4

Applied Research Laboratory

Technical Report

A DECISION SUPPORT SYSTEM FOR CONTROL AND AUTOMATION OF DYNAMICAL PROCESSES

by

Steve Nann

PENNSTATE
1855

4

The Pennsylvania State University
APPLIED RESEARCH LABORATORY
P.O. Box 30
State College, PA 16804

A DECISION SUPPORT SYSTEM FOR CONTROL AND AUTOMATION OF DYNAMICAL PROCESSES

by

Steve Nann

Technical Report No. TR 90-002
March 1990

Supported by:
Space and Naval Warfare Systems Command

L.R. Hettche, Director
Applied Research Laboratory

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b DECLASSIFICATION / DOWNGRADING SCHEDULE

3 DISTRIBUTION / AVAILABILITY OF REPORT
Approved for public release; distribution unlimited.

4 PERFORMING ORGANIZATION REPORT NUMBER(S)
TR-90-002

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a NAME OF PERFORMING ORGANIZATION
Applied Research Laboratory
The Pennsylvania State University

6b OFFICE SYMBOL *(If applicable)*
ARL

7a. NAME OF MONITORING ORGANIZATION

6c ADDRESS *(City, State, and ZIP Code)*
P. O. Box 30
State College, PA 16804

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Space and Naval Warfare Systems Command

8b OFFICE SYMBOL *(If applicable)*
SPAWAR

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c ADDRESS *(City, State, and ZIP Code)*
Department of the Navy
Washington, DC 20363

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.

11 TITLE *(Include Security Classification)*
A Decision Support System for Control and Automation of Dynamical Processes

12 PERSONAL AUTHOR(S)
Steven R. Nann

13a TYPE OF REPORT
M. S. Thesis

13b. TIME COVERED
FROM ________ TO ________

14. DATE OF REPORT *(Year, Month, Day)*

15. PAGE COUNT
152

16 SUPPLEMENTARY NOTATION

17 COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*

The thesis presents the concept and development of a diagnostic decision support system for real-time control and automation of dynamic processes. This system, known as DECA (Diagnostic Evaluation and Corrective Action), will take advantage of the computer's ability to manipulate vast amounts of data, and employ qualitative reasoning for the monitoring and diagnosis of dynamical processes during time-constrained, routine, and emergency situations where an immediate response is necessary to avoid catastrophic failure of the system.

The software system's architecture has been structured in such a manner that it can be applied to any dynamic process without reprogramming. DECA is written in Lisp and was verified using the data from the Three Mile Island Nuclear Reactor Accident.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED □ SAME AS RPT □ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a NAME OF RESPONSIBLE INDIVIDUAL

22b TELEPHONE *(Include Area Code)*
(814) 865-6344

22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

ABSTRACT

The thesis presents the concept and development of a diagnostic decision support system for real-time control and automation of dynamic processes. This system, known as DECA (Diagnostic Evaluation and Corrective Action), will take advantage of the computer's ability to manipulate vast amounts of data, and employ qualitative reasoning for the monitoring and diagnosis of dynamical processes during time-constrained, routine, and emergency situations where an immediate response is necessary to avoid catastrophic failure of the system.

The software system's architecture has been structured in such a manner that it can be applied to any dynamic process without reprogramming. DECA is written in Lisp and was verified using the data from the Three Mile Island Nuclear Reactor Accident.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY OF TERMS

AI - Artificial Intelligence

ES - Expert System

DECA - Diagnostic Evaluation and Corrective Action

TMI-2 - Three Mile Island Unit 2 reactor

CKW - Local Optimization Algorithm

OOB - Out Of Bounds, refers to system operating conditions

OLI - On-Line Information database, contains sensor readings

SDB - Setpoint Database

L, LL, LLL - qualitative range below predetermined value

H, HH, HHH - qualitative range above predetermined value

SQL - System Query Language

DML - Data Manipulation Language

Input parameters for the TMI-2 accident test scenario

PZR-P	Pressurizer Pressure
PZR-L	Pressurizer Level
HL1-T	Hot Leg 1 Temperature
SG1-P	Steam Generator 1 Pressure
SG1-L	Steam Generator 1 Level
QNT-P	Drain Tank Pressure
SG2-P	Steam Generator 2 Pressure
SG2-L	Steam Generator 2 Level
CL1-T	Cold Leg 1 Temperature

Scenario numbers and descriptions.

1	Pressurizer Leak
2	Block Valve Leak
3	Pipe Rupture - (drain tank)
4	Drain Tank
5	Pipe Rupture - Hot Leg, Primary Coolant System
6	Pipe Rupture - Cold Leg, Primary Coolant System
7	Reactor Pump
8	Steam Generator - Primary Coolant System
9	Steam Generator - Secondary Coolant System
10	Pipe Rupture - Secondary Coolant System
11	Feedwater Pump - Secondary Coolant System
12	Turbine Trip - Secondary Coolant System

ACKNOWLEDGEMENTS

Any major undertaking, such as this thesis, requires the devotion of much time and effort to insure its successful completion. I would like to express my deepest appreciation to those whose assistance enabled me to complete this work.

I would like to thank my Advisor, Asok Ray, for giving me the opportunity to become involved in the Artificial Intelligence field, and for his guidance in developing this thesis.

Also, I would like to thank Jim Stover for his help and advice on techniques for the development of Expert Systems.

I am grateful to Ron Gibson for his help with the generation of the computer output and file translations.

Most of all, I would like to express my deepest appreciation to Soundar Kumara whose help and insight were instrumental in the completion of this research.

Finally, I would like to thank the Applied Research Laboratory for offering an assistantship which enabled me to pursue my Master of Science degree in Mechanical Engineering.

Chapter 1

Introduction

Expert Systems have been proclaimed as the panacea for Industry's problems in recent years. They are said to incorporate vast amounts of knowledge, learn and use this knowledge, all in a broad domain. While in the realm of implementations, the past history shows that the most useful Expert Systems have been applied to a specific domain to carry out a set of specific tasks. Some examples of successful Expert Systems are MYCIN [Bucha84] and R1 [McDer82]. In the realm of real-time processing, some successful Expert Systems are Picon [Leinw87], Falcon [Shirl87], and Ecesis [Dicke84]. The reason for their success is that they have a workable size knowledge base and the complex interactions are not beyond the computational power currently available.

1.1 Motivation

The advent of powerful computers has created useful applications for dynamic control. The computer's ability to handle vast amounts of information efficiently makes it ideal for monitoring large dynamical systems.

Earlier work has been in the area of dynamic programming, where the globally optimal solution to the problem is used. As the system's complexity increases in the number of parameters, processing time required to find the optimal solution increases exponentially. The increased processing time required makes the dynamic processing methods impractical for dynamic process control or monitoring systems in real-time.

Previous research concentrated on developing a locally optimal solution which will enable the real-time monitoring of dynamic processes. Jow's work [Jow84] in the implementation of the CKW local optimization algorithm (appendix C) makes real-time monitoring of many parameters in a complex dynamic process feasible.

In recent years, with the development of efficient computers for running Artificial Intelligence (AI) applications, the development of AI based applications for monitoring and control of complex dynamic processes in real-time has become a possibility. This thesis explores one possible way to monitor these dynamic systems using AI and Expert System techniques. It builds upon Jow's work, but instead of an algorithmic approach, it uses qualitative reasoning. This thesis uses some of the fundamental notions of Milne's theory of diagnosis [Milne87].

1.2 The DECA System

In this thesis, the domain of dynamical processes is considered as a candidate for real-time monitoring and information prioritization. If the common elements of all dynamical processes are focused upon, it appears feasible to be able to develop a generic Expert System that can be used with these various systems. The outcome is expected to be largely similar to the project presented here; DECA, **Diagnostic Evaluation and Corrective Action.** DECA is designed to implement a control strategy for dynamical processes during routine, time-constrained, and emergency situations. The goal is to develop DECA into a general purpose shell which would be 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 can be interfaced with simulators or with the plant under control. The major objective of DECA is to support human operators in the decision making process. In the hierarchical decision structure, DECA will function less autonomously at higher levels.

1.3 Overview of DECA

The system is a multi-stage one. Given the current state of the system, in real time, DECA tries to diagnose the causes for any malfunction, based on qualitative reasoning [DeKle83]. At the lower level, for the given diagnosis, DECA tries to identify the relevant variables along with more detailed analysis of the current state so that any impending disaster can be avoided by effectively implementing a set of corrective actions.

There are three main levels in the DECA system: Diagnostic-1, the Classifier; the Prioritizer; and Diagnostic-2, the Corrective Action implementor. Figure 1.1 shows the system structure. Chapter 3 gives a detailed process flow diagram and explanation of the system.

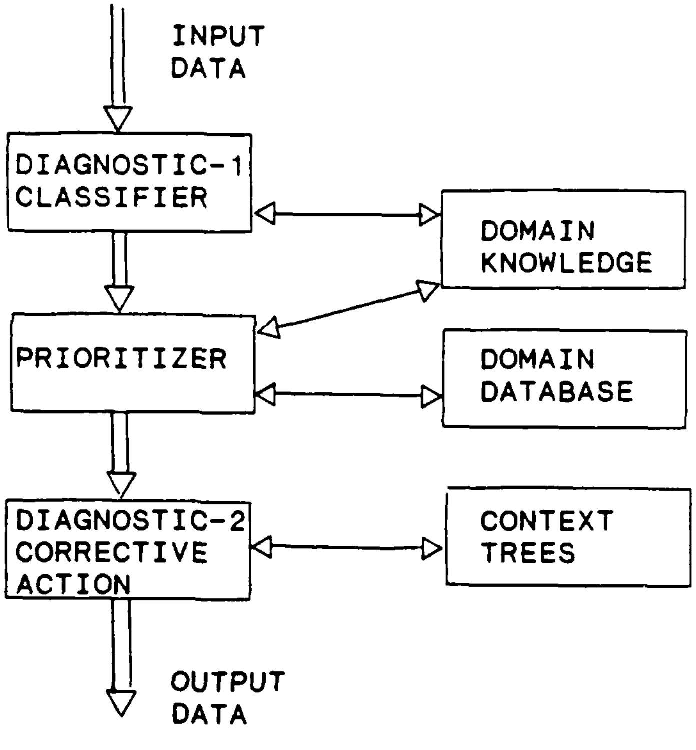


Figure 1.1 Overview of DECA System

In the first level, the Classifier, DECA reads the input data, determines what parameters are beyond normal operation and their severity. The second stage, the Prioritizer, evaluates the parameters and finds their importance as well as searches the database for possible disaster scenarios. The third stage, Corrective Action, pulls together the parameter priority, scenario likelihood and determines the cause of the emergency. Once the scenario is selected, it searches its knowledge base for a likely solution or action for the operator to implement.

1.4 Areas of Application

DECA is particularly suitable for dynamical systems where many parameters need to be monitored continuously. For example, the Space Station will need constant monitoring of its vibrational characteristics and stability [Firsc86, Ray87]. Another area of application is the control of advanced fighter aircraft [Ander84, Pisan84]. 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. An Expert System can be used to monitor all the rudimentary lower level controls, allowing the pilot to concentrate on the immediate danger at hand - the enemy. Chemical and nuclear processes can also benefit from DECA in a similar way.

The system's control process can be invoked via manual intervention or through an automatic mode. In the former case, the diagnostic system will serve as an overall advisor. In the latter case, it, by virtue of proper interfaces would be able to control the process. For example, a space probe in the outer reaches of the solar system would be able to take some emergency actions to preserve itself.

1.5 Application to Evaluate System

The general framework of DECA and its efficacy have been tested using real world problem of the Three Mile Island nuclear power plant number 2 (TMI-2) accident. This particular example was considered due to the availability of data and earlier reports [NSAC-1, NSAC-1S] which can be used for evaluating the performance of DECA.

The main problem which plagued the accident was information overload. Studies have shown that the average human can handle about seven pieces of information before reaching information overload [Mille56]. This is exactly what happened at TMI. 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 culprit - the stuck block valve. It was not discovered for over two hours, after the damage was done.

DECA could have been of value in TMI because it has the capability to keep track of many parameters, figure out which are the most critical, and take corrective action or give the operator a summary of its knowledge. With this knowledge, the operators will be able to solve the root of the problem and the side effects will resolve themselves.

1.6 Contribution of the Thesis

The objective of this thesis is to develop a kernel of a future Expert System for autonomous dynamical process control. As it is refined, more capabilities will be added to enable it to automatically control a simulated or physical system. In its current stage of development, DECA plays the role of an advisor to the system operators.

This thesis demonstrates an inference engine that can be used for the automation of monitoring dynamical processes in real-time. The contribution of the DECA system is as follows:

1-Develop and implement a new architecture specifically designed to automate the monitoring and control of dynamical processes.

2-The capability to run in real-time.

3-Develop the system's diagnostic capabilities to utilize qualitative reasoning.

4-Give DECA the ability to integrate analytical models along with the qualitative models.

5-Keep the system modular to enhance software maintenance.

6-Develop the system in such a manner to insure portability.

1.7 Implementation of DECA

DECA is being implemented on a Symbolics 3670 using common LISP in conjunction with Flavors [Weinr80]. Currently, the system development uses the Three Mile Island reactor accident data. However, the framework of DECA is general enough to be applied to a variety of dynamic control situations.

1.8 Organization of Thesis

This thesis consists of seven chapters, a list of references, and six appendices.

Chapter 2 discusses theoretical issues and draws parallelisms to Milne's theory of diagnostics. In chapter 3, a detailed description of DECA's structure is given and a discussion of the inference engine is located in chapter 4. Chapter 5 provides the details of developing the DECA program on the Symbolics computer. Chapter 6 reviews and discusses the results from the test runs of the software and draws conclusions about the system. In chapter 7, the goals of future research are given.

A list of the references is provided after chapter 7. Appendix A provides background information about the System Query Language. Additional background information is given in appendix B and C. Appendix B describes the control strategies in MYCIN, while appendix C discusses the CKW algorithm for the local optimization of systems in real time. Appendix D consists of the data from the execution of the DECA program for the test run and for the Three Mile Island Reactor Accident. Appendix E describes the knowledge base for the Three Mile Island Reactor and gives listings of the files which contain the data. Finally, appendix F gives the listing of the DECA program.

Chapter 2
Milne's Theory of Diagnosis

In recent years there has been a tremendous leap forward in technology leading to new applications of Artificial Intelligence and Expert Systems, especially in the area of diagnostics. According to Milne [Milne87], there are many new techniques available giving us the ability to build and reason about models dealing with a large domain of information (e.g. learning from experience, probabilistic information, and learning from examples). The DECA system's architecture in many respects, parallels Milne's concept of diagnostics and reasoning.

2.1 Levels of Diagnosis

In a diagnostic system, the key to a successful implementation is through the system's ability to be flexible in its interface with the physical system. The ability to accept input data in a format or description that is most logical with a particular domain is also critical for a successful implementation.

The manner in which Milne creates this is by having a network of different levels which can readily pass data between one another giving it modularity. This flexibility makes the approach "generic", and usable, for a wide variety of diagnostic applications.

In the subsequent discussion Milne's framework is explained and its parallel with DECA is illustrated.

In Milne's diagnostic scheme [Milne87] there are four levels that are layered together. They are:

1 - Structural
2 - Behavioral
3 - Functional
4 - Pattern Matching

The four levels are connected together serially 1-2-3-4, and each level has both input and output. With this setup, one "can build a diagnostic system based on knowledge which has been input at any level and stop at any level" [Milne87 p. 334]. Figure 2.1 graphically depicts the interrelationships between the levels and system *input/output* (*i/o*).

In the first level of diagnosis, the system uses the knowledge about the system structure for diagnosis. The system contains a small number of hypotheses of what is wrong. It will derive tests to discriminate between the hypotheses. In the structural level the expert system uses the structural knowledge about the process and system to simulate possible faults and compare the results to an on-line library. From the results of the simulation, it will use forward reasoning to qualitatively select the proper diagnosis. The qualitative reasoning capability is not extensive in the first stage, and the depth of knowledge about the system is generalized. Since this is not an extensive model of the systems, a diagnostic system using only the first stage is called a "shallow" knowledge system.

The second stage of Milne's diagnostic architecture is the behavioral stage. It performs the following function: "Given a representation of the behavior of components of the devices, system, and a representation of the components, the ability to generate the behavioral description of the device as a whole is an important part of causal reasoning" [Milne87, p. 83]. In the second stage the reasoning becomes much more complex than in the first. There are two methods to carry out the reasoning: qualitative simulation, and the consolidation method [Dekle83].

In general, for the diagnosis of a simple system, only the first two stages are needed. They give a fragmented evaluation of the interrelationships of the devices in a larger system. If the application is a very complex system, there will likely be a need to tie together the fragmented information in a hierarchical structure. On a large system one "can often put together function of the device and relationship to its structure" [Milne87, p. 334] using the two lowest levels.

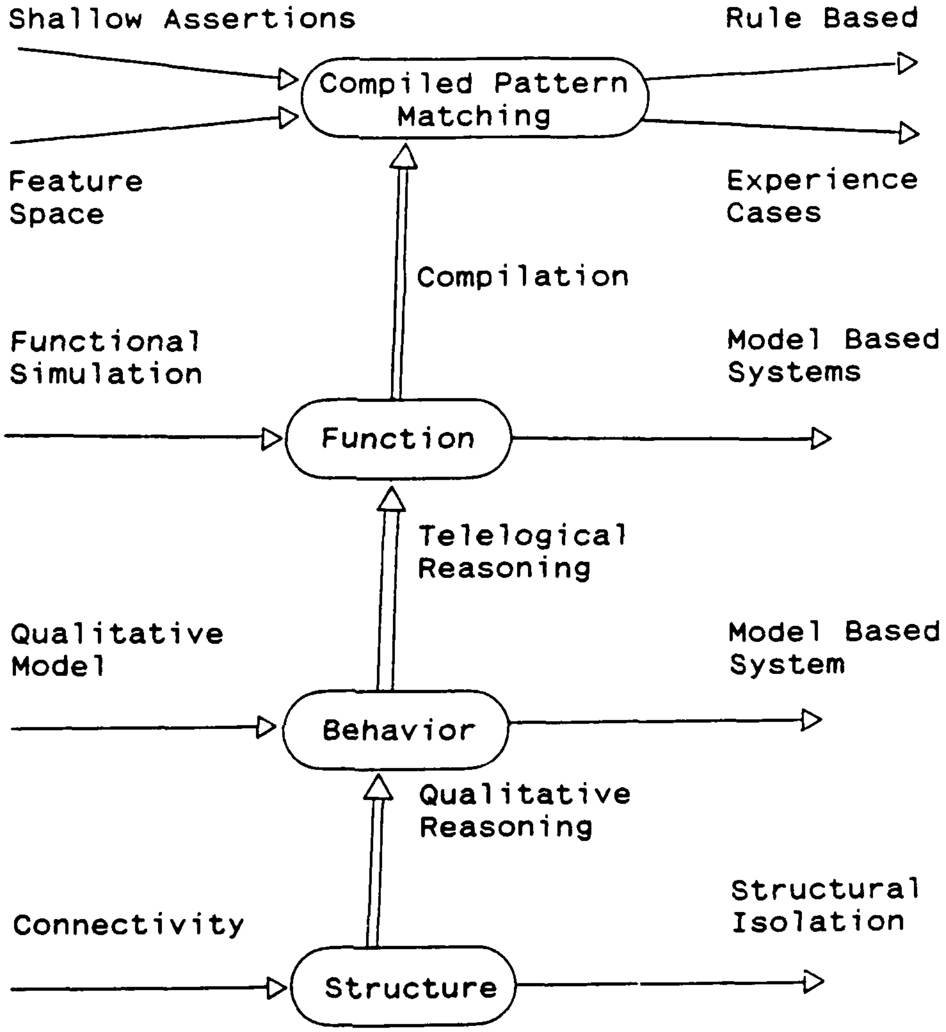


Figure 2.1 Milne's Levels of Diagnostic Reasoning [Milne87, p. 334]

The third stage is the functional level. Sometimes the knowledge deduced by the first two stages is enough to diagnose a problem in a device of the system, but not the complete evaluation of the whole system. To perform a diagnosis it may be necessary to have the behavior of the device go through an abstraction to a higher level of knowledge representation. The higher level generally relates the interactions between function and structure. It can also be patterned into a hierarchy of the interrelationships between the devices in the system.

The fourth stage is what Milne refers to as the "Deep Function Model-Based Diagnosis System" [Milne87, p. 335]. Taking the model and having a knowledge representation to relate function, behavior, and structure, it can perform the top-level pattern-matching.

The deep function model-based diagnostic system would enable the information to enter at any of the four levels, utilize the strengths of one or more level and exit at any level. This would basically yield any of four types of diagnostic systems available.

2.2 Correlation Between Milne's Levels and DECA

Milne's frame work forms the basis for DECA's architecture. Referring to figure 2.1, it is apparent that DECA employs the concepts of Milne's first three levels, Structure, Behavior, and Function. Essentially the correlation between DECA and Milne is as follows; DECA's Classifier is the same as Milne's Structure, Prioritizer is equivalent to the Behavior, and the Corrective Action module is equivalent to Milne's Function level.

The Classifier uses the system structure to determine where the problems are arising on a subsystem level, and selecting general scenarios which may be feasible. It then feeds this data to the Prioritizer.

The Prioritizer section correlates well with Milne's Behavior level, for the Prioritizer decides which parameters are the most critical from the data given to it about the system. It also takes into account the interactions of the elements of the system and general tendencies of the system's components under a given condition. The Prioritizer will determine the priority of the system parameters and select a few most likely scenarios as to what the malfunction is.

The Corrective Action segment of the DECA's inference engine closely correlates with Milne's Function layer. The duty of the corrective action layer is to determine which of the scenarios chosen by the Prioritizer is the actual system malfunction and then figure out what would be the best remedy to implement. In the event that DECA cannot find a solution, the Corrective Action segment will give the operator a list of the parameter priorities and a brief description of which part of the system he should concentrate his efforts on.

In summary, one can see that DECA's structure closely parallels Milne's. DECA's ability to adapt its structure and internal function, and its modularity make it feasible to be implemented in a variety of applications in automatic diagnosis and control for dynamical systems.

Chapter 3

Detailed Description of the DECA Kernel

This chapter gives a detailed description of DECA's internal architecture and process flow. An overview of the major components of DECA are discussed in section 1.3.

3.1 Design Goals for DECA

When the DECA architecture was developed, the following objectives were incorporated into the system. They are:

1- For DECA to monitor many parameters in a real-time fashion.

2- The ability to quickly separate all relevant data from extraneous information.

3- Diagnose the system's malfunction/abnormality, and if not possible

4- Output the relevant parameters and their priority in order to focus the operator's attentions to the part of the system which the problem emanates from, thus eliminate side effect distractions.

