate on the immediate danger at hand—the enemy. Nuclear power and chemical plants can benefit from DECA in a similar way.

C. Recommendations for Future Research

DECA is in the developmental stage. Initial results, generated from its prototype, support the need for expansion of its capabilities. Further research and developmental work is recommended in the following areas.

Use of Fuzzy Logic in quantifying the qualitative data. Plant variable priority and scenario ranking are potential areas for application of fuzzy mathematics.[19] This could significantly increase the resolution of the results. In the current version, DECA ranks the scenarios into three categories: major, minor, and improbable. More subtle points can be brought into consideration with fuzzy mathematics.

Integration of Analytical Modules. Analytical models and simple simulations of subsystems would complement DECA's qualitative models. Several analytical modules, running in parallel, can be integrated with DECA. For example, predictive models could be used for forecasting catastrophic failures or impending crises.

Source Code Translation. DECA is presently written in Lisp which provides for rapid prototyping and handling abstract concepts at the expense of its computational overhead. Since the primary concern of DECA is real-time processing, it will eventually be interfaced with physical controllers and sensors. Therefore the second stage of DECA needs to be developed using a source-level language like C that provides fast and efficient execution. The main problem to be addressed in this conversion is that the languages like C do not have dynamic database capabilities like Lisp. Data structures have to be carefully constructed to reduce the burden on the processor.

The authors acknowledge the benefits of discussion with Dr. James Stover of the Applied Research Laboratory of the Pennsylvania State University.

GLOSSARY AND SYMBOLS

Glossary

Abnormal operation: A plant component is said to be in abnormal operation if its performance has degraded beyond an *a priori* defined threshold relative to that of the normal operation. An abnormal operation does not imply a failure.

Disruption: A plant component is said to undergo a disruption from the normal operation if there are changes in one or more parameters beyond *a priori* defined threshold(s). A disruption may or may not cause an abnormal operation.

Failure: A plant component is said to undergo a failure if it is unable to perform the function for which it is designed. A failure is a gradual or an abrupt deviation from the normal operation to an abnormal operation.

Malfunction: Any detectable deviation from the normal operation of the plant is classified as a malfunction. A malfunction of the plant can be caused by disruption(s) or abnormal operation(s) or failure(s) of plant component(s).

Plant: The plant is a collection of interacting components which are integrated with the objective of fulfilling a certain objective. Examples are a nuclear power plant, a spacecraft, and an aircraft.

Scenario: A scenario is the state of the plant as a result of malfunction(s).

Symbols

DECA	Diagnostic Evaluation and Corrective Action
H	High
HH	Very high
HHH	Extremely high
L	Low
LL	Very low
LLL	Extremely low
Lisp	List processing language
NPR	Nominal point record
oop	Out of bounds
OLI	On-line information
PCS	Primary Coolant System
SCS	Secondary Coolant System
TMI	Three Mile Island

References

1. B. Buchanan and E. Shortliffe, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, MA, 1984.
2. O. Firschein et al., *Artificial Intelligence for Space Station Automation*, Noyes Publications, Park Ridge, NJ, 1986.
3. N.H. Narayanan and N. Viswanadham, "A methodology for knowledge acquisition and reasoning in failure analysis of systems," *IEEE Trans. on Systems, Man, and Cybernetics*, **SMC–17**(2), 274–288 (1987).
4. S.R.T. Kumara, S. Joshi, R.L. Kashyap, C.L. Moodie, and T.S. Chang, "Expert systems in industrial engineering," *International Journal of Production Research*, September–October 1987.
5. F.J. Dickey and A.L. Toussaint, "ECESIS: An application of expert systems to manned space stations," *The First Conference on Artificial Intelligence Applications*, December 1984, pp. 483–489.
6. D. Leinweber, "Expert systems in space," *IEEE Expert*, 26–36 (1987).
7. R.S. Shirley, "Some lessons using expert systems for process control," *Proceedings of the 1987 American Control Conference*, June 1987, pp. 1342–1346.
8. H.J. Jow, *Prioritization of Nuclear Power Plant Variables For Operator Assistance During Transients*, Ph.D. Thesis, Massachussetts Institute of Technology, Cambridge, MA, May 1984.
9. R. Milne, "Strategies for diagnosis," *IEEE Trans. on Systems, Man, and Cybernetics*, **SMC–17**(3), 333–339 (1987).
10. *Analysis of Three Mile Island Unit–2 Accident*, Nuclear Safety Analysis Center (NSAC–1), Electric Power Research Institute, Palo Alto, CA, July 1979.
11. *Supplement to Analysis of Three Mile Island Unit–2 Accident*, NSAC–1 Supplement, Electric Power Research Institute, Palo Alto, CA, October 1979.
12. D. Weinreb and D. Moon, *Flavors: Message Passing in the Lisp Machine*, AI Memo No. 602, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1980.
13. A. Ray, S.M. Joshi, C.K. Whitney and H.N. Jow, "Information prioritization for control and automation of space operations," *Proceedings of the 1987 American Control Conference*, June 1987, pp. 958–960.
14. A. Ray and M. Desai, "A redundancy management procedure for fault detection