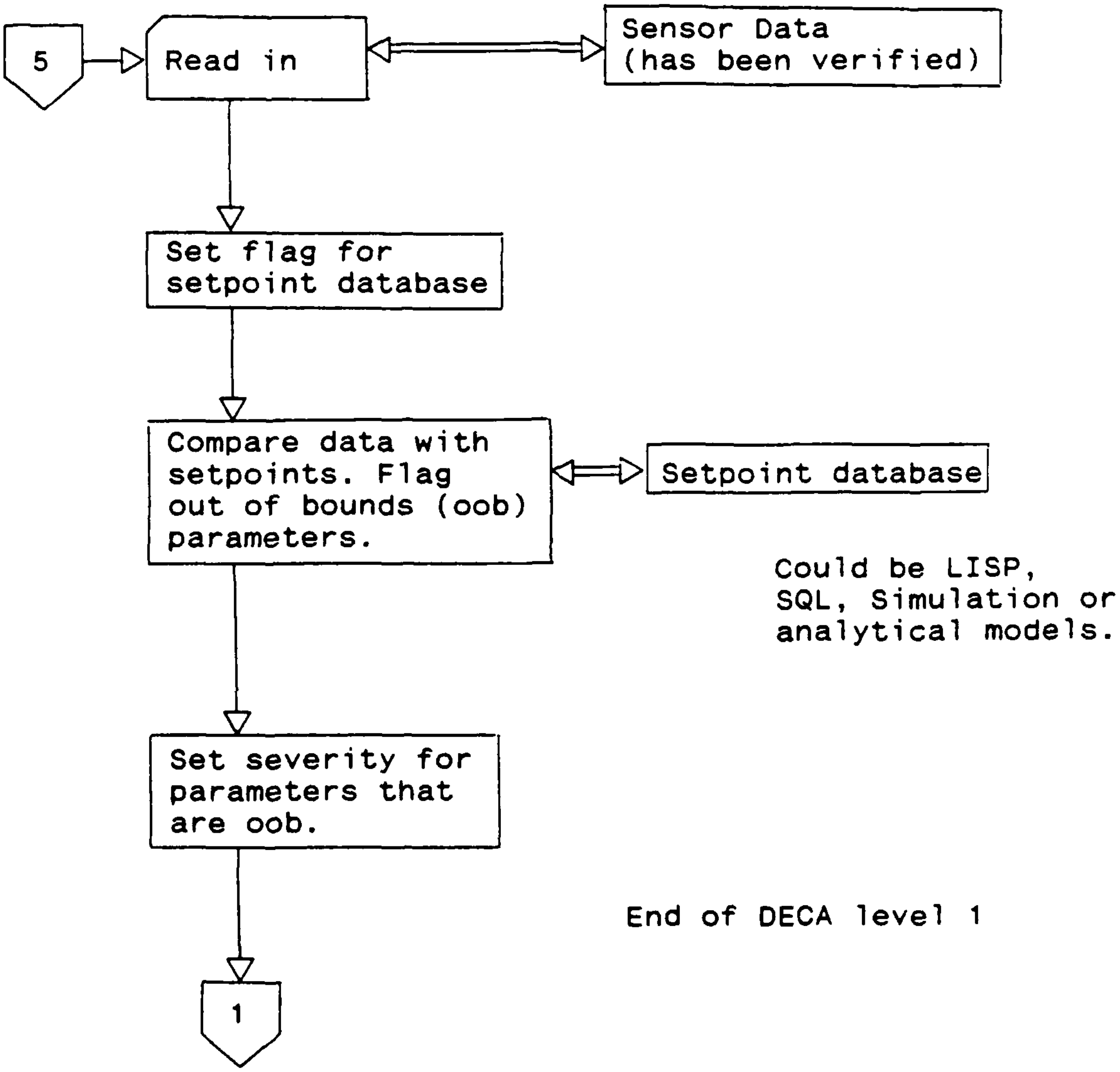
3.2 Process Flow Chart

Figure 3.1, contains a detailed process flow diagram of the DECA system. In subsequent discussion, a detailed explanation of DECA kernel is given.

From the DECA process flow, it can be seen that the knowledge bases for the application problem have been kept separate from the inference engine. This has been done for two reasons. First, keeping the domain data in a separate database enables the operator to easily update the system to represent any changes in the physical system. Secondly, with a separate inference engine, DECA can be adapted to a great many different dynamical processes. Only the domain knowledge needs to be incorporated for each new application.

The modular structure will also help improve DECA's versatility, for the reference databases need only be changed when DECA is applied to another problem domain. Also, DECA can be modified to run subroutines instead of referencing data in certain databases. For example, DECA can be told to access an analytical model or run a simulation for retrieving setpoint data instead of looking up a data table. It can interact with its databases, analytical models and simulation modules in a coherent manner. The data and subroutines do not even have to be resident in the same computer, enabling DECA to use distributed computing techniques. This also allows DECA to take advantage of previous information without recoding it.

The following pages deal with the detailed analysis of the DECA kernel. Figure 3.1 contains the flowchart for DECA.



Could be LISP, SQL, Simulation or analytical models.

End of DECA level 1

Fig. 3.1 DECA Process Flow

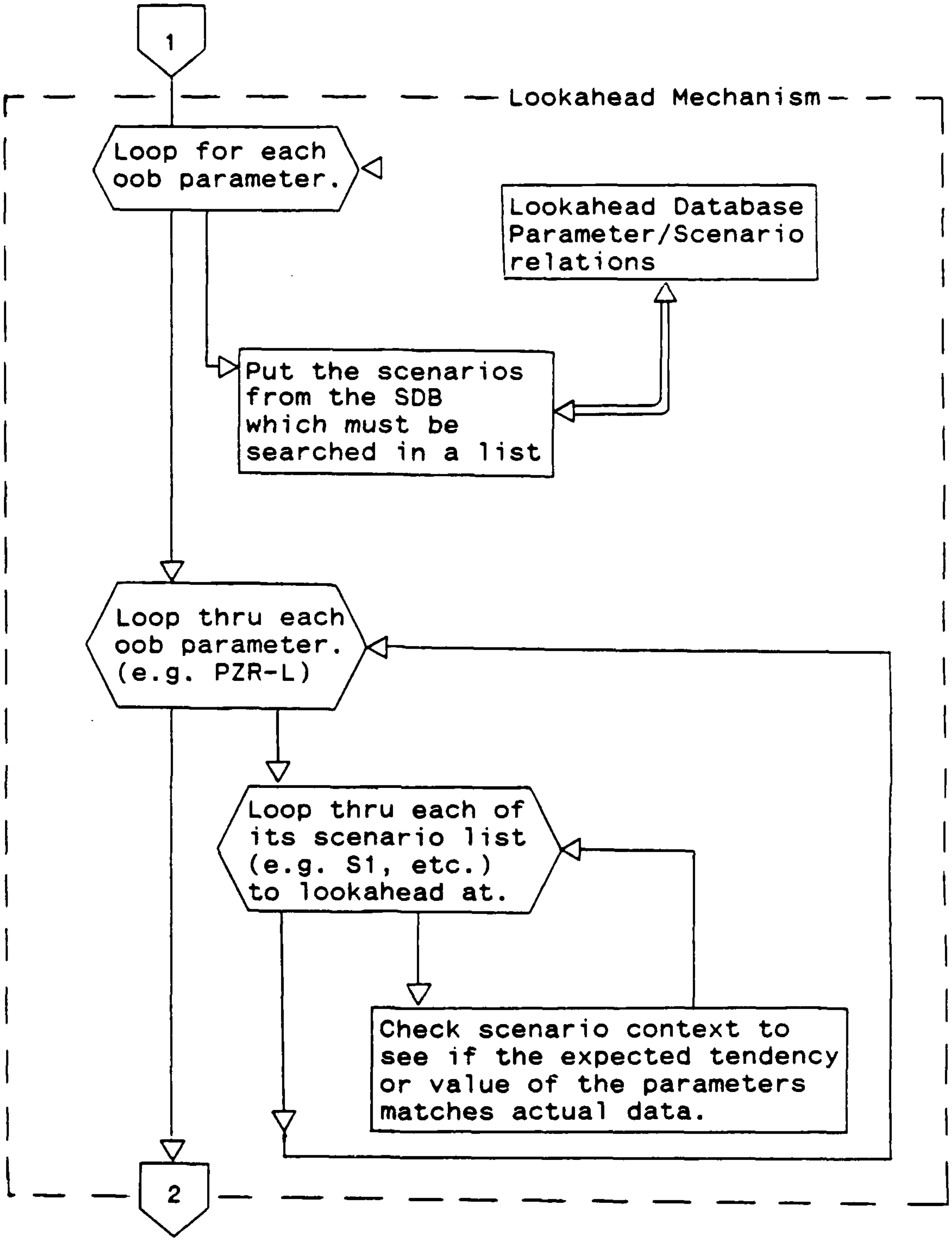

1
Lookahead Mechanism
Loop for each oob parameter.
Lookahead Database Parameter/Scenario relations
Put the scenarios from the SDB which must be searched in a list
Loop thru each oob parameter. (e.g. PZR-L)
Loop thru each of its scenario list (e.g. S1, etc.) to lookahead at.
Check scenario context to see if the expected tendency or value of the parameters matches actual data.
2

Fig. 3.1 continued

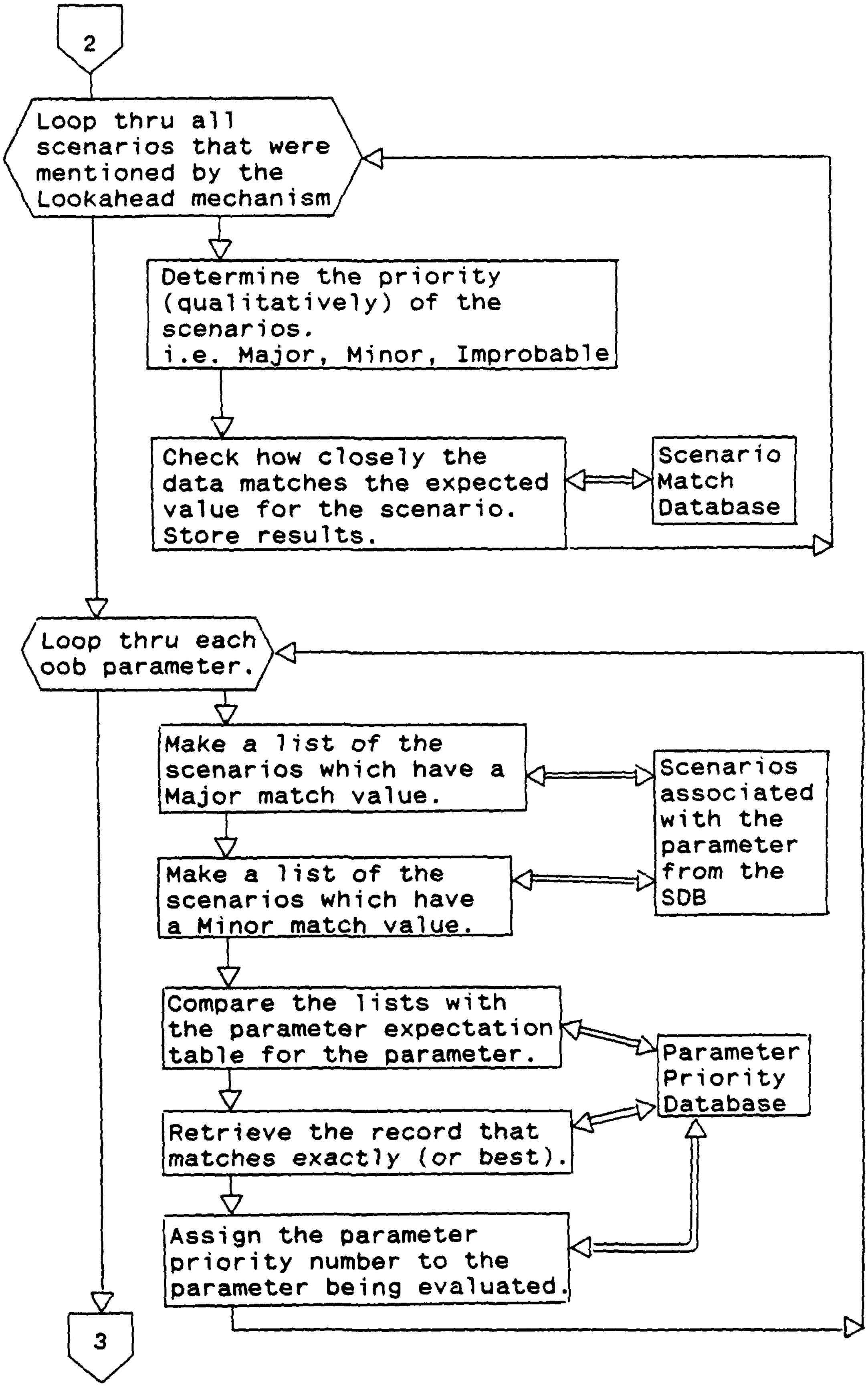

2
Loop thru all scenarios that were mentioned by the Lookahead mechanism
Determine the priority (qualitatively) of the scenarios.
i.e. Major, Minor, Improbable
Check how closely the data matches the expected value for the scenario. Store results.
Scenario Match Database
Loop thru each oob parameter.
Make a list of the scenarios which have a Major match value.
Scenarios associated with the parameter from the SDB
Make a list of the scenarios which have a Minor match value.
Compare the lists with the parameter expectation table for the parameter.
Parameter Priority Database
Retrieve the record that matches exactly (or best).
Assign the parameter priority number to the parameter being evaluated.
3

Fig. 3.1 continued

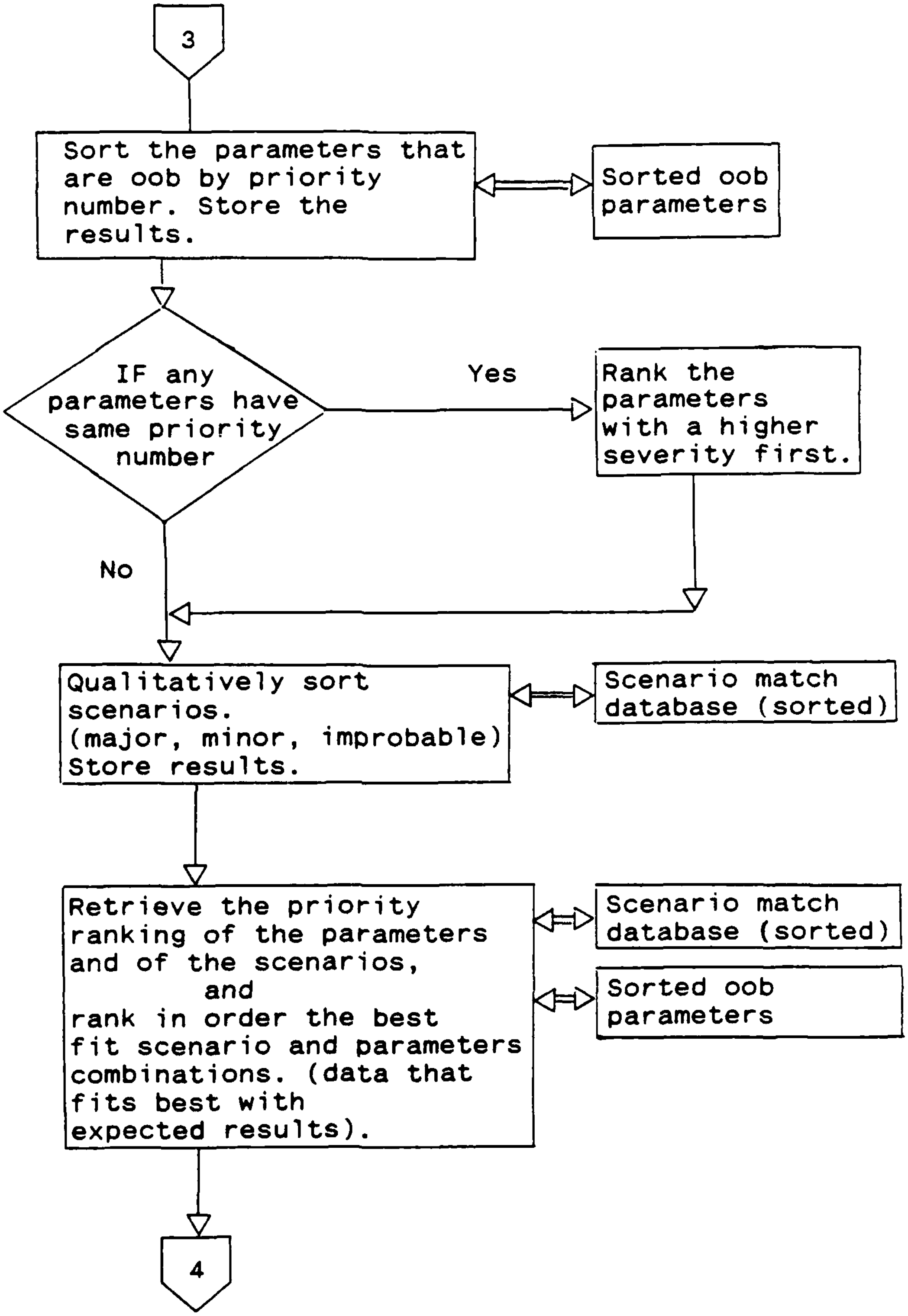

3
Sort the parameters that are oob by priority number. Store the results.
Sorted oob parameters
IF any parameters have same priority number
Yes
Rank the parameters with a higher severity first.
No
Qualitatively sort scenarios. (major, minor, improbable) Store results.
Scenario match database (sorted)
Retrieve the priority ranking of the parameters and of the scenarios, and rank in order the best fit scenario and parameters combinations. (data that fits best with expected results).
Scenario match database (sorted)
Sorted oob parameters
4

Fig. 3.1 continued

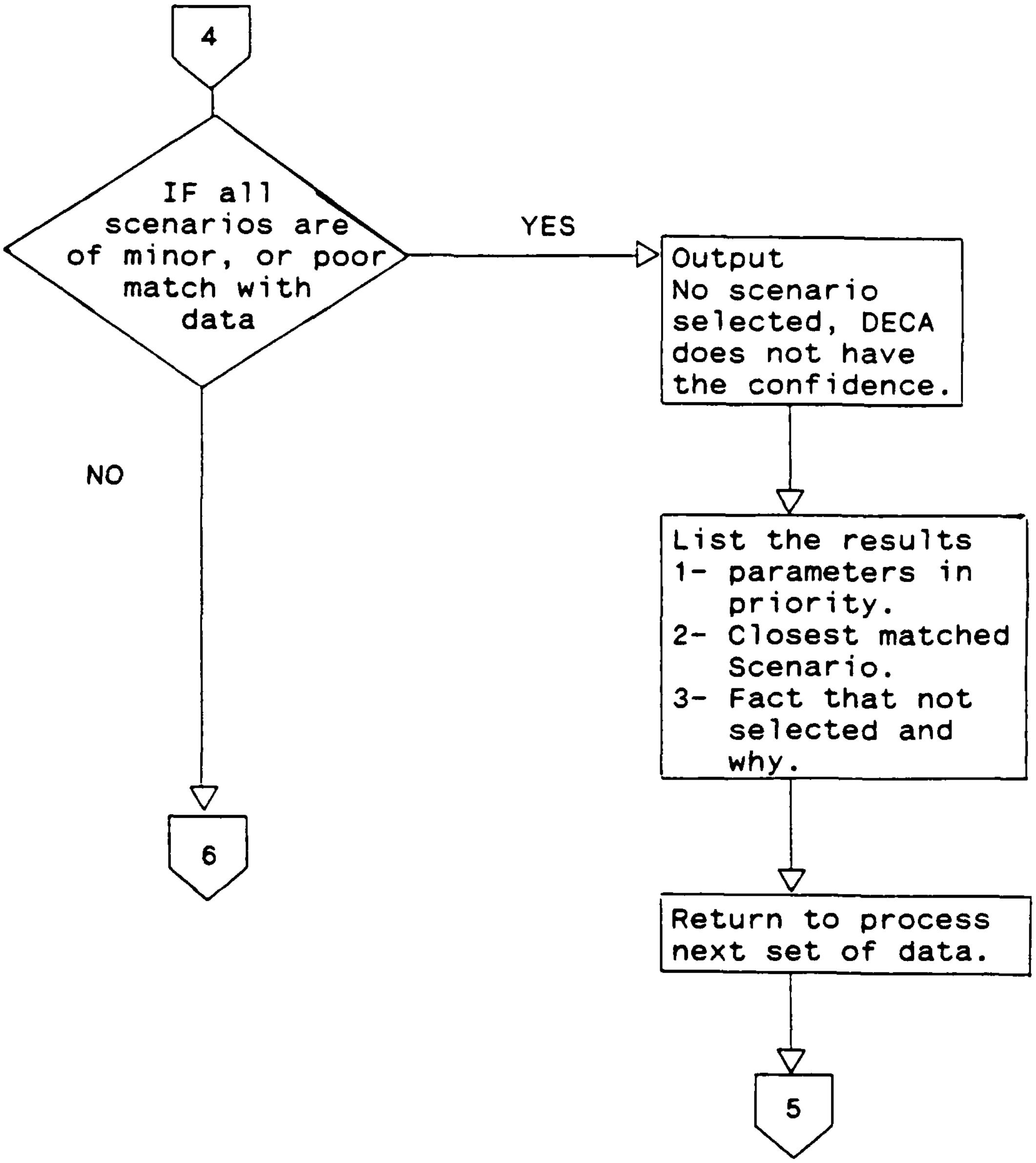

4
IF all scenarios are of minor, or poor match with data
YES
NO
Output No scenario selected, DECA does not have the confidence.
List the results
1- parameters in priority.
2- Closest matched Scenario.
3- Fact that not selected and why.
Return to process next set of data.
6
5

Fig. 3.1 continued

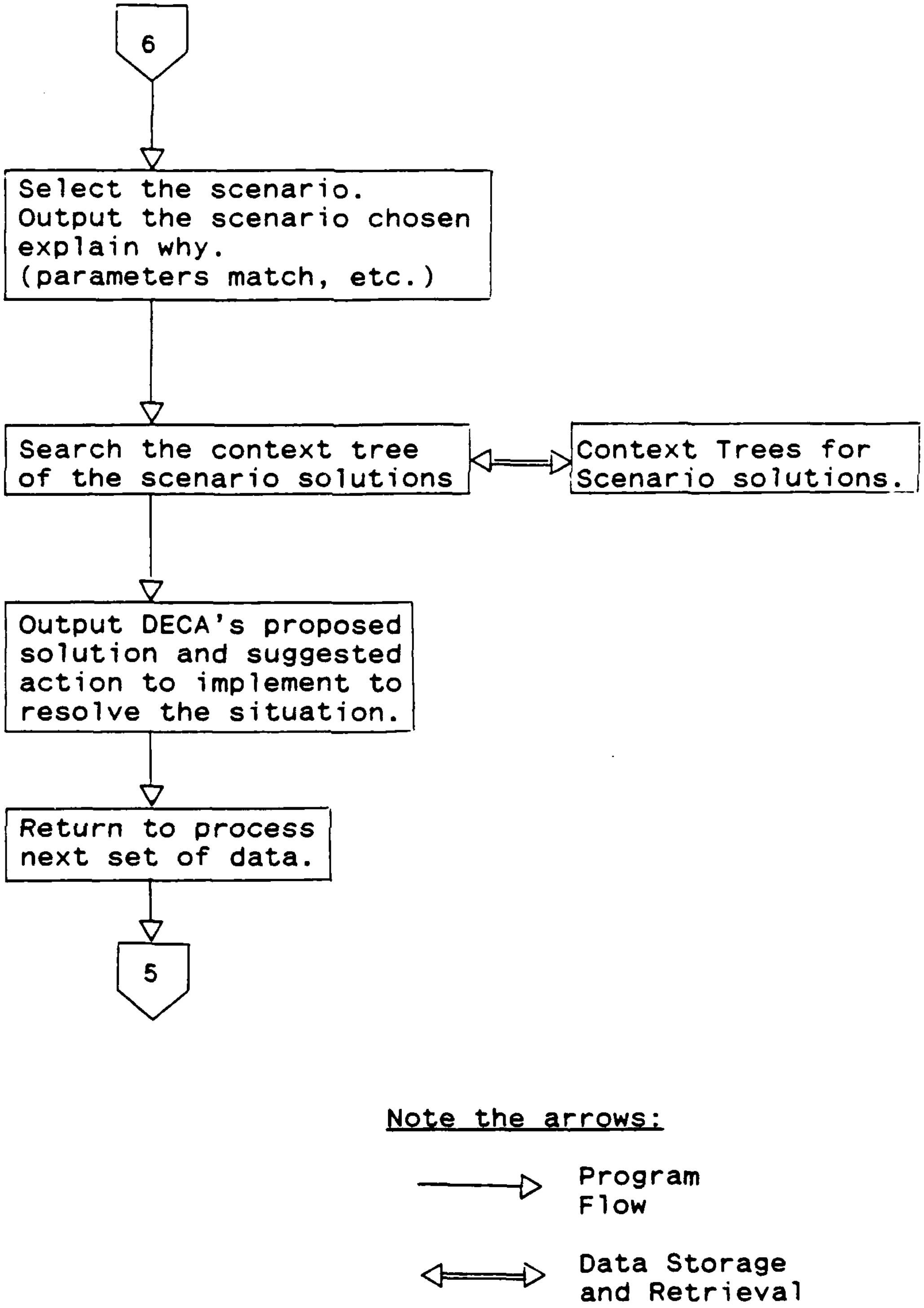


Fig. 3.1 continued

3.3 System Inputs

The input to the system is through on line sensors. For the example run of Three Mile Island Unit Two (TMI-2) there are nine parameters.

PZR-P	Pressurizer Pressure	psig
PZR-L	Pressurizer Level	inches
HL1-T	Hot Leg Temperature	deg F
CL1-T	Cold Leg Temperature	psig
SG1-P	Steam Generator #1 Pressure	inches
SG1-L	Steam Generator #1 Level	psig
SG2-P	Steam Generator #2 Pressure	psig
SG2-L	Steam Generator #2 Level	inches
QNT-P	Drain Tank Pressure	deg F

The data comes into the system in a set order known a priori. That is, we assume that the control processor gathers the data, checks the accuracy and validity of the data and sends out a stream of numbers. Since the data has been authenticated by the instrumentation, DECA assumes that the data is accurate through the instrumentation's use of fault detection and verification.

3.3.1 On-Line Information

As the sensor data is read into DECA, it is stored in an On-line Information database (OLI). Table 3.1 shows a sample of the OLI database for time period 0 to tx.

Table 3.1 On-line Information Database

ORDER	PZR-P	PZR-L	HL1-T	SG1-P	SG1-L	SG2-P	SG2-L	QNT-P	CL1-T
0	2145	218	607	944	123	3	930	116	559
15	2260	253	611	1022	79	6.3	1012	80	571
30	1905	182	587	998	26	7.8	987	30	577
45	1855	160	579	1000	17	9.3	993	20	576
60	1790	158	578	990	14	12	969	18	576
75	1760	162	577	1011	10	14.3	997	16	576
90	1725	175	578	1023	11	17.5	1005	16	577
105	1685	187	579	1021	11	19.6	1005	16	577
120	1650	200	579	1011	11	22.2	1000	16	579
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
tx			data for time x						

0, 15, 30,...,tx is the index for OLI. For our example, the values of tx will correlate to the time intervals during the TMI-2 accident. Even though we will only be concerned with the present record (or data sample for a given time), having the history will enable DECA to determine the relative speeds of the transients which will contribute information towards qualitatively deducing an urgency for various parameters.

3.3.2 Setpoint Database

Processing of information will begin once the OLI database has been established. First it takes a record t_n which corresponds to the data at the instant of time under investigation (usually the most recent), and checks each parameter against the setpoints for normal operation. For normal reactor operation the relational data base for setpoints (SDB) is as follows [Jow84, p. 5-13]:

Table 3.2 Setpoint Database (SDB)

VARIABLE	LLL	LL	L	N	H	HH	HHH	UNITS
PZR-P	1200	1900	2055	2150	2250	2355	2400	psig
PZR-L	45	150	200	222	240	260	280	inches
HL1-T	300	400	500	606	610	619	630	deg F
SG1-P	800	850	900	940	1050	1070	1105	psig
SG1-L	10	30	45	160	170	180	190	inches
QNT-P	1	2	2.5	3	35	80	122	psig
SG2-P	800	850	900	940	1050	1070	1105	psig
SG2-L	10	30	45	160	170	180	190	inches
CL1-T	300	400	500	558	610	619	630	deg F

The versatility of the system can be enhanced by utilizing DECA's ability to access different setpoint databases. When DECA looks at a setpoint database, it must check a system flag to determine where to look for the data. The system flag is set from an external source, such as data acquisition equipment or from the operator's console. The value of the flag reflects the present state of operation the system is in (e.g. normal, reactor shutdown, etc.). For example, the database shown above (Table 3.2) is for normal operating mode. If the reactor were to be shut down or reduce power output, there would be some transients which are normal to this action, but DECA would trigger alarms because some values are indicated as being out of bounds. With the flag set, DECA will be directed to look at a shutdown database, or it will be directed to call a subroutine which contains an analytical model to determine the appropriate setpoints for the present state of the system. This ability to switch between setpoint databases will alleviate the triggering of false alarms, thus letting DECA focus its attention on the problem.

3.4 Evaluating Sensor Data, Diagnostic-1

When processing a record of parameter values from the OLI database, the system will take the parameter data and compare it with its setpoint value. Retrieval of data in the SDB can be done, uniquely through the variable name as the key. By comparing the data against the setpoints, DECA determines whether it's in normal operating range or beyond its normal range. If it is normal, the parameter is left alone, but if out of bounds, DECA will determine the degree to which it is out (e.g. L, LLL, HH, etc.). This flag and corresponding thresholds indicate the severity associated with the system parameters. Also, DECA can incorporate fuzzy set theory into the system, and the degree to which each parameter is bound can be more accurately determined. For example, if a parameter has a severity of H (for High), then employing a blending function [Zadeh65, 86] can assign a value indicating the degree of membership. For example, 0.25 would indicate that it is only slightly over the High threshold. This degree of membership is loosely analogous to its "percentage of highness", but is an excellent way of quantifying the abstract.

There are several ways in which the comparison of parameter data and setpoint data can be retrieved and analyzed. This versatility is necessary to keep DECA in a generic format enabling it to be used in a variety of dynamic processes. As an illustration, for the first part of the monitoring, DECA takes the sensor data and compares it to the values in the setpoint database to determine if the parameters are out of bounds. The system uses two rules which would be applied to each of the system parameters.

Rule one: read the value of the parameter and compare it to the values for H and L in the SDB. If the parameter is greater than H or if the parameter is less than L tag the parameter as being out of bounds.

Rule two: if the parameter is flagged as being out of bounds, then compare it to the SDB values for LLL, LL, L, H, HH, HHH and determine which range they fall in. This range (e.g. LL) will be used as the severity of out of bounds.

In an algorithmic fashion, the rules would be as follows:

```
Rule 1: Read the value of the parameter := P1;
           Call procedure comparison;

                Procedure comparison;
                   Retrieve SDB;
                     if P1 > H and
                        P1 < L
                      out of bounds = true;
                   return;

Rule 2: Read the value of the parameter
        which is out of bounds := P1;
           Call procedure comparison;

                Procedure comparison;
                   Retrieve SDB;
                     If P1 is in range
                       LLL to L or
                       H to HHH
                   Severity := LLL, LL, L,
                               H, HH, HHH;
                   return;
```

Also, if most of the setpoint data was stored in a typical relational database, one could use SQL data retrieval language to access the data (see appendix A for detailed description of SQL). The rules necessary to compare the data are shown below in the SQL format. These rules would be applied to each of the monitored parameters of the system.

Rule to flag a parameter:

```
SELECT TIME, VARIABLE, PARAM
FROM  OLI
WHERE PARAM IN
   SELECT *
   FROM SDB
   GROUP BY VARIABLE
   HAVING  PARAM < L
      OR   PARAM > H
```

Rule to assign the severity:

```
CREATE TIME, VARIABLE, PARAM, SEVERITY
FROM   OLI
WHERE  PARAM   IN
   SELECT  P1, *
```

```
FROM    SDB
GROUP BY VARIABLE
HAVING  P1 < LLL
     CREATE P1, SEVERITY='LLL'
OR      P1 > LLL AND P1 < LL
     CREATE P1, SEVERITY='LL'
OR      P1 > LL  AND P1 < L
     CREATE P1, SEVERITY='L'
OR      P1 > H   AND P1 < HH
     CREATE P1, SEVERITY='H'
OR      P1 > HH  AND P1 < HHH
     CREATE P1, SEVERITY='HH'
OR      P1 > HHH
     CREATE P1, SEVERITY='HHH'
```

All the data values are retrieved from the setpoint database (SDB) and the on-line information database (OLI), and reference the current time frame the system is monitoring.

Taking a rule-based approach more analogous to a structured programming language, some of the comparison rules would look more like the following in the if..then..else format:

Rule: Take sensor value of parameter and retrieve SDB data for the parameter

```
IF DATA < L OR
    DATA > H
THEN
    FLAG PARAMETER
```

The above rule is applied to each parameter in the OLI data record. Then the following rule will be applied to only those parameters which are flagged.

```
Rule IF flagged THEN
          IF data <= L THEN
             IF data > LL THEN
                severity = L
             ELSEIF data > LLL THEN
                severity = LL
             ELSEIF data <= LLL THEN
                severity = LLL
             ENDIF
          ELSEIF data >= H THEN
             IF data < HH THEN
                severity = H
             ELSEIF data < HHH THEN
                severity = HH
             ELSEIF data >= HHH THEN
                severity = HHH
             ENDIF
          ENDIF
        ENDIF
```

In a LISP implementation the rules would be as follows:

```
(Defun Flag-Rule (parameter-looking-at)
  (Let ((data (value-of-data parameter-looking-at))
        (L    (value-of-SDB Low))
        (H    (value-of-SDB High)))
    (cond ((or (< data L)
               (> data H))
      (Flag-the-parameter parameter-looking-at)))))
```

The rule for determining severity:

```
(Defun Severity-Rule (data LLL LL L H HH HHH)
  (Cond ((<= data L)
         (Cond ((>  data LL)  (setq severity L))
               ((>  data LLL) (setq severity LL))
               ((<= data LLL) (setq severity LLL))))
        ((>= data H)
         (Cond ((<  data HH)  (setq severity H))
               ((<  data HHH) (setq severity HH))
               ((>= data HHH) (setq severity HHH))))))
```

This finishes the first phase of the DECA system. At this point, all the flags have been set for the out of bound parameters, and the severity of out of boundedness has been determined.

3.5 DECA's Knowledge Base

Next, the submodules must be defined (listed in appendix E). The submodules consist of the encoded knowledge of possible disaster scenarios which could occur. These submodules only represent what the "expert" has anticipated as possible disasters, and the completeness of the knowledge base depends greatly upon the knowledge of the expert, the thoroughness of the system design, and the completeness of the analysis and coding. In general, this will not be enough to cover everything possible and that is where the second key objective of DECA comes in. The second objective of DECA is to discriminate between the root cause and the side effects. Since these large systems have built in redundancies, usually there will be only one piece of equipment failing at a time. After DECA has ascertained what is really important, it can then point the operator in the right direction even though it has not found out what the exact cause of the problem is.

One important facet of DECA is its ability to direct the user toward the source of the problem if it can not figure out the exact cause of the problem. This feature is important for 2 reasons: 1) People can not think of all the possibilities of what might go wrong with a large system, and 2) The system will "ignore" the side effect alarms and give the operators the guidance so they can focus their attention on the problem. Also this direction will prevent the operators from being overburdened with "too much information". When DECA fails to clearly diagnose the malfunction in the system, it will list the parameters, their priorities, and where the operators should concentrate their efforts. This is something that was needed during the TMI-2 accident, for there were so many side-effect alarms going off, that the operators failed to notice the block valve was stuck open until after the damage was done to the reactor core.

In the example of TMI-2 (appendix E), there are only nine parameters monitored, so there are not that many scenario submodules defined, but enough to prove the validity of the system. Table 3.3 shows a list of scenarios and the parameters which would be affected for the TMI-2 accident. For example, the scenario 9 and 10 are affected by the parameters SG1-P, SG1-L, SG2-P, SG2-L, and CL1-T.

Table 3.3 Scenario - Parameter Relation Chart

#	Scenario	PARAMETERS								
		PZR-P	PZR-L	HL1-T	SG1-P	SG1-L	QNT-P	SG2-P	SG2-L	CL1-T
1	Pressurizer Leak	X	X		X	X		X	X	
2	Block Valve Leak	X	X		X	X	X	X	X	
3	Pipe Rupture (Drain)	X	X				X			
4	Drain Tank	X	X				X			
5	Pipe Rupture PCS hot				X	X		X	X	X
6	Pipe Rupture PCS cold			X	X	X		X	X	
7	Reactor Pump			X	X			X		X
8	Steam Generator PCS			X	X	X		X	X	X
9	Steam Generator SCS				X	X		X	X	X
10	Pipe Rupture SCS				X	X		X	X	X
11	SCS Feedwater Pump				X	X		X	X	X
12	SCS Turbine Trip				X	X		X	X	X

In a format more conducive to list processing (Lisp) we arrange the parameters in the following manner:

Table 3.4 Parameter - Scenario List

PARAMETER	POSSIBLE SCENARIO
PZR-P	1 2 3 4
PZR-L	1 2 3 4
HL1-T	6 7 8
SG1-P	1 2 5 6 7 8 9 10 11 12
SG1-L	1 2 5 6 8 9 10 11 12
QNT-P	2 3 4
SG2-P	1 2 5 6 7 8 9 10 11 12
SG2-L	1 2 5 6 8 9 10 11 12
CL1-T	5 7 8 9 10 11 12

These are the submodules which will have to be searched via the lookahead capability (see appendix E) of DECA because of the out of bounds condition in the parameters. The object is to see how closely actual data matches the expectations and draw conclusions from these correlations.

Chapter 4

Inference Engine

The previous chapter explains the knowledge base structure and the mechanism to assign severity. The possible scenarios (disasters) have also been defined. This chapter deals with the decision mechanism and parameter usage in DECA.

4.1 Lookahead Mechanism and Scenario Evaluation

Consider, for example, an instance of the parameter PZR-P is 1180 psig. Through the first stage DECA will flag PZR-P and give it a severity of LLL (qualitatively translated as very very low). Next PZR-P is checked to see which submodule (context tree) should be searched. From the table it indicates the scenarios 1, 2, 3, and 4 (i.e. pressurizer leak, pressurizer block valve, pipe rupture in the drain line, and the drain tank). This method is similar to the MYCIN Lookahead mechanism (Findout and Monitor; see appendix B). What DECA does before it reaches a conclusion (e.g. that there is a pressurizer leak), is it will look ahead at these possible scenarios and determine if the criterion is met for each possible scenario to occur. The scenario which most closely fits the data will be the one chosen by DECA as the disaster. If there isn't a close enough match, then DECA will predict probable cause(s) of malfunction, the parameter priority, and suggest items or subsystems to be considered more closely. In the TMI-2 example, with PZR-P out of bounds, the lookup mechanism will indicate that scenario #1 (pressurizer leak) is a possibility, but the following parameters will also have to be out of bounds: PZR-P, SG1-P, SG1-L, SG2-P, SG2-L in order to have a strong likelihood of occuring.

4.2 Solution Search

Next DECA executes the rules to check the severity of the parameters to see how closely they match with the context trees (see appendix F). If there is a good match between the scenarios, expected data, and parameter criticality, then a prompt would appear on the screen:

```
Scenario selected is;
Scenario Number 8
Scenario Description: Steam Generator -
                      Primary Coolant System
Confidence  5/6
```

If there is more than one plausible scenario considered, DECA would list them in rank order from highest to lowest likelihood similar to this example:

```
Scenarios that were considered as possible choices but not
selected are:

Scenario Ratio Description

    8     5/6   Steam Generator - Primary Coolant System
    5     4/5   Pipe Rupture    - Hot Leg,  Primary Coolant
    1     2/3   Pressurizer Leak
    12    3/5   Turbine Trip    - Secondary Coolant System
    11    3/5   Feedwater Pump  - Secondary Coolant System
    10    3/5   Pipe Rupture    - Secondary Coolant System
    2     4/7   Block Valve Leak
    6     2/5   Pipe Rupture     - Cold Leg, Primary Coolant
    4     1/3   Drain Tank
    3     1/3   Pipe Rupture     - (drain tank)
```

If the system could not make up its mind, then it would also list the out of bounds parameters on the screen in order of highest to lowest priority.

```
 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-L    10
   QNT-P    10
   PZR-P    10
   SG1-L    9.3
   SG2-L    9.3
   CL1-T    8.6
   SG1-P    8.5

Scenarios that were considered as possible choices but not
selected are:

Scenario Ratio   Description

    8     2/3    Steam Generator - Primary Coolant System
    11    3/5    Feedwater Pump - Secondary Coolant System
    10    3/5    Pipe Rupture  - Secondary Coolant System
    5     3/5    Pipe Rupture  - Hot Leg,  Primary Coolant
    1     1/2    Pressurizer Leak
    2     3/7    Block Valve Leak
    12    2/5    Turbine Trip  - Secondary Coolant System
    6     2/5    Pipe Rupture  - Cold Leg, Primary Coolant
    4     1/3    Drain Tank
    3     1/3    Pipe Rupture  - (drain tank)
```

This way, even if the system fails to generate a solution, it will be able to direct the operator to the source of the trouble.

4.3 Context Trees and Scenario Ranking

If we take scenario number one (Table 3.3), the pressurizer leak from the lookahead database, it shows that the following parameters should not be in normal operating mode: PZR-P, PZR-L, SG1-P, SG1-L, SG2-P, SG2-L. The context tree (discussion given in appendix E) contains rules to check the parameters (i.e. High (H) or Low (L)) as well as the severity to see how well the present data fits the scenario.

For the pressurizer leak, encoded into the context trees are the rules to match the data with the anticipated data of the scenario. From this matching operation the mechanism will determine a qualitative value of the match. DECA gives three levels of matching: High, Medium, and Low. These three levels will give DECA a guide for further consideration of the scenario it is evaluating. If there is a high match, DECA will give it major consideration; a medium match will get a minor consideration, and for a low match the scenario will probably not get considered. Figure 4.1 summarizes the pattern matching process.

```
                PRESSURIZER LEAK
                     / ¦ \
                    /  ¦  \
                   /   ¦   \
          rules and patterns to match with data

              High Match     - Major Concern
              Moderate Match - Minor Concern
              Low Match      - Improbable
```

Figure 4.1 Qualitative Match for Scenarios

4.4 Parameter Prioritization

At the same time, the system will look at the problem from a parameter's viewpoint. This perspective is as follows: the parameter (e.g. PZR-P) gets the following data after checking the database for possible scenarios (e.g. 1, 2, 3, 4). DECA then looks at the scenario ratio match data to see how well each scenario correlates with the out of bounds parameters. The better the match of the other parameters present with these scenarios, the more likely one of these scenarios is occurring in the process. An higher likelihood for a given scenario will increase the parameter importance. DECA then assigns a greater likelihood of each of these scenarios as of being present. For the parameter PZR-P, the Lookahead mechanism points to scenarios 1, 2, 3, and 4 (Table 4.1).

Table 4.1 Parameter Priority Database for PZR-P

PZR-P					
Expectation Match				Parameter Priority	Rank
1	2	3	4	HIGHEST	10
	2	3	4		8.5
1		3	4		8.5
1	2	3			6
1	2		4		6
1		3			4
1			4		4
	2	3			4
	2		4		4
		3			2.5
			4		2.5
	2			LOWEST	1
1					1

This analysis will be performed for each flagged parameter and have a ranking of parameter priorities. In the case of one or more parameters having the same ranking, the severity (e.g. H, LL, etc.) will be used to determine the relative differences in importance.

4.5 Determining a Solution

In a similar fashion now that there is a list of parameter priorities (which will be displayed to the operator soon), a priority ranking for each possible scenario is given. Earlier, it was quantitatively determined how critical the scenario is (i.e. severity). Now DECA refines the ranking of the scenarios to find the most critical problem. For our example with the parameter PZR-P, the scenarios to look at (S1, S2, S3, and S4) have already been determined. Also parameter rank for all the out of setpoint bounds parameters have been determined. Now, search a context tree. Table 4.2 shows the different combinations of the context tree for PZR-P.

DECA then takes the priority rank and the scenario which matches the closest with the system data, and outputs what it thinks is the most likely scenario and a list of the parameters in rank order. Also, the output will have a recommendation of how to attack the situation at hand. This output will meet several objectives, they are: 1) the ability to take vast amount of information of the system state and keep only relevant data, 2) Figure out the most likely cause of what is wrong with the system, and 3) Give a priority of the parameters so the technicians can concentrate the root of the problem rather than treat the symptoms.

Another aspect of the system which will help in its deployment in the field is the way the operators can modify the knowledge. In the debug phase for a new application, DECA can also display the a priori information for ranking of the parameters, the combinations of scenarios, the submodule data, and expectations of qualitative knowledge, as well as all On Line Databases so the operator can be consulted and initiate refinement of the data. This feature can be thought of as "Off-Line Configuration", or "Off-Line Learning".