and isolation," *ASME Journal of Dynamic Systems, Measurement, and Control*, 248–254 (1986).
15. J.C. Deckert, J.L. Fisher, D.B. Laning, and A. Ray, "Signal validation for nuclear power plants," *ASME Journal of Dynamic Systems, Measurement, and Control*, 24–29 (1983).
16. M. Kitamura, "Detection of sensor failures in nuclear plants using analytic redundancy," *Trans. Am. Nuc. Soc.*, **34**, 581–584, (1980).
17. A. Ray, A. Geiger, M. Desai, and J. Deyst, "Analytic redundancy for on-line fault diagnosis in a nuclear reactor," *AIAA Journal of Energy*, 367–373 (1983).
18. S.R. Nann, *A Decision Support System for Control and Automation of Dynamical Processes*, M.S.M.E. Thesis, Pennsylvania State University, December 1988.
19. A. Kandel, *Fuzzy Mathematical Techniques with Applications*, Addison-Wesley, Reading, MA, 1986.

APPENDIX A
MILNE'S THEORY OF DIAGNOSIS

Milne's diagnostic scheme is composed of four interconnected levels: (1) Structural; (2) Behavioral; (3) Functional; and (3) Pattern Matching. These four levels are serially connected as depicted in Figure A.1. The first level of diagnosis makes use of the knowledge about the plant structure. The expert system uses the structural knowledge about the process to simulate possible faults. On

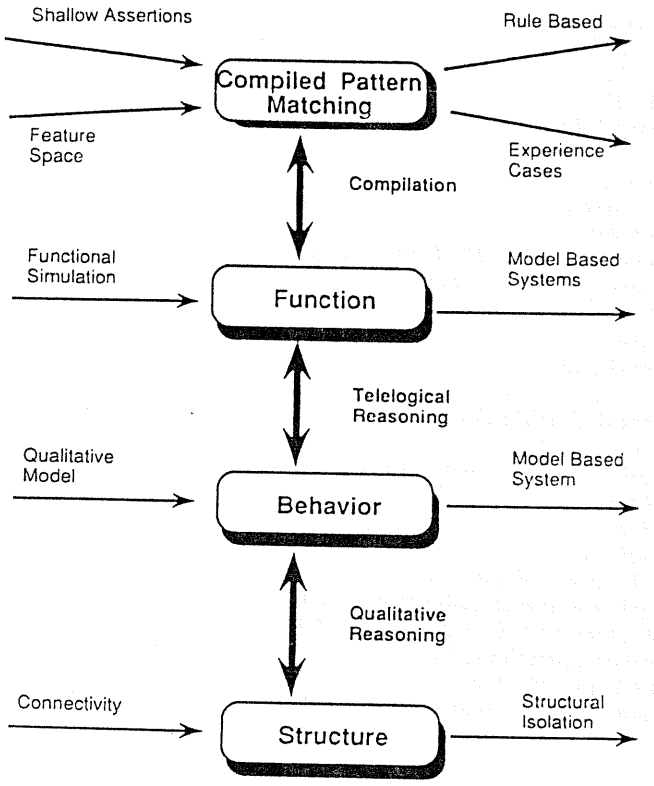

Figure A.1. Levels of diagnostic reasoning.

the basis of the simulation results, forward reasoning is used to arrive at a decision for fault diagnosis. This qualitative reasoning capability is not extensive in the first stage, and the knowledge is rather general. The diagnostic system that only uses the first stage is said to have *shallow* knowledge.