Table 4.2 Combining Parameter and Scenario Priorities

Context:	Most Critical Parameter
Lookahead:	S1, S2, S3, S4

Parameter: Criticality w.r.t. the analysis data

highest	1	(PZR-P)
¦	2	(PZR-L)
¦	3	(SG1-P)
¦	4	(SG2-P)
¦	5	(SG1-L)
V	6	(SG2-L)
lowest	7	(QNT-P)

Expectation of Scenario rank

(S1 V S2) ^ (S3 V S4)	Critical	Major
S1 ^ S2		Major
S1 ^ S3		Minor
S1 ^ S4		Minor
S2 ^ S3		Minor
S2 ^ S4		Minor
S3 ^ S4		Improbable

Chapter 5

Implementation of the DECA System

The DECA system has been successfully implemented on a Symbolics 3670 Lisp Machine using the Symbolics Common Lisp Language and its object oriented extension Flavors. It also has been tested using real data from the TMI-2 accident.

5.1 Selection of Lisp as the Programming Language

Lisp has been chosen as a language for implementation for several reasons. First, the comprehensive set of tools to work with make it ideally suited for rapid prototyping. If a change is needed, it is quite easily implemented. Second is the modular structure of the Lisp language. Being able to set up many functions enables the system to be broken up into functional parts, this in turn also helps in the software maintenance.

Another aspect which is very important to the DECA application is Lisp's ability to evaluate both numbers and symbols. The symbolic processing feature enables the machine to run in a manner similar to the way humans think, with symbols. DECA is able to use symbols as keywords in its reasoning. For example, the setpoint databases contain an item which is used to indicate what the mode of operation is (e.g. normal). It can use symbolic data as easily as it can use numeric data, thus the system can be designed more closely to the way humans think. The symbolic design facilitates the encoding of knowledge from the system experts and the debugging of the application DECA is being applied to.

Compared to other structured languages, Lisp has another significant advantage, dynamic data structures. With most languages, the programmer must set aside the precise amount of space for every conceivable data structure which the application may come across. This can be cumbersome and tends to make the system inflexible. Lisp on the other hand does not require this. Instead, if it needs more space, it will dynamically allocate it. Now the operator and programmers do not have to think of every possible situation, and if the machine comes across with something new it can just add it to the lists. The dynamic data structures is one of the primary reasons why Lisp is used for rapid prototyping applications. After the system has been thoroughly tested, it then can be translated to a language such as C, which interfaces with hardware better, if necessary.

The object oriented extension, Flavors [Weinr80], was incorporated into DECA. Flavors help keep track of the data and can be organized into objects. For example, all the setpoint values for a variable (e.g. PZR-P) were organized into an object. Flavors arranges the data into a well organized structure which contain the interrelationships between the pieces of data. This organization helps facilitate the rapid prototyping, and dynamic databases.

5.2 Speed Considerations

The purpose of the DECA system is to be able to monitor large systems in real time. Unless it can do all the calculations in the time between sensor data readings, the system will be of little use. For the applications being looked at in the thesis; chemical process control, nuclear reactor control, space systems telemetry, and flight control systems; DECA will need to have all calculations completed in a time period of 5 - 10 seconds for the chemical and nuclear processes, and 10 - 100 msec. for the flight controls. The differential in time is due, in part, to the nature of the implementation purpose. For a chemical process, DECA will be more of a supervisor/advisor for the system operators and humans will not react much faster than the 5 - 10 seconds. While for the flight control systems, DECA will be an automated system, initiating all of its own conclusions.

To help the system meet its processing time constraints, it must be deployed with fast hardware, for example the Symbolics computers, Lisp on a chip microprocessors (e.g. Symbolics Ivory, TI Explorer Chip), or 32-bit high speed microprocessors (e.g. Intel 80386, Motorola 68030).

Another consideration for speed is the implementation language. For chemical or nuclear processes (slow), Lisp will generally work. While for a high speed dynamical process (e.g. flight controls), specialized Lisp hardware or a language such as C may be necessary to use, for they are optimized for high speed execution.

Furthermore, the amount and type of sorting performed on the data should be carefully controlled. There should not be any more sorting than necessary, and the type of algorithm to perform the sorting must be carefully selected. For DECA, sorting is done only when a ranking is needed, and it employs a modified quicksort routine.

A final consideration for execution speed is with the data structure used. To keep calculations to a minimum, data should be kept to a minimum. Lisp and its dynamic data lists also help increase execution speed due to the fact that for each cycle it only searches data structures as large as the data contained in it.

5.3 Computer Input/Output

Computer input/output (i/o) must also be addressed judiciously. In general, terminal and disk i/o can lead to system bottlenecks since they typically reduced throughput compared to the processor.

When designing and evaluating the system, the i/o must be taken into consideration as part of the system computational requirement. In general it means allowing extra time to load up the data files containing the knowledge of the dynamic process (e.g. setpoints, scenarios) as well as the time to load in new system data (e.g. new setpoints for different operating state, sensor data). At the other end of the process is the i/o to the terminal screen and/or writing the data to disk. When writing to the screen, the system is constrained for time in two ways; first, the speed which the terminal runs is usually the slowest of any part of the computer system. Second, any information which must be absorbed by the operator cannot leave the screen until the operator signals to. So if there is more than one full screen of data, there will be a tremendous amount of idle cpu time while the computer waits for the operator to digest the information.

If DECA is used with a dynamic process where the operator is advised by DECA such as a nuclear plant, then the i/o becomes the major bottleneck to DECA's performance. On the other hand, when DECA is employed in a completely autonomous fashion, such as a space vehicle controller, then the terminal i/o is not employed, and the disk i/o requirement will probably be minimal, but the time between sensor reports will be at least 1000 times smaller, thus raw cpu speed is the major factor. Chapter 6 shows how the processing times differ with different i/o loads.

5.4 Dynamic Databases

As mentioned earlier, Lisp makes it very easy to implement dynamic databases. This is because the basic Lisp form, the list, can be modified quickly and in many different ways. It can also contain both numeric and symbolic data, and functions can easily manipulate them.

The DECA system took advantage of Lisp's list manipulation abilities. For example, when processing a run, DECA will search many lists for the appropriate data which it may reference or it may add new data to the list. One way it accomplishes this is by incorporating the setf function into the code. The setf is a function in lisp which will retrieve the part of a list which matches the structure given (first argument) with the new structure encountered (second argument). Functions similar to the setf function increase the speed of data manipulation tremendously. Also, they make the coding easier than it would be using other languages, since one does not have to worry where the data is stored aside from the name of the list. In C, Pascal, or Ada, there will be a large effort just to control the data, while Lisp will let one use the data.

5.5 Separation of the Knowledge Base and Inference Engine

During the development of the DECA system, great care was taken to make sure that the knowledge base and the inference engine were kept separate. There are several reasons why the two major parts of the Expert System should be separate.

First, it will enhance software maintainability. The inference engine will exist in several modules. Thus if one wants to change some function of DECA, just access the module, make the change, and recompile. To update the knowledge about the process, then only the data files need be updated. Overall, the separate modular format enables, the users to easily access any part, as well as keep everything organized. If the knowledge and data were threaded together in the same code, it would be nearly impossible to maintain and update the system for a large application.

Secondly, it makes the DECA system portable. That is, DECA has been designed to be used in all dynamical processes. For its demonstration of feasibility in this thesis, DECA was applied to the Three Mile Island accident. If one wants to apply it to monitor some other process, new data files would have to be written which contain the knowledge of the process to be controlled.

Finally, the separation kept the development of the inference engine generic. More specifically, when coding the inference engine there was not any influences on the software design attributed to any particular application. The structure was designed for use with any dynamical process.

5.6 DECA Architecture Planning

When implementing DECA, great efforts were taken to fully develop the design of the system architecture before any code was written. It was important to make sure that the whole process was carefully planned out. Some standard steps should be taken whenever any software is being developed. They are:

-Carefully research the topic cᶠ the application, and develop the problem thoroughly.

-Define the process flow. Developing a flowchart will help visualize what is occurring in the system.

-Use the flowchart to develop the code. Breaking the code into modules according to function will facilitate software debugging and maintenance.

-Employ a top down approach to the system design, and bottom up approach for coding.

Other features which were incorporated into DECA were software interfaces to outside models and ability to use data and databases from multiple sources.

The incorporation of interfaces will increase the usefulness of the software for it will enable DECA to use other information to arrive at its decisions. For example, in the TMI-2 example, a single setpoint database was used. DECA has the ability to use multiple setpoint databases located on several computers. In TMI 2 the setpoint was labeled normal, there could have been another setpoint database for a shutdown mode. This shutdown mode database could have been on another computer, and DECA's setpoint-data-list would direct DECA to the other computer. Also, the databases do not have to be an array of numbers, it could be an analytical model which calculates the setpoints.

Another useful capability is to hook into simulation models. As an example, DECA could call on a computer simulation to validate its conclusions before it makes a recommendation.

The ability to access other information makes DECA more useful. DECA adds more capability to the system monitoring without losing the benefits of the past work done in the area. It could be considered analogous to computer hardware being upward compatible; older software can be used on new and improved hardware.

Chapter 6

Results and Conclusions

In this chapter, the runtime results of the DECA system will be discussed. DECA was evaluated using the real time data from the Three Mile Island accident. Appendix E contains the data used for the knowledge base data files.

6.1 Test Runs

Before the system was executed using TMI-2 sensor data, DECA was debugged using fictitious data which would test the extremes which DECA might encounter during a real application. The following data was used for the test run for fictitious times of 5, 10, and 15 seconds.

```
((05 (2150 222 606 940 160 3   940 160 558))
 (10 (2260 270 606 870  42 1.9 910  42 635))
 (15 (2380 282 610 870  29 0.9 860  42 635)))
```

The sublist associated with each system time value contains the readings of each of the nine parameters in the TMI-2 example. The parameters are always read into DECA in the same order. The order for TMI-2 was PZR-P, PZR-L, HL1-T, SG1-P, SG1-L, QNT-P, SG2-P, SG2-L, and CL1-T.

The above data was used to test DECA's ability to work in the middle of the road and at the two extremes. Time 05 was used to test DECA when none of the parameters were out of bounds. Looking at the data output file, appendix D, it is apparent that DECA just skimmed through its routines without doing anything since everything was alright.

Time step 10 represents what could be thought of as a typical load to DECA, that is several parameters it is monitoring are out of bounds. The output file in appendix D shows the intermediate values as DECA is running. For a conclusion, DECA was not confident enough with the data to decide on a scenario, so it just listed out the parameters, their ranks, the scenarios and their ratios which it had considered. DECA's conclusions are shown below.

```
DECA's conclusions for system time: 10
 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-L     10
   QNT-P     10
   PZR-P     10
   SG1-L     9.3
   SG2-L     9.3
   CL1-T     8.6
   SG1-P     8.5

 Scenarios that were considered as possible choices but not
 selected are:

Scenario Ratio Description

   8      2/3   Steam Generator - Primary Coolant System
   11     3/5   Feedwater Pump - Secondary Coolant System
   10     3/5   Pipe Rupture - Secondary Coolant System
   5      3/5   Pipe Rupture - Hot Leg,  Primary Coolant
   1      1/2   Pressurizer Leak
   2      3/7   Block Valve Leak
   12     2/5   Turbine Trip - Secondary Coolant System
   6      2/5   Pipe Rupture - Cold Leg, Primary Coolant
   4      1/3   Drain Tank
   3      1/3   Pipe Rupture - (drain tank)

End of data evaluation for system time: 10
```

Time step 15 is an example of the sensor data correlating well with DECA's knowledge. Thus it is confident enough to select a scenario as likely occurring in the system at that time. For this data, it has decided that the problem is occurring in the steam generator on the primary coolant side. The conclusion is shown below (extracted from the runtime output file appendix D).

```
DECA's conclusions for system time: 15

 Scenario selected is;
 Scenario Number 8
 Scenario Description (Steam Generator - Primary Coolant
System )
 Confidence  5/6

 The parameter and priorities are as follows:

   PZR-L    10
   QNT-P    10
   PZR-P    10
   SG1-L    9.3
   SG2-L    9.3
   CL1-T    8.6
   SG1-P    8.5
   SG2-P    8.5

 Scenarios that were considered as possible choices but not
 selected are:

Scenario Ratio  Description

  8       5/6     Steam Generator - Primary Coolant System
  5       4/5     Pipe Rupture - Hot Leg,  Primary Coolant
  1       2/3     Pressurizer Leak
  12      3/5     Turbine Trip   - Secondary Coolant System
  11      3/5     Feedwater Pump - Secondary Coolant System
  10      3/5     Pipe Rupture   - Secondary Coolant System
  2       4/7     Block Valve Leak
  6       2/5     Pipe Rupture - Cold Leg, Primary Coolant
  4       1/3     Drain Tank
  3       1/3     Pipe Rupture - (drain tank)

End of data evaluation for system time: 15
```

The output is useful in several ways. First, it organizes the data for the operators. It also, displays it in rank order. Displaying in rank order will avoid giving the operator information overload for they can just look at the top of the list and see what is most important. Finally, DECA also lists the scenarios which it considered. Looking at the scenarios considered and seeing the ranks of the parameters, the operators can use their system knowledge to assess the problem. In time step 15, they would probably concentrate on the Primary Coolant System piping, since it was considered most often by DECA.

From DECA's output, it can be seen that the system has met several of its design goals, they are:

-It identifies and ranks the sensor data parameters in order of importance.

-DECA searches for the scenario which is most likely occurring and selects one, only if it is confident enough in its data correlation.

-It lists all the relevant scenarios which it considered.

-DECA gives a summary which the operators can easily understand the information in it and know where they must concentrate their efforts.

6.2 Runtime Log

Since there is no relatively easy way to check the system operation during execution (e.g. MYCIN has an interface which the user can ask questions), a runtime log was created. The runtime log consists of intermediate variables written to disk during DECA's execution. After DECA runs through a module of code, it writes out the values to disk. As an example, below is the listing of the log for the variables from the test run data for time step 10.

```
Intermediate parameters for system time: 10

Sensor-record  (2260 270 606 870 42 1.9 910 42 635)
Oob-parameters (PZR-P PZR-L SG1-P SG1-L QNT-P SG2-L CL1-T)
Oob-parameters-values (2260 270 870 42 1.9 42 635)
Oob-severity (H HH L L LL L HHH)

Lookahead-scenarios  (1 2 3 4 5 6 7 8 9 10 11 12)

Scenario-data-match-list

((1 (SG2-L SG1-L SG1-P)) (2 (SG2-L SG1-L SG1-P))
 (3 (QNT-P)) (4 (PZR-L)) (5 (SG2-L SG1-L SG1-P))
 (6 (SG2-L SG1-L)) (7 (CL1-T))
 (8 (CL1-T SG2-L SG1-L SG1-P))  (9 (CL1-T))
 (10 (CL1-T SG2-L SG1-L)) (11 (CL1-T SG2-L SG1-L))
 (12 (CL1-T SG1-P)))

Parameters-per-scenario-expect
((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5)
 (10 5) (11 5) (12 5))

Scenario-ratio-match-list
((1 1/2 MINOR) (2 3/7 MINOR) (3 1/3 MINOR) (4 1/3 MINOR)
 (5 3/5 MINOR) (6 2/5 MINOR) (7 1/4 IMPROBABLE)
 (8 2/3 MINOR) (9 1/5 IMPROBABLE) (10 3/5 MINOR)
 (11 3/5 MINOR) (12 2/5 MINOR))

Scenario-major NIL
Scenario-minor ((8 2/3) (11 3/5) (10 3/5) (5 3/5) (1 1/2)
                (2 3/7) (12 2/5) (6 2/5) (4 1/3) (3 1/3))
Scenario-improbable ((7 1/4) (9 1/5))
```

```
Parameter-ratio-match
((PZR-P ((4) NIL (4 3 2 1) NIL))
 (PZR-L ((4) NIL (4 3 2 1) NIL))
 (HL1-T NIL)
 (SG1-P ((10) NIL (12 11 10 8 6 5 2 1) NIL))
 (SG1-L ((9) NIL (12 11 10 8 6 5 2 1) NIL))
 (QNT-P ((3) NIL (4 3 2) NIL))
 (SG2-P NIL)
 (SG2-L ((9) NIL (12 11 10 8 6 5 2 1) NIL))
 (CL1-T ((7) NIL (12 11 10 8 5) NIL)))

Parameter-rank-list
((QNT-P 10) (PZR-L 10) (PZR-P 10) (SG2-L 9.3) (SG1-L 9.3)
 (CL1-T 8.6) (SG1-P 8.5))

Parameter-rank-list
((PZR-P PZR-L QNT-P 10) (SG1-L SG2-L 9.3)
 (CL1-T 8.6) (SG1-P 8.5))

Parameter-rank-list
((PZR-L 1C) (QNT-P 10) (PZR-P 10) (SG1-L 9.3) (SG2-L 9.3)
 (CL1-T 8.6) (SG1-P 8.5))

Possible-scenarios-for-situation
((8 2/3) (11 3/5) (10 3/5) (5 3/5) (1 1/2) (2 3/7) (12 2/5)
 (6 2/5) (4 1/3) (3 1/3))

End of variable log.
```

The parameters' values are created while DECA is executing a particular function. Table 6.1 contains lists of the function names and the variables which were modified upon execution of the function. See appendix F for the complete listing of all of DECA's functions.

Table 6.1 Function Variable References

Function Name (in capitals)	Variables Modified
COMPARE-SENSOR-DATA	
	oob-parameters
	oob-parameters-values
	oob-severity
GET-SCENARIOS	
	lookahead scenarios
MATCH-SCENARIO-TENDENCY	
	scenario-data-match-list
MAKE-LIST-OF-NUM-PARAMS-EXPECT	
	parameters-per-scenario-expect
SCENARIO-QUAL-MATCH	
	scenario-ratio-match-list
SPLIT-INTO-MAJ-MINOR	
	scenario-major
	scenario-minor
	scenario-improbable
MAKE-PARAMETER-COMPARISON	
	parameter-ratio-match
	parameter-rank-list
REFINE-PARAMETER-RANK-TOP	
	parameter-rank-list
ORDER-MULTIPLES	
	parameter-rank-list
PUT-SCENARIOS-TOGETHER	
	possible-scenario-for-situation

6.3 Experimentation of DECA with TMI-2 Data

After the evaluation of the test runs, it was determined that DECA appeared to be working according to design.

To show DECA's efficacy, the system was run with data from the TMI-2 accident. See appendix E for the sensor readings data. The run consisted of nine time steps at 0, 15, 30, 45, 60, 75, 90, 105, and 120 seconds after the turbine trip occurred in the reactor.

DECA completed the run without a problem. The log and conclusions for each time step are given in appendix D. From those results, it can be seen that DECA consistently directed the operators to scrutinize the subsystem where the pressurizer, block valve, and drain tank are located in the reactor. This is precisely where the problem was. The stuck block valve in the pressurizer was allowing the reactor coolant to drain out of the system. For this run, DECA was only monitoring nine parameters, thus it did not have the fine resolution to extract the intricate nuances present in the system. Since the block valve, drain tank, and pressurizer directly affect each other, having DECA select these three problems continuously confirms its ability to determine the area of most importance.

Also, it should be noted that the knowledge base data for the scenarios, their tendencies, the parameter tendencies and lookahead scenarios (that is all except the setpoints and sensor data) were derived at with only the author's engineering experience and did not utilize anyone's nuclear reactor expertise. Thus having such promising results from DECA, continuously directing the operators' attention to the part of the reactor where the block valve is located, demonstrates the efficacy to the methodology of qualitative reasoning for monitoring dynamic processes. Referring to the runtime output file (appendix D), we see consistently the pressurizer pressure and level (PZR-P, PZR-L) are among the most important parameters (i.e. highest priority). These two sensors are located adjacent to the block valve.

6.4 Computational Requirements

Having the qualitative reasoning approach working meets one of the criteria of DECA, but the system is not very useful unless it can meet the real time processing requirements.

At present, DECA is in a prototype stage. That is, its primary purpose is be able to monitor a system and advise its operators, and to perform its task in a time limit approaching the real time constraints. In actuality, the DECA system is able to perform its task with impressive real time capabilities. The execution times vary according to input/output (i/o) load and are explained below.

To get accurate results for the execution times of the

DECA system, Lisp's "time" function was used. The time function accurately keeps track of elapsed time, time spent 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 where the bottlenecks occur during program execution. Note, all times are in physical seconds.

The first test was a single run. The knowledge base data files were all loaded, and then DECA evaluated the sensor data for a single time step. DECA then displayed its results on the computer terminal. The time required for this was 14.6732 seconds.

The second test consisted of reading in the knowledge base data files, evaluate a single time step, and not output any information to the terminal. For this run, 6.1779 seconds were required.

From the first two runs, it is determined that DECA takes about 8.5 seconds for terminal i/o for a small sized application. The runs were repeated several times, each run yielding consistent results due to the fact that the Symbolics is a single user system and one does not have to wait for other jobs unlike a timesharing system.

The third test run consisted of a single time step evaluation without any i/o to disk or terminal. This test is evaluating raw computing power of the system. From the run, the time to process the data was 0.120383 seconds, or 8.3 time steps could be evaluated per second. After the third test run, it was determined that the computer requires about 5.9 seconds to load the TMI-2 data files from disk.

It is obvious that for a more complex process these times will increase, but the time of 0.12 seconds to process the data which DECA is monitoring is well within the design goals of 5-10 seconds for the prototype. Also, the time spent reading in the knowledge base may be eliminated if it is loaded into active memory prior to DECA's operation. That would alleviate a large portion of the overhead and make DECA practical for some processes requiring a little faster turnaround (e.g. 0.5 to 1.0 seconds).

In the final test run, DECA was tested with all nine time steps. It consisted of loading in the knowledge base, sensor data, evaluating each time step, and writing to disk the variable log and DECA's conclusions. For this, DECA required 13.8393 seconds. The time interval between sensor readings was 15 seconds or 135 seconds for all nine readings. Since DECA did the calculations in 13.8 seconds, the system is more than adequate in terms of meeting computational requirements. This extra time will be eaten up when DECA is run with applications consisting of many more parameters and scenarios.

6.5 Conclusions

In most of the physical systems, reasoning is based on qualitative premises. For example, when a mechanic fixes a car, he will not hook up a vast array of sensors, devise mathematical models of the car and employ optimization techniques to simulate and determine what is wrong with it. The garage has neither the time nor the money to do it. Instead, the mechanic will use his experience and qualitative reasoning to determine and fix the problems afflicting the automobile.

In large dynamic systems, financial resources may be adequate, but time will be a constraint to monitor and evaluate the system in real time, using complex analytical models and global optimization techniques. DECA typically can be applied to such scenarios. It employs qualitative reasoning to narrow down possibilities for real time monitoring and diagnosis of dynamical processes.

An implementation of the DECA system was successfully developed in Lisp on a Symbolics artificial intelligence computer. Using the Three Mile Island Unit 2 Accident as a real world application, it was shown that DECA is able to monitor some dynamic processes in real time.

Another accomplishment derived from this research is the development of a comprehensive software architecture for diagnosis and evaluation of any dynamical process. This architecture was based on past work of Milne [Milne87] and offers flexibility in the use of knowledge in a diagnostic expert system.

One more important point of interest is in the development of the knowledge base. In dynamical systems, most often the data is dynamic hence to incorporate it as a part of the expert system's knowledge base may not be useful from a computational point of view. DECA uses a relational schema for the data which is interfaced with the expert modules. This architecture is seen to be beneficial from the standpoint of execution time.

To summarize, the experiences with DECA has led to the following conclusions:

1- For real time monitoring, diagnosis, and control of dynamic processes, qualitative reasoning will be of immense use.

2- Incorporation of qualitative reasoning as opposed to the pure optimization approaches will help in identifying the possible critical parameters along with their relative importance which will help reduce the processing time required.

3- It is necessary to handle data distinct from the knowledge base. It is apparent that a relational schema interface with the reasoning system will be of value due to the fact that the data is dynamic.

4- This experimentation has confirmed the fact that Milne's [Milne87] architecture integrated with the structuring of a dynamic database will be of importance in dynamical systems.

5- DECA demonstrates the point that all dynamical systems share the commonalty of prioritization and the generic scheme developed in this research is useful in almost every dynamical process control scenario.

Chapter 7

Future Research

Though the DECA system is a prototype, the initial results are encouraging and provide a strong argument toward further expansion of the system's capabilities.

7.1 Solution Generation

So far, the main efforts of DECA were to prove the concepts of qualitative information prioritization. Not much of the effort in this work was devoted to the development of customized solutions for every detail of a process's operation. Most efforts went into creating the ability to properly analyze the data and determine the critical areas of the process application in a real-time manner. Unless the system performs in real-time, there is little need for a comprehensive solution generation capability. Also, the solution generation is more specific to the application which DECA is being applied to, while the thrust of this thesis is to prove the viability of DECA to all dynamic processes.

Additional effort will be to devise a methodology in which the recommended solution for each scenario can be tailored according to the parameter rankings of the sensor data. Special considerations, similar to the ones for parameter expectancy data, would be needed to avoid a combinatorially explosive number of solutions (i.e. one for every single parameter combination).

7.2 Fuzzy Logic

Fuzzy mathematics [Zadeh65] have a great potential in the application of quantifying qualitative data. Applying fuzzy math to the evaluations of parameter and scenario ranks would help increase the resolution of the results. For example, at the present, DECA breaks up the likelihood of the scenarios into three categories (major, minor, and improbable). With fuzzy mathematics more subtle points can be brought into consideration increasing the thoroughness of DECA's evaluation.

7.3 Source Code Translation

DECA is presently written in Lisp. It is an ideal language for rapid prototyping of systems and handling abstract concepts, but it is also known for its computational overhead though Lisp Machines such as the Symbolics help reduce this overhead. Since a primary concern of DECA is for real-time processing, and the system will eventually be interfaced with physical controllers and sensors, another language has been targeted for the second stage of DECA's development. The language chosen is C. A couple of desired traits of C are its very fast and efficient execution, and there are many controllers set up to be easily interfaced with it.

The main problem to be addressed in this conversion is C does not have dynamic database capabilities like Lisp. If not done carefully, unwieldy data structures could add an unacceptable burden on the processor.

7.4 Integrating Analytical Modules

Though large analytical models may require too much cpu time, a simplified model or simulation of a subsystem can yield valuable insight into the current state of a process or subsystem. Also, a great deal of effort has been spent on the development of analytical methods. Future work on DECA can concentrate on implementing several analytical models which could run in parallel with DECA's qualitative model. The analytical models could be used as a verification to DECA's proposed conclusions.

Another aspect where the use of analytical models would be beneficial is to simulate the proposed solution before implementing it on the real system. This way DECA could avoid a potentially catastrophic mistake for it would be caught in the simulation.

Overall, the application of analytical elements in the DECA system would complement its qualitative abilities and increase the system's reliability.

7.5 Distributed Processing

DECA is being developed to monitor large scale systems with hundreds or thousands of parameters. In order for it to be successful in this area, the application process should be divided up into its subsystem with a separate DECA system monitoring each subsystem. To integrate all the distributed processors together into one working entity, there will be another DECA system overseeing all the DECA subsystems in a meta level fashion. It will monitor, evaluate, and rank all the conclusions of every subsystem. The meta level will work with the whole system and allow each subsystem to run independently, but have an override capability to resolve conflicts between subsystems.

The distributed processing scheme is a methodology to incorporate extra cpu power via multiple processors, yet still maintaining control over the entire dynamic process being monitored.

7.6 Operator Interface

DECA is acting as an advisor to the operators of the dynamic process being monitored. Thus it is very important to be sure that the transfer of information between DECA and the operators is being correctly interpreted. During future development phases, investigations will be made to see what is the best way to present the information, especially in large scale applications.

Another area to be developed is a user friendly interface to be used by the process experts, who may not be computer experts, to facilitate DECA's acquisition of knowledge from them. A properly developed interface will greatly enhance DECA's utility.

References

[Ander84] Anderson, Bruce M., et al., "Intelligent Automation of Emergency Procedures in Advanced Fighter Aircraft", The First Conference on Artificial Intelligence Applications, IEEE Computer Society, December 1984, pp 496-501.

[Bray85] Bray, M. A., Sebo, D. E., and Dixon, B. W., "Reactor Safety Assessment System - A Situation Assessment Aid for USNRC", Expert Systems in Government Symposium, IEEE Computer Society Press, 1985, pp 246-251.

[Bucha84] Buchanan, Bruce, Shortliffe, Edward, Rule-Based Expert Systems The MYCIN Experiments of the Stanford Heuristic Programming Project, Addison-Wesley, Reading, MA, 1984.

[Charn86] Charniak, E., McDermott, D., Introduction to Artificial Intelligence, Addison-Wesley, Reading, MA, 1986.

[DeKle83] De Kleer, J., Brown, J. S., "The Origin, Form and Logic of Qualitative Physical Laws", International Joint Conferences on Artificial Intelligence, Vol. 2, August 1983, pp 1158-1169.

[Dicke84] Dickey, F. J., Toussaint, A. L., "ECESIS: An Application of Expert Systems to Manned Space Stations", The First Conference on Artificial Intelligence Applications, IEEE Computer Society, December 1984, pp 483-489.

[Firsc86] Firschein, O., et al., Artificial Intelligence for Space Station Automation, Noyes Publications, Park Ridge, NJ, 1986.

[Frosc85] Froscher, Judith N., Jacob, Robert J. K., "Designing Expert Systems for Ease of Change", Expert Systems in Government Symposium, IEEE Computer Society Press, 1985, pp 246-251.

[Jow84] Jow, Hong-Nian J., Prioritization of Nuclear Power Plant Variables For Operator Assistance During Transients, Doctor of Philosophy Thesis - MIT, Cambridge, MA, May 1984.

[Kroen83] Kroenke, D., Database Processing, 2nd Ed., Science Research Associates Inc., Chicago, IL, 1983, pp 265-282.

[Kumar86] Kumara, S. R. T., Joshi, S., Kashyap, R. L., Moodie, C. L., and Chang, T. C., "Expert Systems in Industrial Engineering", International Journal of Production Research, Sep./Oct. 1986.

[Laffe87] Laffey, T. J., Schmidt, J. L., Read, J. Y., Kao, S. M., "A Multiprocessing Architecture for Real-Time Monitoring", Third Conference on Artificial Intelligence for Space Applications, NASA Conference Publication 2492, November 1987, pp 155-160.

[Leinw86] Leinweber, D., Gidwani, K., "Real-Time Expert System Development Techniques and Applications", Proceedings IEEE WESTEX-86, IEEE Computer Society, June 1986, pp 69-77.

[Leinw87] Leinweber, D., "Expert Systems in Space", IEEE Expert, Spring 1987, pp 26-36.

[McDer82] McDermott, J., "R1: A Rule-Based Configurer of Computer Systems", Artificial Intelligence, Vol. 19, 1982, pp 39-88

[Mille56] Miller, G. A., "The Magic Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information", Psychology Review, Vol. 63, No. 2, 1956, pp 81-97.

[Milne87] Milne, Robert, "Strategies for Diagnosis", IEEE Transactions on Systems, Man, and Cybernetics, Vol. smc-17, No. 3, May/June 1987, pp 333-339.

[Naray87] Narayanan, N. H., Viswanadham, N., "A Methodology for Knowledge Acquisition and Reasoning in Failure Analysis of Systems", IEEE Transactions on Systems, Man, and Cybernetics, Vol. smc-17, No. 2, March/April 1987, pp 274-288.

[Nilss80] Nilsson, N. J., Principles of Artificial Intelligence, Tioga Publishing Co., Palo Alto, CA, 1980.

[NSAC-1] Nuclear Safety Analysis Center (NSAC-1), Analysis of Three Mile Island Unit-2 Accident, Electric Power Research Institute, Palo Alto, CA, July 1979.

[NSAC-1S] NSAC-1 Supplement, Supplement to Analysis of Three Mile Island Unit-2 Accident, Electric Power Research Institute, Palo Alto, CA, Oct. 1979.

[Pisan84] Pisano, A. D., Jones, H. L., "An Expert Systems Approach to Adaptive Tactical Navigation", The First Conference on Artificial Intelligence Applications, IEEE Computer Society, December 1984, pp 460-464.

[Ray87] Ray, A., Joshi, S. M., Whitney, C. K., Jow, H. N., "Information Prioritization for Control and Automation of Space Operations", Proceedings of the 1987 American Control Conference, June 1987, pp 958-960.

[Shirl87] Shirley, R. S., "Some Lessons Using Expert Systems for Process Control", Proceedings of the 1987 American Control Conference, June 1987, pp 1342-1346.

[Steel84] Steele, Guy L. Jr., Common Lisp: The Language, Digital Press, Bedford, MA, 1984.

[Symbo86] Symbolics Computer Documentation Genera 7.1, Vols. 0-10, Symbolics Corporation, Cambridge, MA, June 1986.

[Tatar87] Tatar, Deborah G., A Programmer's Guide to Common Lisp, Digital Press, Bedford , MA, c 1987.

[Thomp88] Thompson, D. R., Ray, A., Kumara, S., "A Hierarchically Structured Knowledge-Based System for Welding Automation and Control", Journal of Engineering for Industry, Vol. 110, Feb. 1988, pp 71-76.

[Weinr80] Weinreb, D., Moon, D., "Flavors: Message Passing in the Lisp Machine", AI Memo No. 602, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1980.

[Whitn81] Whitney, C. K., "On the Philosophical Problem of Making a List", Physics as Natural Philosophy, MIT Press, Cambridge, MA, 1981.

[Zadeh65] Zadeh, L. A., "Fuzzy Sets", Information and Control, Vol. 8, 1965, pp 338-353.

[Zedeh86] Zadeh, L. A., "Outline of a Theory of Usuality Based on Fuzzy Logic", Fuzzy Sets Theory and Applications, A. Jones et al., D. Reidel Publishing, 1986, pp 79-97.

Appendix A

The System Query Language (SQL)

Database Manipulation Language

In recent years the emergence of powerful database management systems (DMS), enabling efficient manipulation of large databases with relative ease, has lent itself to widespread use as a primary method of information storage. One such database design is the relational database. The basic structure of the relational model consists of data tables. These data tables are called relations, and they represent the data itself and the output of processing of data too. In a loose analogy, it could be thought of being similar to Frames and Slots, a common knowledge representation in the Artificial Intelligence field. The only difference, and a rather large one at that, is the relational databases do not support message passing or property inheritance between relations, thus limiting its domain of application.

The relations are flat files; a two dimensional table containing several properties of the data. The basic structure which makes up the tables of a relational data base is the tuple. Each row of a relation is called a tuple. Each element of the tuple falls in to a different column. Each column of the table is an attribute of the system. This is somewhat similar to the top level structure of frames. In a relational model the term key refers to an attribute which can be used to uniquely identify a particular tuple. The use of keys to extract information out of the database makes it somewhat recursive in nature, which is generally an ideal way to approach knowledge retrieval in Expert Systems.

The retrieval of data is done through the use of a Data Manipulation Language (DML). There are four categories of these languages for relational data bases. They are relational algebra, relational calculus, transform-oriented languages, and graphic systems. System Query Language (SQL) is a common DML, uses a transform-oriented language. It provides a nonprocedural capability and uses relations to manipulate given the data into wanted results. The language has an English like syntax making it easy to use. SQL is also available on most large computers, thus making it a good candidate for having DECA interface to outside databases with it and tap the large amounts of data which are on these mainframes.

Some of the basic keywords of SQL are: SELECT, FROM, WHERE, IN, COUNT, SUM, AVG, MAX, MIN, GROUP BY, NOT, <, >, HAVING, EXISTS. SQL's syntax is straight forward lending to a powerful interface to retrieve data.

As an example, consider the setpoint database (SDB) in table A.1, along with some typical queries:

Table A.1 Setpoint Data Relation Table

Setpoint[VAR, LLL, LL, L, N, H, HH, HHH]

VARIABLE	LLL	LL	L	N	H	HH	HHH	UNITS
PZR-P	1200	1900	2055	2150	2250	2355	2400	psig
PZR-L	45	150	200	222	240	260	280	inches
HL1-T	300	400	500	606	610	619	630	deg F
SG1-P	800	850	900	940	1050	1070	1105	psig
SG1-L	10	30	45	160	170	180	190	inches
QNT-P	1	2	2.5	3	35	80	122	psig
SG2-P	800	850	900	940	1050	1070	1105	psig
SG2-L	10	30	45	160	170	180	190	inches
CL1-T	300	400	500	558	610	619	630	deg F

Query 1: Get the high (H) values desired for all relations from the database. The retrieval schema is as follows:

```
SELECT     H
FROM       SETPOINT
```

the results returned would be the data in column H.

```
2250
 240
 610
1050
 170
  35
1050
 170
 610
```

Query 2: Get the name of the parameter which has a high setpoint value that is greater than 2000 psig. The retrieval schema is as follows:

```
SELECT        VARIABLE
FROM          SETPOINT
   WHERE      UNITS='PSIG'
     AND      H > 2000
```

This would return the variable PZR-P.

One can see SQL has a very powerful interface, and is fairly straight forward to interface with any other program. Only the code would have to be written to enable DECA to send its own SQL commands to the database computer. This would help reduce the development time of DECA since the data management facility for various databases would not have to be written from scratch.

Appendix B

MYCIN's Control Mechanisms

MYCIN was one of the first successful Expert System implementations. Its purpose is to diagnose infectious bacterial diseases and to decide a treatment for the patient. It is an interactive consultant used by the physician as a diagnostic assistant. MYCIN relies on information from test results, patient consultation, and internal system inferences to arrive at its diagnosis. This appendix describes some of MYCIN's internals as presented in [Bucha84].

MYCIN's task involves a four-step decision problem:

1- Decide which organisms, if any, are causing significant disease.

2- Determine the likely identity of the significant organisms.

3- Decide which drugs are potentially useful.

4- Select the best drug or drugs.

MYCIN is a rule based Expert system. Typically the rules are of the form:

```
IF <antecedent is true>
THEN <take the designated action>
```

An example rule is as follows [Bucha84]:

```
RULE040
IF: 1) The site of the culture is blood, and
    2) The identity of the organism may be
       pseudomonas, and
    3) The patient has ecthyma gangrenosum skin
THEN:
    There is a strong suggestive evidence (.8)
    that the identity of the organism is
    pseudomonas.
```

RULE040 contains the Lookahead property in its structure. That is, before RULE040 can be executed as true, the system will have to verify whether or not premise 1, 2, and 3 are true by executing other rules in the system. This forward looking before making a decision is the Lookahead mechanism.

In MYCIN, there are some rules which have some of the same parameters in both the premise and consequent (action) statements. Such rules are known as self-referencing. When an Expert System contains rules which are self-referencing, there must also be a control structure to prevent the system from entering an infinite loop.

MYCIN uses a goal-oriented approach for executing rules. Two procedures, FINDOUT and MONITOR, are used to control the rule execution as well as prevent the system from entering an infinite loop. MONITOR analyzes the premise of a rule, one condition at a time, to see if it should execute the consequent. A block diagram of MONITOR is shown in figure B.2. In FINDOUT (fig B.1), its purpose is to obtain the missing information for MONITOR via other rules or asking the user for data input.

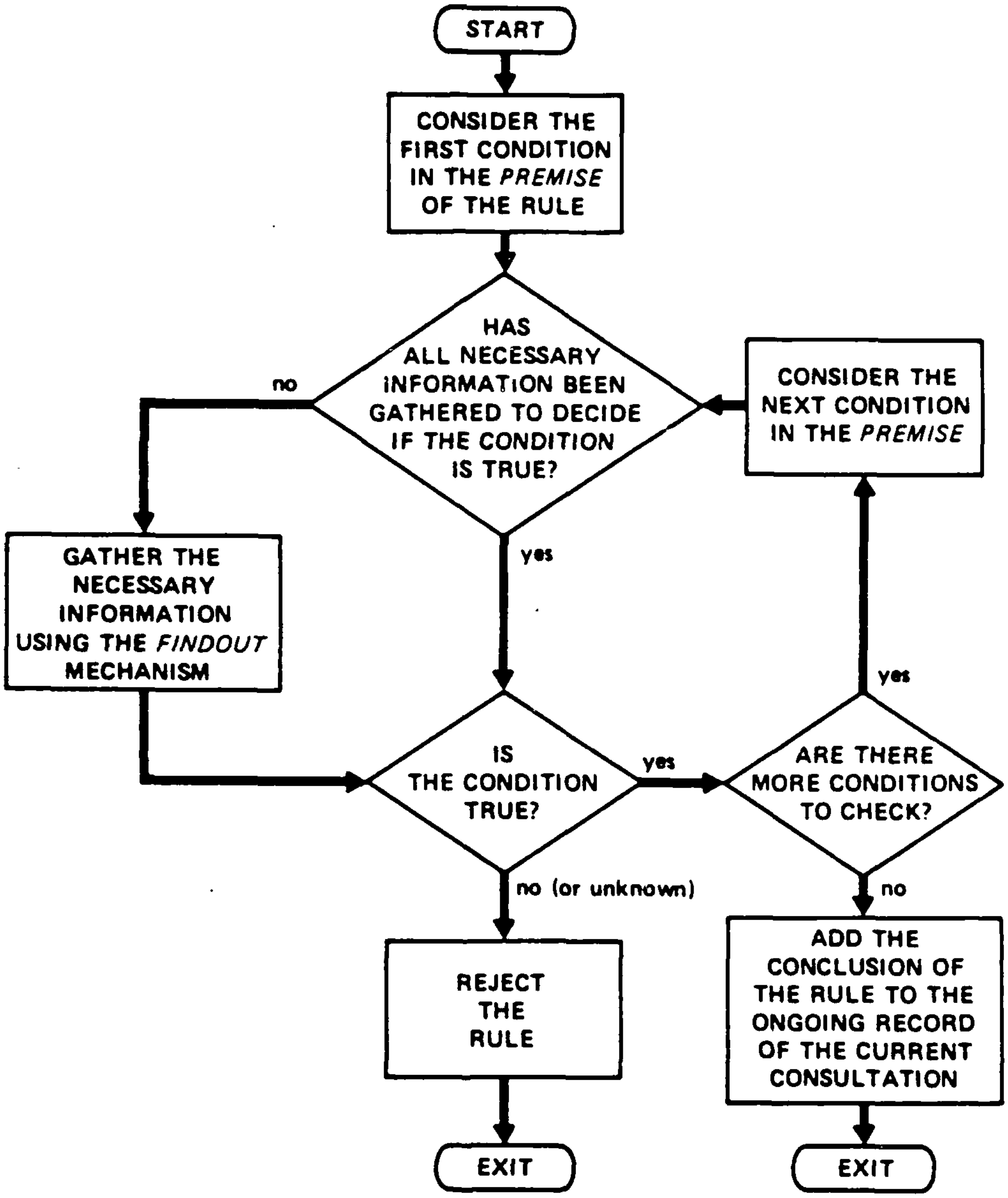

START
CONSIDER THE FIRST CONDITION IN THE *PREMISE* OF THE RULE
HAS ALL NECESSARY INFORMATION BEEN GATHERED TO DECIDE IF THE CONDITION IS TRUE?
no
yes
GATHER THE NECESSARY INFORMATION USING THE *FINDOUT* MECHANISM
CONSIDER THE NEXT CONDITION IN THE *PREMISE*
IS THE CONDITION TRUE?
yes
ARE THERE MORE CONDITIONS TO CHECK?
yes
no (or unknown)
no
REJECT THE RULE
ADD THE CONCLUSION OF THE RULE TO THE ONGOING RECORD OF THE CURRENT CONSULTATION
EXIT
EXIT

Figure B.1 MONITOR Mechanism [Bucha84, p. 106]

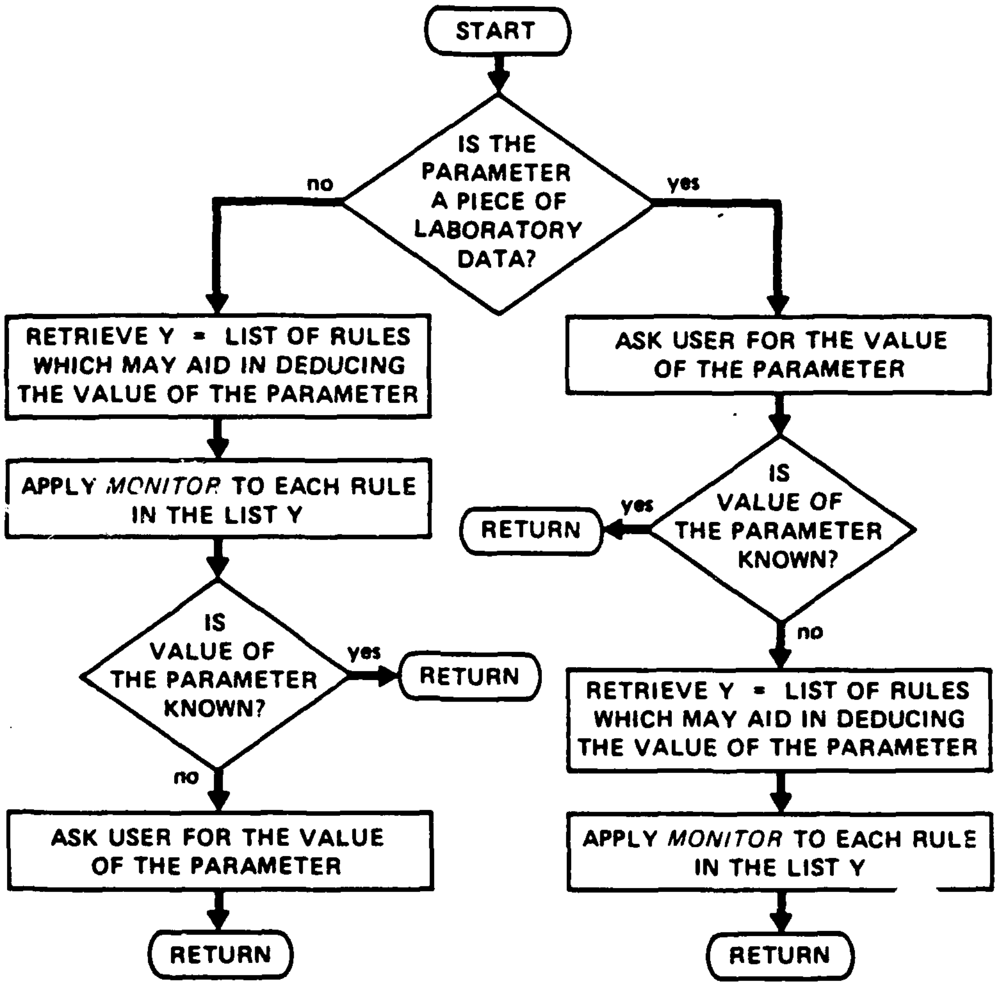

START
IS THE PARAMETER A PIECE OF LABORATORY DATA?
no
yes
RETRIEVE Y = LIST OF RULES WHICH MAY AID IN DEDUCING THE VALUE OF THE PARAMETER
APPLY MONITOR TO EACH RULE IN THE LIST Y
IS VALUE OF THE PARAMETER KNOWN?
yes
RETURN
no
ASK USER FOR THE VALUE OF THE PARAMETER
RETURN
ASK USER FOR THE VALUE OF THE PARAMETER
IS VALUE OF THE PARAMETER KNOWN?
yes
RETURN
no
RETRIEVE Y = LIST OF RULES WHICH MAY AID IN DEDUCING THE VALUE OF THE PARAMETER
APPLY MONITOR TO EACH RULE IN THE LIST Y
RETURN

Figure B.2 FINDOUT Mechanism [Bucha84, p. 107]

Note that FINDOUT is accessed from MONITOR, and MONITOR may be accessed from FINDOUT. This recursive feature enables the generation of a reasoning network which is best suited for each patient, and it also will cause MYCIN to select the necessary questions and rules to use.