The second layer of Milne's architecture is the behavioral stage where the process of abstracting information involves a hierarchical organization of the relationship between a function and its structure. The idea is to organize an intended function as a series of behavioral states of the device and demonstrate how each behavior state transition can be understood as an interaction between two or more components. The reasonings in the second stage are much more complex than those in the first.

The knowledge deduced by the first two stages often suffices to diagnose a local problem in a specific device or a subsystem but not a global problem that may affect the entire plant. The third level of functional knowledge performs a diagnosis in which the behavior of the device is examined at a higher level of knowledge representation. This higher level knowledge generally relates the interactions between a function and its structure. It can also be structured into a hierarchical relationship between the subsystems. The last stage of pattern matching is referred to as *Deep Function Model-Based Diagnosis System* which allows the information to enter at any one of the four levels, utilize the strengths of one or more level in arriving at a decision, and exit at any level.

APPENDIX B
DESCRIPTION OF THE TMI–2 ACCIDENT

The nuclear steam supply system (NSSS) of the TMI–2 nuclear power plant is equipped with a pressurized water reactor (PWR) and two once-through steam generators. Scenario of the accident that happened in March 1979 at TMI–2 is briefly described below:[10]

The accident was initiated by a loss of feedwater to the steam generators resulting in a turbine trip. This instant of accident is denoted as $t = 0$. The interruption of feedwater flow to, and of steam flow from the steam generators caused a reduction in heat removal from the reactor coolant system (RCS). The RCS responded to this event in a normal manner which is described as follows: RCS pressure increased because thermal energy was not being removed from the RCS at a sufficient rate; the power operated relief valve (PORV), also called electromatic valve, opened to relieve pressure; the reactor was automatically shut down because of a high-pressure trip signal; heat generation from the reactor dropped to the decay heat level; within a few seconds the RCS pressure dropped to the normal level. Up to this point, normal reactor protection features functioned as designed. At $t \approx 40$ s, the steam generator level dropped to the point where automatic controls called for emergency feedwater to maintain a minimum steam generator level. However, closed valves between the steam generators and control valves prevented emergency feedwater being delivered to the steam generators. (These valves were opened by the plant operator at $t \approx$ 8 min) The opening of the electromatic relief valve in the event of a loss of

feedwater transient, whether or not feedwater is available, is normal in accordance with plant design.

The electromatic valve, which relieved excess pressure as intended, should have automatically closed as pressure was reduced sufficiently. Instead, it remained open, thereby allowing continued coolant discharge from the RCS, and causing a further decrease in RCS pressure. At $t \approx 2$ min, the safety injection system became active in response to low pressure signal (1640 psig) in the RCS. Because of the continuing flow from the electromatic valve into the drain tank, a safety valve on this tank opened at $t \approx 3$ min and rupture disk on the tank burst at $t \approx 15$ min. This loss of the reactor coolant continued without interruption for approximately 2.4 hours when the stuck-open valve was eventually closed.

The above description summarizes the critical phenomena in the TMI–2 accident. During the first two minutes after the accident, the key events are listed below:[8]

- The turbine tripped automatically as a result of trip of both feedwater pumps.
- Emergency feedwater pumps started automatically after the trip of feedwater pumps.
- Reactor tripped on high reactor coolant system (RCS) pressure.
- Electromatic relief valve opened to relieve RCS pressure but stuck open after pressure was reduced.
- RCS pressure continued to decrease.
- Reactor coolant drain tank pressure steadily increased.
- Both steam generators boiled dry on the secondary side.
- High pressure safety injection was actuated on low RCS pressure.

During this two-minute period the operators were overwhelmed with too many alarms while responding to the turbine trip, reactor trip, and complications in the condensate and feedwater system.