Another important control structure is that FINDOUT doesn't check whether the premise of the rule is true. It only exhaustively traces a parameter and returns its value to MONITOR. Then in MONITOR, the condition may, with its new information, be evaluated. With this control structure FINDOUT is called only once for each parameter while MONITOR may be called multiple times. Also, when MONITOR reaches the question [Bucha84, p. 106]; "HAS ALL THE NECESSARY INFORMATION BEEN GATHERED TO DECIDE IF THE CONDITION IS TRUE?" (see figure B.2), the parameter is then passed to FINDOUT unless it is marked as already being traced. These two features are what prevents MYCIN from going into an infinite loop.

This concludes the explanation of MYCIN control structure. One can see that the architecture enables MYCIN to be flexible with the generation of its inquiry, as well as adaptive, in that it asks only pertinent questions.

DECA also has a Lookahead Mechanism. In it, DECA tries to determine whether other parameters for a given scenario are out of bounds before determining if some of conditions for the scenario are present. Basically it looks ahead to see that all preconditions are satisfied. As the rules in

DECA are nonself-referencing, its control cycle architecture is not as complex as that of MYCIN.

In summary, the setup of the Lookahead mechanism adds the capability for both DECA and MYCIN to customize their "thought process" for more efficient operation. This is important for DECA since it is operating in a real time environment.

Appendix C

The CKW Local Optimization Algorithm

This appendix deals with Jow's work in the development of the CKW algorithm. Most parts of this appendix are extracted from Jow's thesis [Jow84].

The motivating factor behind the CKW algorithm comes from the ability to operate in real-time on a typical minicomputer found in a nuclear power plant (e.g. VAX 11/780) and advise any maladies in the system.

In order to enable the system to operate in real time, the CKW algorithm could not yield a globally optimal solution (i.e., an optimal solution over the entire problem space), due to computational limitations. Thus the developers pursued a sub-optimal solution employing Local Optimization techniques.

Jow has pointed out a few characteristics of Local Optimization [Jow84, p. 3-2].

1- It is not necessarily the globally optimal solution, though it can be.

2- It provides for an algorithm which has a polynomial rate of increase of computational complexity with rate of growth in the number of parameters.

The CKW algorithm was first proposed by Dr. Cynthia K. Whitney for a scheduling problem with the High Energy Laser

Weapon [Whitn81]. The purpose is to schedule the weapon to irradiate N number of different threats in a limited time period. The globally optimal solution has an exponential increase in computational complexity with a linear increase in the number of threats. Thus with a potentially vast number of threats, a methodology to make the increase of computational complexity of a polynomial order to a linear increase in the number of threats was needed. This lead to the development of a local optimization.

The basic features of the CKW algorithm are [Whitn81, Jow84]:

1- It decides what member (parameter) should be chosen at each stage of the decision immediately after looking at one member beyond the stage under consideration.
2- At any stage of the decision (search), it looks at the performance measure of each competing member using the satisfactory outcome of the remaining or pending members which have not been chosen so far.
3- At any stage of the decision (search), it uses a "lumped urgency" (a combinatorial argument based on the number of members that remain pending and available opportunities for them) to assist in the selection of a member.

The system will try to find the best solution at each stage via feature one, with feature two and three acting as moderators to the decision process of the CKW algorithm. This moderating effect prevents the CKW algorithm from making a too premature decision.

The CKW algorithm sets up an algorithmic procedure to select the order of importance of the parameters of a system in a time constrained environment. The concepts presented in Jow's work to develop the CKW algorithm into a decision support system is the basic motivation factor for the development of the DECA system. The author sees a great potential in harnessing Artificial Intelligence and Expert System techniques as a second method of problem solving for decision support systems. Eventually both methods could be used in one system which could then take advantage of both the analytical strengths of the CKW and the qualitative reasoning ideas of DECA. The general architecture of the DECA system is designed to readily facilitate the integration of analytical submodules into the system.

Appendix D

Runtime Output Data

This appendix contains the output for DECA's test run and for the TMI-2 accident run.

D.1 Test Run

The DECA system was verified to be working after successfully completing the test run. It consisted of the following three sets of sensor data which DECA evaluated. The sensor data is shown below:

```
((05 (2150 222 606 940 160 3 940 160 558))
 (10 (2260 270 606 870 42 1.9 910 42 635))
 (15 (2380 282 610 870 29 0.9 860 42 635)))
```

The output from DECA's run showing the values of the intermediate variables and DECA's conclusions is shown below:

```
Intermediate parameters for system time: 5

Sensor-record  (2150 222 606 940 160 3 940 160 558)
Oob-parameters NIL
Oob-parameters-values NIL
Oob-severity NIL

Lookahead-scenarios  NIL

Scenario-data-match-list ((1 NIL) (2 NIL) (3 NIL) (4 NIL) (5 NIL) (6 NIL) (7 NIL) (8 NIL) (9 NIL) (10 NIL) (11 N
 (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 0 IMPROBABLE) (2 0 IMPROBABLE) (3 0 IMPROBABLE) (4 0 IMPROBABLE) (5 0 IMPROBABLE)
 (6 0 IMPROBABLE) (7 0 IMPROBABLE) (8 0 IMPROBABLE) (9 0 IMPROBABLE) (10 0 IMPROBABLE)
 (11 0 IMPROBABLE) (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor NIL
Scenario-improbable ((12 0) (11 0) (10 0) (9 0) (8 0) (7 0) (6 0) (5 0) (4 0) (3 0) (2 0) (1 0))

Parameter-ratio-match ((PZR-P NIL) (PZR-L NIL) (HL1-T NIL) (SG1-P NIL) (SG1-L NIL) (QNT-P NIL) (SG2-P NIL)
 (SG2-L NIL) (CL1-T NIL))
Parameter-rank-list NIL

Parameter-rank-list (NIL)

Parameter-rank-list NIL

Possible-scenarios-for-situation NIL


DECA's conclusions for system time: 5

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:


 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description
```

```
End of data evaluation for system time: 5

Intermediate parameters for system time: 10

Sensor-record  (2260 270 606 870 42 1.9 910 42 635)
Oob-parameters (PZR-P PZR-L SG1-P SG1-L QNT-P SG2-L CL1-T)
Oob-parameters-values (2260 270 870 42 1.9 42 635)
Oob-severity (H HH L L LL L HHH)

Lookahead-scenarios  (1 2 3 4 5 6 7 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L SG1-P)) (2 (SG2-L SG1-L SG1-P)) (3 (QNT-P)) (4 (PZR-L))
 (5 (SG2-L SG1-L SG1-P)) (6 (SG2-L SG1-L)) (7 (CL1-T)) (8 (CL1-T SG2-L SG1-L SG1-P))
 (9 (CL1-T)) (10 (CL1-T SG2-L SG1-L)) (11 (CL1-T SG2-L SG1-L)) (12 (CL1-T SG1-P)))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 1/2 MINOR) (2 3/7 MINOR) (3 1/3 MINOR) (4 1/3 MINOR) (5 3/5 MINOR) (6 2/5 MINOR)
 (7 1/4 IMPROBABLE) (8 2/3 MINOR) (9 1/5 IMPROBABLE) (10 3/5 MINOR) (11 3/5 MINOR)
 (12 2/5 MINOR))

Scenario-major NIL
Scenario-minor ((8 2/3) (11 3/5) (10 3/5) (5 3/5) (1 1/2) (2 3/7) (12 2/5) (6 2/5) (4 1/3) (3 1/3))
Scenario-improbable ((7 1/4) (9 1/5))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL)
 (SG1-P ((10) NIL (12 11 10 8 6 5 2 1) NIL)) (SG1-L ((9) NIL (12 11 10 8 6 5 2 1) NIL))
 (QNT-P ((3) NIL (4 3 2) NIL)) (SG2-P NIL) (SG2-L ((9) NIL (12 11 10 8 6 5 2 1) NIL))
 (CL1-T ((7) NIL (12 11 10 8 5) NIL)))
Parameter-rank-list ((QNT-P 10) (PZR-L 10) (PZR-P 10) (SG2-L 9.3) (SG1-L 9.3) (CL1-T 8.6) (SG1-P 8.5))

Parameter-rank-list ((PZR-P PZR-L QNT-P 10) (SG1-L SG2-L 9.3) (CL1-T 8.6) (SG1-P 8.5))

Parameter-rank-list ((PZR-L 10) (QNT-P 10) (PZR-P 10) (SG1-L 9.3) (SG2-L 9.3) (CL1-T 8.6) (SG1-P 8.5))

Possible-scenarios-for-situation ((8 2/3) (11 3/5) (10 3/5) (5 3/5) (1 1/2) (2 3/7) (12 2/5) (6 2/5) (4 1/3) (3

DECA's conclusions for system time: 10

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-L     10
   QNT-P     10
```

```
  PZR-P    10
  SG1-L    9.3
  SG2-L    9.3
  CL1-T    8.6
  SG1-P    8.5


 Scenarios that were considered as possible choices but not
 selected are:

 Scenario  Ratio   Description

   8    2/3     Steam Generator - Primary Coolant System
   11    3/5     Feedwater Pump - Secondary Coolant System
   10    3/5     Pipe Rupture   - Secondary Coolant System
   5    3/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
   1    1/2     Pressurizer Leak
   2    3/7     Block Valve Leak
   12    2/5     Turbine Trip   - Secondary Coolant System
   6    2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
   4    1/3     Drain Tank
   3    1/3     Pipe Rupture    - (drain tank)


End of data evaluation for system time: 10

Intermediate parameters for system time: 15

Sensor-record  (2380 282 610 870 29 0.9 860 42 635)
Oob-parameters (PZR-P PZR-L SG1-P SG1-L QNT-P SG2-P SG2-L CL1-T)
Oob-parameters-values (2380 282 870 29 0.9 860 42 635)
Oob-severity (HH HHH L LL LLL L L HHH)

Lookahead-scenarios  (1 2 3 4 5 6 7 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG2-P SG1-L SG1-P)) (2 (SG2-L SG2-P SG1-L SG1-P)) (3 (QNT-P)) (4 (PZR-L))
 (5 (SG2-L SG2-P SG1-L SG1-P)) (6 (SG2-L SG1-L)) (7 (CL1-T))
 (8 (CL1-T SG2-L SG2-P SG1-L SG1-P)) (9 (CL1-T)) (10 (CL1-T SG2-L SG1-L))
 (11 (CL1-T SG2-L SG1-L)) (12 (CL1-T SG2-P SG1-P)))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 1/3 MINOR) (4 1/3 MINOR) (5 4/5 MAJOR) (6 2/5 MINOR)
 (7 1/4 IMPROBABLE) (8 5/6 MAJOR) (9 1/5 IMPROBABLE) (10 3/5 MINOR) (11 3/5 MINOR)
 (12 3/5 MINOR))

Scenario-major ((8 5/6) (5 4/5))
```

```
Scenario-minor ((1 2/3) (12 3/5) (11 3/5) (10 3/5) (2 4/7) (6 2/5) (4 1/3) (3 1/3))
Scenario-improbable ((7 1/4) (9 1/5))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL)
 (SG1-P ((10) (8 5) (12 11 10 6 2 1) NIL)) (SG1-L ((9) (8 5) (12 11 10 6 2 1) NIL))
 (QNT-P ((3) NIL (4 3 2) NIL)) (SG2-P ((10) (8 5) (12 11 10 6 2 1) NIL))
 (SG2-L ((9) (8 5) (12 11 10 6 2 1) NIL)) (CL1-T ((7) (8 5) (12 11 10) NIL)))
Parameter-rank-list ((QNT-P 10) (PZR-L 10) (PZR-P 10) (SG2-L 9.3) (SG1-L 9.3) (CL1-T 8.6) (SG2-P 8.5) (SG1-P 8.5

Parameter-rank-list ((PZR-P PZR-L QNT-P 10) (SG1-L SG2-L 9.3) (CL1-T 8.6) (SG1-P SG2-P 8.5))

Parameter-rank-list ((PZR-L 10) (QNT-P 10) (PZR-P 10) (SG1-L 9.3) (SG2-L 9.3) (CL1-T 8.6) (SG1-P 8.5) (SG2-P 8.5

Possible-scenarios-for-situation ((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


DECA's conclusions for system time: 15

 Scenario selected is;
 Scenario Number 8
 Scenario Description (Steam Generator - Primary Coolant System )
 Confidence  5/6


 The parameter and priorities are as follows:

   PZR-L    10
   QNT-P    10
   PZR-P    10
   SG1-L    9.3
   SG2-L    9.3
   CL1-T    8.6
   SG1-P    8.5
   SG2-P    8.5


 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description

    8    5/6     Steam Generator - Primary Coolant System
    5    4/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
    1    2/3     Pressurizer Leak
    12    3/5     Turbine Trip   - Secondary Coolant System
    11    3/5     Feedwater Pump - Secondary Coolant System
    10    3/5     Pipe Rupture   - Secondary Coolant System
    2    4/7     Block Valve Leak
```

```
     6    2/5    Pipe Rupture    - Cold Leg, Primary Coolant System
     4    1/3    Drain Tank
     3    1/3    Pipe Rupture    - (drain tank)

End of data evaluation for system time: 15
```

D.2 TMI-2 Run

The concepts of the DECA system were verified using actual data from the TMI-2 accident. It consisted of sensor data from nine different times during the accident. The sensor data shown below was extracted from [Jow84, pp. 5-7 to 5-11].

```
((0   (2145 218 607  944 123  3    930 116 559))
 (15  (2260 253 611 1022  79  6.3 1012  80 571))
 (30  (1905 182 587  998  26  7.8  987  30 577))
 (45  (1855 160 579 1000  17  9.3  993  20 576))
 (60  (1790 158 578  990  14 12    969  18 576))
 (75  (1760 162 577 1011  10 14.3  997  16 576))
 (90  (1725 175 578 1023  11 17.5 1005  16 577))
 (105 (1685 187 579 1021  11 19.6 1005  16 577))
 (120 (1650 200 579 1011  11 22.2 1000  16 579)))
```

```
Intermediate parameters for system time: 0

Sensor-record  (2145 218 607 944 123 3 930 116 559)
Oob-parameters NIL
Oob-parameters-values NIL
Oob-severity NIL

Lookahead-scenarios  NIL

Scenario-data-match-list ((1 NIL) (2 NIL) (3 NIL) (4 NIL) (5 NIL) (6 NIL) (7 NIL) (8 NIL) (9 NIL) (10 NIL) (11 N
 (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 0 IMPROBABLE) (2 0 IMPROBABLE) (3 0 IMPROBABLE) (4 0 IMPROBABLE) (5 0 IMPROBABLE)
 (6 0 IMPROBABLE) (7 0 IMPROBABLE) (8 0 IMPROBABLE) (9 0 IMPROBABLE) (10 0 IMPROBABLE)
 (11 0 IMPROBABLE) (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor NIL
Scenario-improbable ((12 0) (11 0) (10 0) (9 0) (8 0) (7 0) (6 0) (5 0) (4 0) (3 0) (2 0) (1 0))

Parameter-ratio-match ((PZR-P NIL) (PZR-L NIL) (HL1-T NIL) (SG1-P NIL) (SG1-L NIL) (QNT-P NIL) (SG2-P NIL)
 (SG2-L NIL) (CL1-T NIL))
Parameter-rank-list NIL

Parameter-rank-list (NIL)

Parameter-rank-list NIL

Possible-scenarios-for-situation NIL


DECA's conclusions for system time: 0

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:



 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description
```

```
End of data evaluation for system time: 0

Intermediate parameters for system time: 15

Sensor-record  (2260 253 611 1022 79 6.3 1012 80 571)
Oob-parameters (PZR-P PZR-L HL1-T)
Oob-parameters-values (2260 253 611)
Oob-severity (H H H)

Lookahead-scenarios  (1 2 3 4 6 7 8)

Scenario-data-match-list ((1 NIL) (2 NIL) (3 NIL) (4 (PZR-L)) (5 NIL) ( [illegible]HL1-T)) (7 (HL1-T)) (8 (HL1-T)) (9 NIL
 (10 NIL) (11 NIL) (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 0 IMPROBABLE) (2 0 IMPROBABLE) (3 0 IMPROBABLE) (4 1/3 MINOR) (5 0 IMPROBABLE)
 (6 1/5 IMPROBABLE) (7 1/4 IMPROBABLE) (8 1/6 IMPROBABLE) (9 0 IMPROBABLE) (10 0 IMPROBABLE)
 (11 0 IMPROBABLE) (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor ((4 1/3))
Scenario-improbable ((7 1/4) (6 1/5) (8 1/6) (12 0) (11 0) (10 0) (9 0) (5 0) (3 0) (2 0) (1 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4) NIL)) (PZR-L ((4) NIL (4) NIL)) (HL1-T ((3) NIL NIL NIL)) (SG1-P NIL)
 (SG1-L NIL) (QNT-P NIL) (SG2-P NIL) (SG2-L NIL) (CL1-T NIL))
Parameter-rank-list ((PZR-L 2.2) (PZR-P 2.2) (HL1-T 1))

Parameter-rank-list ((PZR-P PZR-L 2.2) (HL1-T 1))

Parameter-rank-list ((PZR-P 2.2) (PZR-L 2.2) (HL1-T 1))

Possible-scenarios-for-situation ((4 1/3))


DECA's conclusions for system time: 15

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    2.2
   PZR-L    2.2
   HL1-T    1
 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description

    4        1/3      Drain Tank


End of data evaluation for system time: 15

Intermediate parameters for system time: 30

Sensor-record  (1905 182 587 998 26 7.8 987 30 577)
Oob-parameters (PZR-P PZR-L SG1-L SG2-L)
Oob-parameters-values (1905 182 26 30)
Oob-severity (L L LL LL)

Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L PZR-L PZR-P)) (2 (SG2-L SG1-L PZR-L PZR-P)) (3 (PZR-L PZR-P)) (4 (PZR-
 (5 (SG2-L SG1-L)) (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L))
 (11 (SG2-L SG1-L)) (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 2/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
 (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
 (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))
Scenario-improbable ((12 0) (9 0) (7 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL) (SG1-P NIL)
 (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (QNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-L 10) (PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P PZR-L 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (PZR-L 10) (SG1-L 8.5) (SG2-L 8.5))

Possible-scenarios-for-situation ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))


DECA's conclusions for system time: 30
```

```
No scenario selected, not confident enough.

The parameter and priorities are as follows:

  PZR-P    10
  PZR-L    10
  SG1-L    8.5
  SG2-L    8.5

Scenarios that were considered as possible choices but not
selected are:

Scenario   Ratio   Description

    3     2/3     Pipe Rupture    - (drain tank)
    1     2/3     Pressurizer Leak
    2     4/7     Block Valve Leak
    11     2/5     Feedwater Pump - Secondary Coolant System
    10     2/5     Pipe Rupture   - Secondary Coolant System
    6     2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
    5     2/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
    8     1/3     Steam Generator - Primary Coolant System
    4     1/3     Drain Tank


End of data evaluation for system time: 30

Intermediate parameters for system time: 45

Sensor-record  (1855 160 579 1000 17 9.3 993 20 576)
Oob-parameters (PZR-P PZR-L SG1-L SG2-L)
Oob-parameters-values (1855 160 17 20)
Oob-severity (LL L LL LL)

Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L PZR-L PZR-P)) (2 (SG2-L SG1-L PZR-L PZR-P)) (3 (PZR-L PZR-P)) (4 (PZR-
 (5 (SG2-L SG1-L)) (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L))
 (11 (SG2-L SG1-L)) (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 2/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
 (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
 (12 0 IMPROBABLE))
```

```
Scenario-major NIL
Scenario-minor ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))
Scenario-improbable ((12 0) (9 0) (7 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL) (SG1-P NIL)
 (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (QNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-L 10) (PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P PZR-L 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (PZR-L 10) (SG1-L 8.5) (SG2-L 8.5))

Possible-scenarios-for-situation ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))

DECA's conclusions for system time: 45

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    10
   PZR-L    10
   SG1-L    8.5
   SG2-L    8.5

 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description

    3     2/3     Pipe Rupture    - (drain tank)
    1     2/3     Pressurizer Leak
    2     4/7     Block Valve Leak
    11     2/5     Feedwater Pump - Secondary Coolant System
    10     2/5     Pipe Rupture   - Secondary Coolant System
    6     2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
    5     2/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
    8     1/3     Steam Generator - Primary Coolant System
    4     1/3     Drain Tank


End of data evaluation for system time: 45

Intermediate parameters for system time: 60
```

```
Sensor-record  (1790 158 578 990 14 12 969 18 576)
Oob-parameters (PZR-P PZR-L SG1-L SG2-L)
Oob-parameters-values (1790 158 14 18)
Oob-severity (LL L LL LL)

Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L PZR-L PZR-P)) (2 (SG2-L SG1-L PZR-L PZR-P)) (3 (PZR-L PZR-P)) (4 (PZR-
 (5 (SG2-L SG1-L)) (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L))
 (11 (SG2-L SG1-L)) (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 2/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
 (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
 (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))
Scenario-improbable ((12 0) (9 0) (7 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL) (SG1-P NIL)
 (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (QNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-L 10) (PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P PZR-L 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (PZR-L 10) (SG1-L 8.5) (SG2-L 8.5))

Possible-scenarios-for-situation ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))


DECA's conclusions for system time: 60

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    10
   PZR-L    10
   SG1-L    8.5
   SG2-L    8.5


 Scenarios that were considered as possible choices but not
 selected are:
```

```
Scenario  Ratio   Description

   3     2/3     Pipe Rupture    - (drain tank)
   1     2/3     Pressurizer Leak
   2     4/7     Block Valve Leak
   11     2/5     Feedwater Pump - Secondary Coolant System
   10     2/5     Pipe Rupture   - Secondary Coolant System
   6     2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
   5     2/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
   8     1/3     Steam Generator - Primary Coolant System
   4     1/3     Drain Tank


End of data evaluation for system time: 60

Intermediate parameters for system time: 75

Sensor-record  (1760 162 577 1011 10 14.3 997 16 576)
Oob-parameters (PZR-P PZR-L SG1-L SG2-L)
Oob-parameters-values (1760 162 10 16)
Oob-severity (LL L LLL LL)

Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L PZR-L PZR-P)) (2 (SG2-L SG1-L PZR-L PZR-P)) (3 (PZR-L PZR-P)) (4 (PZR-
 (5 (SG2-L SG1-L)) (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L))
 (11 (SG2-L SG1-L)) (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 2/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
 (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
 (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))
Scenario-improbable ((12 0) (9 0) (7 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL) (SG1-P NIL)
 (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (QNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-L 10) (PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P PZR-L 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (PZR-L 10) (SG1-L 8.5) (SG2-L 8.5))
```

```
Possible-scenarios-for-situation ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))

DECA's conclusions for system time: 75

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    10
   PZR-L    10
   SG1-L    8.5
   SG2-L    8.5

 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description

    3    2/3     Pipe Rupture    - (drain tank)
    1    2/3     Pressurizer Leak
    2    4/7     Block Valve Leak
    11    2/5     Feedwater Pump - Secondary Coolant System
    10    2/5     Pipe Rupture   - Secondary Coolant System
    6    2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
    5    2/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
    8    1/3     Steam Generator - Primary Coolant System
    4    1/3     Drain Tank


End of data evaluation for system time: 75

Intermediate parameters for system time: 90

Sensor-record  (1725 175 578 1023 11 17.5 1005 16 577)
Oob-parameters (PZR-P PZR-L SG1-L SG2-L)
Oob-parameters-values (1725 175 11 16)
Oob-severity (LL L LL LL)

Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L PZR-L PZR-P)) (2 (SG2-L SG1-L PZR-L PZR-P)) (3 (PZR-L PZR-P)) (4 (PZR-
 (5 (SG2-L SG1-L)) (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L))
 (11 (SG2-L SG1-L)) (12 NIL))
```

```
Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 2/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
 (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
 (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))
Scenario-improbable ((12 0) (9 0) (7 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL) (SG1-P NIL)
 (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (QNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-L 10) (PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P PZR-L 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (PZR-L 10) (SG1-L 8.5) (SG2-L 8.5))

Possible-scenarios-for-situation ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))


DECA's conclusions for system time: 90

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    10
   PZR-L    10
   SG1-L    8.5
   SG2-L    8.5


 Scenarios that were considered as possible choices but not
 selected are:

 Scenario  Ratio   Description

    3     2/3     Pipe Rupture    - (drain tank)
    1     2/3     Pressurizer Leak
    2     4/7     Block Valve Leak
    11     2/5     Feedwater Pump - Secondary Coolant System
    10     2/5     Pipe Rupture   - Secondary Coolant System
    6     2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
    5     2/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
    8     1/3     Steam Generator - Primary Coolant System
    4     1/3     Drain Tank
```

```
End of data evaluation for system time: 90

Intermediate parameters for system time: 105

Sensor-record  (1685 187 579 1021 11 19.6 1005 16 577)
Oob-parameters (PZR-P PZR-L SG1-L SG2-L)
Oob-parameters-values (1685 187 11 16)
Oob-severity (LL L LL LL)

Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

Scenario-data-match-list ((1 (SG2-L SG1-L PZR-L PZR-P)) (2 (SG2-L SG1-L PZR-L PZR-P)) (3 (PZR-L PZR-P)) (4 (PZR-
 (5 (SG2-L SG1-L)) (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L))
 (11 (SG2-L SG1-L)) (12 NIL))

Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

Scenario-ratio-match-list ((1 2/3 MINOR) (2 4/7 MINOR) (3 2/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
 (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
 (12 0 IMPROBABLE))

Scenario-major NIL
Scenario-minor ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))
Scenario-improbable ((12 0) (9 0) (7 0))

Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L ((4) NIL (4 3 2 1) NIL)) (HL1-T NIL) (SG1-P NIL)
 (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (QNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-L 10) (PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P PZR-L 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (PZR-L 10) (SG1-L 8.5) (SG2-L 8.5))

Possible-scenarios-for-situation ((3 2/3) (1 2/3) (2 4/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3))


DECA's conclusions for system time: 105

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    10
   PZR-L    10
   SG1-L    8.5
   SG2-L    8.5

  Scenarios that were considered as possible choices but not
  selected are:

  Scenario   Ratio    Description

     3     2/3     Pipe Rupture     - (drain tank)
     1     2/3     Pressurizer Leak
     2     4/7     Block Valve Leak
     11     2/5     Feedwater Pump - Secondary Coolant System
     10     2/5     Pipe Rupture   - Secondary Coolant System
     6     2/5     Pipe Rupture     - Cold Leg, Primary Coolant System
     5     2/5     Pipe Rupture     - Hot Leg,  Primary Coolant System
     8     1/3     Steam Generator - Primary Coolant System
     4     1/3     Drain Tank


 End of data evaluation for system time: 105

 Intermediate parameters for system time: 120

 Sensor-record  (1650 200 579 1011 11 22.2 1000 16 579)
 Oob-parameters (PZR-P SG1-L SG2-L)
 Oob-parameters-values (1650 11 16)
 Oob-severity (LL LL LL)

 Lookahead-scenarios  (1 2 3 4 5 6 8 9 10 11 12)

 Scenario-data-match-list ((1 (SG2-L SG1-L PZR-P)) (2 (SG2-L SG1-L PZR-P)) (3 (PZR-P)) (4 (PZR-P)) (5 (SG2-L SG1-
  (6 (SG2-L SG1-L)) (7 NIL) (8 (SG2-L SG1-L)) (9 NIL) (10 (SG2-L SG1-L)) (11 (SG2-L SG1-L))
  (12 NIL))

 Parameters-per-scenario-expect ((1 6) (2 7) (3 3) (4 3) (5 5) (6 5) (7 4) (8 6) (9 5) (10 5) (11 5) (12 5))

 Scenario-ratio-match-list ((1 1/2 MINOR) (2 3/7 MINOR) (3 1/3 MINOR) (4 1/3 MINOR) (5 2/5 MINOR) (6 2/5 MINOR)
  (7 0 IMPROBABLE) (8 1/3 MINOR) (9 0 IMPROBABLE) (10 2/5 MINOR) (11 2/5 MINOR)
  (12 0 IMPROBABLE))

 Scenario-major NIL
 Scenario-minor ((1 1/2) (2 3/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3) (3 1/3))
 Scenario-improbable ((12 0) (9 0) (7 0))

 Parameter-ratio-match ((PZR-P ((4) NIL (4 3 2 1) NIL)) (PZR-L NIL) (HL1-T NIL) (SG1-P NIL)
  (SG1-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CNT-P NIL) (SG2-P NIL)
 (SG2-L ((9) NIL (11 10 8 6 5 2 1) NIL)) (CL1-T NIL))
Parameter-rank-list ((PZR-P 10) (SG2-L 8.5) (SG1-L 8.5))

Parameter-rank-list ((PZR-P 10) (SG1-L SG2-L 8.5))

Parameter-rank-list ((PZR-P 10) (SG1-L 8.5) (SG2-L 8.5))

Possible-scenarios-for-situation ((1 1/2) (2 3/7) (11 2/5) (10 2/5) (6 2/5) (5 2/5) (8 1/3) (4 1/3) (3 1/3))

DECA's conclusions for system time: 120

 No scenario selected, not confident enough.

 The parameter and priorities are as follows:

   PZR-P    10
   SG1-L    8.5
   SG2-L    8.5

 Scenarios that were considered as possible choices but not
 selected are:

 Scenario   Ratio    Description

    1     1/2     Pressurizer Leak
    2     3/7     Block Valve Leak
    11     2/5     Feedwater Pump - Secondary Coolant System
    10     2/5     Pipe Rupture   - Secondary Coolant System
    6     2/5     Pipe Rupture    - Cold Leg, Primary Coolant System
    5     2/5     Pipe Rupture    - Hot Leg,  Primary Coolant System
    8     1/3     Steam Generator - Primary Coolant System
    4     1/3     Drain Tank
    3     1/3     Pipe Rupture    - (drain tank)


End of data evaluation for system time: 120
```

APPENDIX E

DECA Knowledge Base for TMI-2 Accident

This appendix contains both explanations and a listing of the various data files which were employed by DECA to give it the knowledge about the nuclear reactor for the Three Mile Island Unit 2 nuclear accident.

DECA is designed to be a generic system, thus its inference engine and knowledge base must be separate. In order to use DECA on a different process, one will just have to load up the new data and not have to make any changes to the computer code. Appendix E contains the knowledge base files listings and Appendix F contains the inference engine listing.

A schematic diagram showing the components of the TMI-2 reactor is shown in figure E.1.

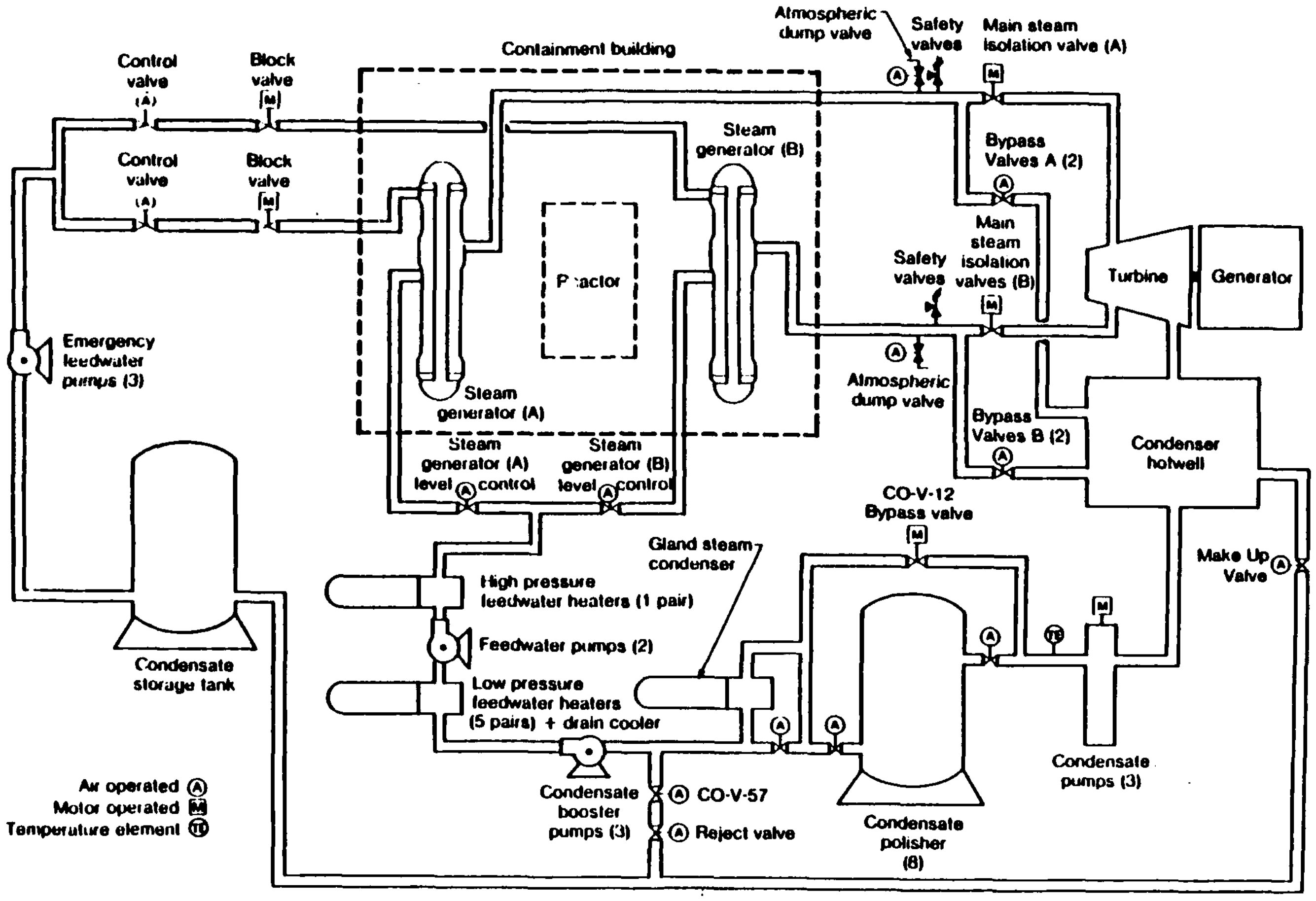


Figure E.1 TMI-2 Schematic [NSAC-1, appendix C/FDW]

E.1 Scenario-description.data File

Listed below is the data file which gives the scenario numbers and a description of the scenario for the TMI-2 runs. For example, scenario 3 is the scenario where there is a pipe rupture in the drain tank in the TMI reactor. These descriptions are assigned since DECA is meant to work with a variety of processes, and thus there must be some specific tags assigned to the scenario numbers to enable DECA to interface with the system operators.

```
((1 "Pressurizer Leak ")
 (2 "Block Valve Leak ")
 (3 "Pipe Rupture - (drain tank) ")
 (4 "Drain Tank ")
 (5 "Pipe Rupture - Hot Leg, Primary Coolant System ")
 (6 "Pipe Rupture - Cold Leg, Primary Coolant System ")
 (7 "Reactor Pump ")
 (8 "Steam Generator - Primary Coolant System ")
 (9 "Steam Generator - Secondary Coolant System ")
 (10 "Pipe Rupture - Secondary Coolant System ")
 (11 "Feedwater Pump - Secondary Coolant System ")
 (12 "Turbine Trip - Secondary Coolant System ")
)
```

E.2 Parameter-expect.data File

This data file is used by DECA to determine the rank of the parameters. 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:

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

For example, for the following parameter QNT-P:

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

one can see that to have a rank of 10, the scenarios 2, 3, and 4 must be considered as possibilities by DECA. Also, if only scenarios 3 and 4 are a possibility, then DECA will give QNT-P a rank of 8. The second sublist is the hybrid part. In it one can see that if any one scenario (not scenario 1) from the list of scenarios for rank 10 is a possibility, then give QNT-P a rank of 4.3.

The second sublist of ranks is to prevent the need to list every possible combination of scenarios for each parameter (combinatorially explosive and computationally impossible if there are many scenarios in a large system). For example, if there were 100 parameters for the system under evaluation and on average there were 50 scenarios associated with each parameter, then there would be in the neighborhood of 50!*100 or $3.0414*10^{66}$ combinations. This is clearly unacceptable from a computational standpoint. Employing the hybrid system of reference, the system looses some subtle interrelationships between scenarios and parameters, but for the 100 parameter case there will probably be only 80 matches to make per parameter or 8000 total. See the function MAKE-PARAMETER-RANKING in the source code (Appendix F) for further explanation of the hybrid system.

Below is the data file contents of the expectancies for all parameters used in the TMI-2 test run.

```
(       (PZR-P ((10  (1 2 3 4))
                (8.5 (2 3 4))
                (8.5 (1 3 4))
                (6   (1 2 3))
                (6   (1 2 4))    )
                (
                 (4 2)
                 (2.2 1)          ))
        (PZR-L ((10  (1 2 3 4))
                (8.5 (2 3 4))
                (8.5 (1 3 4))
                (6   (1 2 3))
                (6   (1 2 4))    )
                (
                 (4 2)
                 (2.2 1)          ))
        (HL1-T ((10 (6 7 8))
                (8.5 (6 7))
                (8.5 (6 8))
                (7   (7 8))      )
               (
                (3 1)             ))
        (SG1-P ((10 (1 2 5 6 7 8 9 10 11 12))
                (9.8 (1 2 5 6 7 8))
                (8.5 (1 2 5 6 7))
                (8.5 (1 2 5 6 8))        )
               (
                (9 9)
                (8.5 8)
                (8.2 7)
        (8 6)
        (6 5)
        (5 4)
        (3 3)
        (2 2)
        (1 1)          ))
(SG1-L ((10 (1 2 5 6 8 9 10 11 12))
        (9.8 (1 2 5 6 8))
        (9  (1 2 5 6))          )
       (
        (9.3 8)
        (8.5 7)
        (7.5 6)
        (6   5)
        (4   4)
        (3   3)
        (1.5 2)
        (1   1)         ))
(QNT-P ((10 (2 3 4))
        (9.5 (2 4))
        (8.5 (2 3))
        (8   (3 4))      )
       (
        (4.3 1)         ))
(SG2-P ((10 (1 2 5 6 7 8 9 10 11 12))
        (9.8 (1 2 5 6 7 8))
        (8.5 (1 2 5 6 7))
        (8.5 (1 2 5 6 8))       )
       (
        (9 9)
        (8.5 8)
        (8.2 7)
        (8 6)
        (6 5)
        (5 4)
        (3 3)
        (2 2)
        (1 1)          ))
(SG2-L ((10 (1 2 5 6 8 9 10 11 12))
        (9.8 (1 2 5 6 8))
        (9  (1 2 5 6))          )
       (
        (9.3 8)
        (8.5 7)
        (7.5 6)
        (6   5)
        (4   4)
        (3   3)
        (1.5 2)
        (1   1)         ))
```

```
          (CL1-T ((10 (5 7 8 9 10 11 12))
                  (9.6 (5 7 8 9))
                  (9  (5 7 8))
                  (6  (5 7))      )
                 (
                  (9.2 6)
                  (8.6 5)
                  (8   4)
                  (4   3)
                  (2.5 2)
                  (1   1)          ))
      )
```

E.3 Scenario-tendency.data File

The file scenario-tendency.data contains the system knowledge of the expected parameter tendency for a given scenario to be true. The better the expected tendencies match the sensor data, the more likely that the scenario is actually occurring.

For example, suppose that from the Lookahead Mechanism, DECA suspects that scenario number 4 is possibly occurring. To verify this DECA looks at the required tendencies of the parameters associated with scenario 4.

```
(4  ((PZR-P LOWER)
     (PZR-L HIGHER)
     (QNT-P HIGHER)
        ))
```

For TMI-2 that would be PZR-P would be lower than the setpoint value, and PZR-L and QNT-P would both be running higher than their setpoint values. If everything matches up then scenario 4 would be considered one of the more likely explanations.

Below is the listing of the scenario-tendency.data file used for the TMI-2 runs on the DECA system.

```
(       (1  (    (PZR-P LOWER)
                 (PZR-L LOWER)
                 (SG1-P LOWER)
                 (SG1-L LOWER)
                 (SG2-P LOWER)
                 (SG2-L LOWER)))
        (2  (    (PZR-P LOWER)
                 (PZR-L LOWER)
                 (SG1-P LOWER)
                 (SG1-L LOWER)
                 (QNT-P HIGHER)
                 (SG2-P LOWER)
                 (SG2-L LOWER)))
```

```
              (3   (    (PZR-P LOWER)
                        (PZR-L LOWER)
                        (QNT-P LOWER)))
              (4   (    (PZR-P LOWER)
                        (PZR-L HIGHER)
                        (QNT-P HIGHER)))
              (5   (    (SG1-P LOWER)
                        (SG1-L LOWER)
                        (SG2-P LOWER)
                        (SG2-L LOWER)
                        (CL1-T LOWER)))
              (6   (    (HL1-T HIGHER)
                        (SG1-P HIGHER)
                        (SG1-L LOWER)
                        (SG2-P HIGHER)
                        (SG2-L LOWER)))
              (7   (    (HL1-T HIGHER)
                        (SG1-P HIGHER)
                        (SG2-P HIGHER)
                        (CL1-T HIGHER)))
              (8   (    (HL1-T HIGHER)
                        (SG1-P LOWER)
                        (SG1-L LOWER)
                        (SG2-P LOWER)
                        (SG2-L LOWER)
                        (CL1-T HIGHER)))
              (9   (    (SG1-P HIGHER)
                        (SG1-L HIGHER)
                        (SG2-P HIGHER)
                        (SG2-L HIGHER)
                        (CL1-T HIGHER)))
              (10   (   (SG1-P HIGHER)
                        (SG1-L LOWER)
                        (SG2-P HIGHER)
                        (SG2-L LOWER)
                        (CL1-T HIGHER)))
              (11   (   (SG1-P HIGHER)
                        (SG1-L LOWER)
                        (SG2-P HIGHER)
                        (SG2-L LOWER)
                        (CL1-T HIGHER)))
              (12   (   (SG1-P LOWER)
                        (SG1-L HIGHER)
                        (SG2-P LOWER)
                        (SG2-L HIGHER)
                        (CL1-T HIGHER)))
)
```

E.4 Scenario.data File

This data file contains all the parameters of the process which DECA is monitoring. For each parameter, there follows a list of scenarios which must be evaluated if that parameter is marked as being out of bounds. This list of scenarios is accessed by DECA's lookahead mechanism.

For example, if the evaluation of the sensor data for parameter PZR-P has determined that it is beyond its setpoint values (and marked as out of bounds), then DECA will access this data and determine that it must check scenarios 1, 2, 3, and 4 as being possible events occurring in the process.

```
PZR-P
(1 2 3 4)
PZR-L
(1 2 3 4)
HL1-T
(6 7 8)
SG1-P
(1 2 5 6 7 8 9 10 11 12)
SG1-L
(1 2 5 6 8 9 10 11 12)
QNT-P
(3 4)
SG2-P
(1 2 5 6 7 8 9 10 11 12)
SG2-L
(1 2 5 6 8 9 10 11 12)
CL1-T
(5 7 8 9 10 11 12)
```

E.5 Setpoint.data File

This file contains the values for each of the process parameters setpoints. DECA uses these setpoints to determine if the system parameter is in a normal operating state or if it is abnormal. If abnormal, DECA also uses the data to determine what is the severity of the parameter.

The first element of the file indicates the number of parameters in the system. Next listed is the parameter, then its setpoint mode (e.g. normal here). There can be more than one setpoint database on-line. The one used would correspond to systems operating condition (e.g. normal, reactor shutdown, refueling). Next listed is the units of measure of the data, and finally a list of the seven different setpoint values.

The data for the TMI-2 shown below was obtained from [Jow84].

```
9
PZR-P
NORMAL
PSIG
(1200 1900 2055 2150 2250 2355 2400)
PZR-L
NORMAL
INCHES
( 45 150 200 222 240 260 280)
HL1-T
NORMAL
F
( 300 400 500 606 610 619 630)
SG1-P
NORMAL
PSIG
( 800 850 900 940 1050 1070 1105)
SG1-L
```

```
NORMAL
INCHES
( 10 30 45 160 170 180 190)
QNT-P
NORMAL
PSIG
( 1 2 2.5 3 35 80 122)
SG2-P
NORMAL
PSIG
( 800 850 900 940 1050 1070 1105)
SG2-L
NORMAL
INCHES
( 10 30 45 160 170 180 190)
CL1-T
NORMAL
F
( 300 400 500 558 610 619 630)
```

E.6 Sensor.data File

The sensor.data file contains the actual measurements from the TMI-2 accident [Jow84]. The format is as follows; each line represents one time step. the first element in the list is the time, and the sublist is the readings for each of the nine parameters being monitored. The time is the seconds after turbine trip during the accident. The sensor data is always read in in the same order: PZR-P, PZR-L, HL1-T, SG1-P, SG1-L, QNT-P, SG2-P, SG2-L, CL1-T.

```
(       (0    (2145 218 607  944 123  3     930 116 559))
        (15   (2260 253 611 1022  79  6.3 1012  80 571))
        (30   (1905 182 587  998  26  7.8  987  30 577))
        (45   (1855 160 579 1000  17  9.3  993  20 576))
        (60   (1790 158 578  990  14 12    969  18 576))
        (75   (1760 162 577 1011  10 14.3  997  16 576))
        (90   (1725 175 578 1023  11 17.5 1005  16 577))
        (105  (1685 187 579 1021  11 19.6 1005  16 577))
        (120  (1650 200 579 1011  11 22.2 1000  16 579))
  )
```

Appendix F

DECA Program Listing

This appendix contains the source code for the DECA system.

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER; Base: 10 -*-
;;;
;;;-------------------------------------------------------------
;;;
;;; PROGRAMMER:       Steven R. Nann
;;;
;;; FILE:             h:>srn>thesis>deca.lisp
;;;
;;; DATE:             Jan - Mar  1988
;;;
;;; LANGUAGE:         Symbolics-Common-Lisp with Flavors
;;;
;;; MACHINE:          Symbolics 3670
;;;
;;; OPERATING SYSTEM: Genera 7.1
;;;
;;; COMMENTS:         This is the kernel of the DECA system
;;;                   software.
;;;
;;;-------------------------------------------------------------
;;;
;;;         COPYRIGHT 1988   -   ALL RIGHTS RESERVED
;;;
;;;-------------------------------------------------------------
;;;
;;; The DECA system is an Expert System for real time process control
;;; during routine, emergency and time constrained situations.
;;;
;;; This Inference Engine is built as a generic one for any dynamic
;;; domain.  The refernces of the parameters and scenarios just
;;; HAPPEN to be from the domian which DECA was tested.  One will
;;; find no refernce within the program to any TMI-2 parameter or
;;; scenario except maybe in the comments where an example of a
;;; Lisp structure is given and the three generate lisp forms.
;;;
;;;-------------------------------------------------------------
;;;
;;; Input parameters for the TMI-2 accident test scenario
;;;
;;; SYS-TIME System time
;;; PZR-P    Pressurizer Pressure
;;; PZR-L    Pressurizer Level
;;; HL1-T    Hot Leg 1 Temperature
;;; SG1-P    Steam Generator 1 Pressure
;;; SG1-L    Steam Generator 1 Level
;;; QNT-P    Drain Tank Pressure
;;; SG2-P    Steam Generator 2 Pressure
;;; SG2-L    Steam Generator 2 Level
;;; CL1-T    Cold Leg 1 Temperature
;;;
;;; Scenario numbers and descriptions.
;;;
;;;  1   Pressurizer Leak
;;;  2   Block Valve Leak
;;;  3   Pipe Rupture    - (drain tank)
;;;  4   Drain Tank
;;;  5   Pipe Rupture    - Hot Leg,  Primary Coolant System
;;;  6   Pipe Rupture    - Cold Leg, Primary Coolant System
;;;  7   Reactor Pump
;;;  8   Steam Generator - Primary Coolant System
;;;  9   Steam Generator - Secondary Coolant System
;;;  10  Pipe Rupture    - Secondary Coolant System
;;;  11  Feedwater Pump  - Secondary Coolant System
;;;  12  Turbine Trip    - Secondary Coolant System
;;;


;;;-------------------------------------------------------------
;;; Define the system (global) variables:
;;;-------------------------------------------------------------

(DEFVAR TEMP-IO-DATA NIL)       ;total inputs of all data
(DEFVAR SENSOR-DATA NIL)        ;cdr's inputs
(DEFVAR SYS-TIME NIL)                   ;time of sensor data
(DEFVAR SENSOR-RECORD NIL)              ;data for one time step
(DEFVAR OPERATION-FLAG 'NORMAL)         ;flag for operating cond
(DEFVAR PARAM-NUM NIL)                  ;number of system parameters
(DEFVAR SETPOINT-DATA NIL)              ;var of all setpoint data
(DEFVAR  SDB-LIST NIL)                  ;list of all setpoint data by
                                ;parameter.
```

```
(DEFVAR OOB-PARAMETERS NIL)            ;out of bounds parameters
(DEFVAR OOB-SEVERITY   NIL)            ;severity list of oob's
(DEFVAR OOB-PARAMETERS-VALUES NIL)   ;sensor value for oob params.

(DEFVAR SCENARIO-LIST  NIL)            ,param/scenario lookahead list
(DEFVAR LOOKAHEAD-SCENARIOS NIL)     ;scenarios to be checked out

(DEFVAR SCENARIO-EXPECTANCY NIL)     ;param tendencies for each scenario
(DEFVAR PARAMETERS-PER-SCENARIO-EXPECT NIL)  ;# of expected params, all scen.
(DEFVAR SCENARIO-DATA-MATCH-LIST NIL)          ;scenario # & params which match
                  ;with the expected tendency in scenario-expectancy
(DEFVAR SCENARIO-RATIO-MATCH-LIST NIL)    ;contains scen #, ratio, qual value
                                  ;eg ( (1 0.6 minor) ....)
(DEFVAR PARAMETER-EXPECTANCY NIL)  ;param, rank 1-10, and scenario combinations
                             ; which will give the param that rank.
(DEFVAR PARAMETER-RATIO-MATCH NIL)        ;list of lists of (parameter ratio)
                      ;where ratio is the #scenario match/#scenario expected.
(DEFVAR SCENARIO-MAJOR NIL)
(DEFVAR SCENARIO-MINOR NIL)
(DEFVAR SCENARIO-IMPROBABLE NIL)

(DEFVAR PARAMETER-RANK-LIST NIL)
(DEFVAR POSSIBLE-SCENARIO-FOR-SITUATION NIL)
(DEFVAR SCENARIO-DESCRIPTION NIL)

(DEFVAR FILE-SPEC NIL)
(DEFVAR OUTPUT-FILE-NAME NIL)

;;;=================================================================
;;; first read in the sensor data:
;;;=================================================================

;; Bring-in-system-data will read in the data for DECA's run.

(DEFUN BRING-IN-SYSTEM-DATA (INPUT-FILE)        ;BRING-IN-SYSTEM-DATA
 (LET ((TEMP-IO-DATA-LIST)
 )
    (SETQ INPUT-FILE (FS:PARSE-PATHNAME INPUT-FILE))
    (WITH-OPEN-FILE (SENSOR-INPUT INPUT-FILE
                     :DIRECTION :INPUT
                     :CHARACTERS T)
      (SETQ TEMP-IO-DATA-LIST (READ SENSOR-INPUT))
     )
    ;;(FORMAT T "~% DATA INPUT SUCESSFULLY ~%")
    (SETQ SENSOR-DATA TEMP-IO-DATA-LIST))
  ;(FORMAT T "SENSOR DATA IMPORTED SUCCESSFULLY. ~%")
  )


;;;=================================================================
;;; Set up the setpoint database - instances
;;;=================================================================

;; Read in the Setpoint Database which are stored in the files
;; h:>srn>thesis>setpoint.data
;;

(DEFFLAVOR SDB (PARAMETER
         MODE
         UNITS
         LLL
         LL
         L
         N
         H
         HH
         HHH)
    ()
  :READABLE-INSTANCE-VARIABLES
  :WRITABLE-INSTANCE-VARIABLES
  :INITABLE-INSTANCE-VARIABLES)

(DEFFLAVOR SDB-ALL (OPERATING-MODE
             DATA-LIST)
    ()
  :READABLE-INSTANCE-VARIABLES
  :WRITABLE-INSTANCE-VARIABLES
  :INITABLE-INSTANCE-VARIABLES)
```

```
(DEFUN MAKE-SETPOINT-DATABASE (INPUT-FILE)
  (LET ((TMP-DATA-LIST)
 (TMP-PARAM)
 (TMP-MODE)
 (TMP-UNIT)
 (SDB-PARAM)
 )
    (SETQ INPUT-FILE (FS:PARSE-PATHNAME INPUT-FILE))
    (WITH-OPEN-FILE (SDB-DATA INPUT-FILE
                     :DIRECTION :INPUT
                     :CHARACTERS T)
      (SETQ PARAM-NUM (READ SDB-DATA))
      (DOTIMES (I  PARAM-NUM)
        (SETQ TMP-PARAM (READ SDB-DATA)
              TMP-MODE  (READ SDB-DATA)
              TMP-UNIT  (READ SDB-DATA)
              TMP-DATA-LIST (READ SDB-DATA))
        (SETQ SDB-PARAM
               (MAKE-INSTANCE 'SDB
                              :PARAMETER TMP-PARAM
                              :MODE      TMP-MODE
                              :UNITS     TMP-UNIT
                              :LLL  (NTH 0 TMP-DATA-LIST)
                              :LL   (NTH 1 TMP-DATA-LIST)
                              :L    (NTH 2 TMP-DATA-LIST)
                              :N    (NTH 3 TMP-DATA-LIST)
                              :H    (NTH 4 TMP-DATA-LIST)
                              :HH   (NTH 5 TMP-DATA-LIST)
                              :HHH  (NTH 6 TMP-DATA-LIST)))
        (SETQ SDB-LIST (CONS SDB-PARAM SDB-LIST))
        ;(DESCRIBE SDB-LIST)
        )
      (SETQ SDB-LIST (REVERSE SDB-LIST))
      (SETQ SETPOINT-DATA (MAKE-INSTANCE 'SDB-ALL
                                   :OPERATING-MODE TMP-MODE
                                   :DATA-LIST      SDB-LIST))
      ;(DESCRIBE SETPOINT-DATA)
      )
    )
  )


;;;------------------------------------------------------------------
;;; Setup the Parameter/Scenario Database for the Lookahead Mech.
;;;------------------------------------------------------------------

(DEFFLAVOR SCENARIO (PARAMETER
               SCENARIOS)
     ()
  :READABLE-INSTANCE-VARIABLES
  :WRITABLE-INSTANCE-VARIABLES
  :INITABLE-INSTANCE-VARIABLES)

(DEFUN MAKE-PARAMETER-SCENARIO-DATABASE (INPUT-FILE)
  (LET ((TEMP-PARAM)
 (TEMP-SCENARIO)
 (TEMP-SCENARIO-LIST)
 )
    (SETQ INPUT-FILE (FS:PARSE-PATHNAME INPUT-FILE))
    (WITH-OPEN-FILE (SCENARIO-INPUT INPUT-FILE
                              :DIRECTION :INPUT
                              :CHARACTERS T)
      (DOTIMES (I  PARAM-NUM)
  (SETQ TEMP-PARAM    (READ SCENARIO-INPUT)
        TEMP-SCENARIO (READ SCENARIO-INPUT))

  (SETQ TEMP-SCENARIO-LIST (MAKE-INSTANCE 'SCENARIO
                                          :PARAMETER TEMP-PARAM
                                          :SCENARIOS TEMP-SCENARIO))
  (SETQ SCENARIO-LIST (CONS  TEMP-SCENARIO-LIST
                             SCENARIO-LIST))
  )
      )
    (SETQ SCENARIO-LIST (REVERSE SCENARIO-LIST))
    ))
```

```
;;;================================================================
;;; Compare the sensor data with the setpoint database
;;; and determine oob parameters & associated severity.
;;; Code correlates to most of DECA - diagnostic level 1's purpose.
;;;================================================================

;; Used when multiple setpoint databases. Will retrieve the
;; appropriate database depending on the operation-flag.
;;
(DEFMETHOD (GET-SDB SDB-ALL) (OPERATION-FLAG)          ;GET-SDB
  (COND ((EQUAL (SDB-ALL-OPERATING-MODE SELF) OPERATION-FLAG)
   (SDB-ALL-DATA-LIST SELF))
 )
  )

;; Used to check whether the parameter is out of bounds
;;
(DEFMETHOD (CHECK-OOB SDB) (PARAM-TMP-VALUE)           ;CHECK-OOB
  (FLAG-RULE LLL LL L H HH HHH PARAMETER PARAM-TMP-VALUE))

;; This checks and records oob parameters, their values, and
;; calls the function to determine the severity.
;;
(DEFUN FLAG-RULE
       (LLL LL L H HH HHH PARAM PARAM-TMP-VALUE)
  (COND ((OR (< PARAM-TMP-VALUE L)
      (> PARAM-TMP-VALUE H))
   (SETQ OOB-PARAMETERS
         (CONS PARAM OOB-PARAMETERS)
        OOB-PARAMETERS-VALUES
         (CONS PARAM-TMP-VALUE OOB-PARAMETERS-VALUES))
   (SEVERITY-RULE PARAM-TMP-VALUE LLL LL L H HH HHH)
   ))
  )

;; This will update the list of the severities which correlate to
;; the parameters in oob-parameters list.
;;
(DEFUN SEVERITY-SET-TO (Q-VALUE)                  ;SEVERITY-SET-TO
  (SETQ OOB-SEVERITY (CONS Q-VALUE OOB-SEVERITY)))

;; This fuction determines the level of severity (eg LL)
;; and calls severity-set-to function to update the list.
;;
(DEFUN SEVERITY-RULE (DATA LLL LL L H HH HHH)           ;SEVERITY-RULE
  (COND ((<= DATA L)
   (COND ((> DATA LL)   (SEVERITY-SET-TO 'L))
         ((> DATA LLL)  (SEVERITY-SET-TO 'LL))
         ((<= DATA LLL) (SEVERITY-SET-TO 'LLL))))
  ((>= DATA H)
   (COND ((< DATA HH)   (SEVERITY-SET-TO 'H))
         ((< DATA HHH)  (SEVERITY-SET-TO 'HH))
         ((>= DATA HHH) (SEVERITY-SET-TO 'HHH))))))

;; This function is the top level controller for the comparison of sensor
;; data and the values in the setpoint database.  Its purpose is to determine
;; the oob parameters and their level of severity for the given instant in time.
;;
(DEFUN COMPARE-SENSOR-DATA (S-DATA SETPOINT-DATA)
  (LET ((TEMP-DATA-LIST)
 (PARAM-TMP-VALUE)
 (PARAM-SETPOINT-LIST)
 )
    (SETQ TEMP-DATA-LIST SDB-LIST)
    (DOTIMES (I  PARAM-NUM)
      (SETQ PARAM-TMP-VALUE    (NTH I S-DATA)
     PARAM-SETPOINT-LIST (NTH I TEMP-DATA-LIST))
      (CHECK-OOB PARAM-SETPOINT-LIST PARAM-TMP-VALUE)
      )
    )
  (SETQ OOB-PARAMETERS (REVERSE OOB-PARAMETERS)
 OOB-PARAMETERS-VALUES (REVERSE OOB-PARAMETERS-VALUES)
 OOB-SEVERITY   (REVERSE OOB-SEVERITY))
  )
```

```
;;;===============================================================
;;; Gathering Lookahead scenario data
;;;===============================================================

;; This function will take the scenarios given and run thru the list
;; and add any scenario not in lookahead-scenarios list. It will then
;; sort the scenaios from smallest to largest.
;;
(DEFUN CHECK-SCENARIO-HERE (SCENARIOS-PICKED)
  (DOLIST (I SCENARIOS-PICKED )
    (COND ((NOT (MEMBER I LOOKAHEAD-SCENARIOS))
     (SETQ LOOKAHEAD-SCENARIOS
           (CONS I LOOKAHEAD-SCENARIOS))
     )) )
  (SETQ LOOKAHZAD-SCENARIOS (SORT LOOKAHEAD-SCENARIOS #'<))
  ;;(FORMAT T "~%lookahead-scenarios ~a " LOOKAHEAD-SCENARIOS)
  )

;; The function get-scenarios retrieves scenarios for lookahead
;; mechanism.  Input -> oob-parameters list (locally its oob-key)
;; checks with the instances of scenarios in the scenario-list
;; and pulls all scenarios for each oob parameter. These will then
;; be combined in one list (ie no repeats) with check-scenario-here
;; function.
;;New version

(DEFUN GET-SCENARIOS (OOB-KEY)
  (DOLIST (I OOB-KEY)
    (DOLIST (J SCENARIO-LIST)
      (GET-SCENARIOS-1 J I)
      )
    )
  )

(DEFMETHOD (GET-SCENARIOS-1 SCENARIO) (TEMP-OOB-PARAM)
  (LET ((TEMP-PARAMETER PARAMETER)
 (TEMP-SCENARIOS SCENARIOS)
 )
    (COND ((EQUAL TEMP-PARAMETER TEMP-OOB-PARAM)
     ;;(format t "~% temp-scenarios ~a" temp-scenarios)
     (CHECK-SCENARIO-HERE TEMP-SCENARIOS)
     ))
    )
  )
;;
;; After the above method is run all the scenarios that are needed for
;; the lookahead mechanism are contained in the global parameter
;; lookahead-scenarios.

;;;===============================================================
;;; Functions for Lookahead to see if the sensor data matches
;;; up with any of the scenario's expectations. It is done for
;;; each scenario contained in the global list  lookahead-scenarios.
;;;===============================================================

;; Bring in the scenario expectancy (tendency) data. It will come
;; in in the form of lists. One list/scenario. They will be put
;; into one global list scenario-expectancy. This list will be used
;; for all the checking of match-up through the prioritizer portion
;; of DECA.
;;
(DEFUN MAKE-SCENARIO-EXPECTANCY (INPUT-FILE)
    (SETQ INPUT-FILE (FS:PARSE-PATHNAME INPUT-FILE))
    (WITH-OPEN-FILE (TENDENCY-INPUT INPUT-FILE
                               :DIRECTION :INPUT
                               :CHARACTERS T)
      (SETQ SCENARIO-EXPECTANCY
     (READ TENDENCY-INPUT))
      )
    )
```

```
;; The next part will be to check the sensor-data with against the
;; expectancy of DECA.  It will use the lists scenario-expectancy
;; and lookahead-scenarios as well as oob-parameters and
;; oob-severity. If the sensor data for the p:rameter matches
;; up with the expectancy, then the parameter is added to
;; the list of matches for that scenario.
;;
(DEFUN MATCH-SCENARIO-TENDENCY ()

  (DOLIST (I LOOKAHEAD-SCENARIOS)
    (DOLIST (J SCENARIO-EXPECTANCY)
      (LET ((TEMP-NUM (CAR J))
     (TEMP-EXPECT (CADR J))
     )
 (IF (EQUAL I TEMP-NUM)
     (DOLIST (K TEMP-EXPECT)
        (LET* ((TEMP1-PARAMETER (CAR K))
         (SCENARIO-EXPECTANCY-DATABASE-DATA (CADR K))
         (TEMP2 (MEMBER TEMP1-PARAMETER OOB-PARAMETERS))
         (TEMP-COUNT)
         )
    ;; Now set the index so know where to look in the oob-severity list.
    ;; Note the oob-severity list ‹ata directly corresponds to the
    ;; parameter at the same location in the oob-parameter list.
    ;;
    (COND (TEMP2
           (SETQ TEMP-COUNT (- (LENGTH OOB-PARAMETERS)
                               (LENGTH TEMP2)))
           ;;Now go retrieve the tendency from the oob-severity list.
           ;;There are only 2 tendencies, higher or lower.
        (LET ((TEMP-SEV-EXPECT (NTH TEMP-COUNT OOB-SEVERITY)) ;ie LL
              (TEMP-EXPECTANCY-SENSOR-DATA)
              )
          (COND ((MEMBER TEMP-SEV-EXPECT '(LLL LL L))
                 (SETQ TEMP-EXPECTANCY-SENSOR-DATA 'LOWER))
                ((MEMBER 'EMP-SEV-EXPECT '(H HH HHH))
                 (SETQ TEMP-EXPECTANCY-SENSOR-DATA 'HIGHER))
                (T
                 (SETQ TEMP-EXPECTANCY-SENSOR-DATA NIL))
                )
          ;;
          ;;Now see if TEMP-ECPECTANCY-SENSOR-DATA matches with
          ;;the scenario-expectancy-database-data. If yes then mark as
          ;;one parameter that matches the expected tendency
          ;;for scenario i.
                                  ;if data matches predicted
          (IF (EQUAL TEMP-EXPECTANCY-SENSOR-DATA
                  SCENARIO-EXPECTANCY-DATABASE-DATA)

              ;; Run function to put `he scenario and parameter
              ;; which matches expectancy in its database.
              ;;
              (MARK-SCENARIO-PARAMETER-DATA I TEMP1-PARAMETER)

              ))) )
    )))
 )
      )
    )
  )   ;end match-scenario-tendency

;; This function will generate the default list for
;; scenario-data-match-list there are 12 scenarios at the moment
;;
(DEFUN GENERATE-LIST ()
  (SETQ SCENARIO-DATA-MATCH-LIST
  (LIST (LIST 1 NIL) (LIST 2 NIL)
        (LIST 3 NIL) (LIST 4 NIL)
        (LIST 5 NIL) (LIST 6 NIL)
        (LIST 7 NIL) (LIST 8 NIL)
        (LIST 9 NIL) (LIST 10 NIL)
        (LIST 11 NIL) (LIST 12 NIL)))
  )
```

```
;; This function will add the scenario and its parameter
;; which agrees with its expectancy to the global list
;; scenario-data-match-list.
;;
(DEFUN MARK-SCENARIO-PARAMETER-DATA (SCENARIO-NUMBER ASSOCIATED-PARAMETER)

  (COND ((NULL SCENARIO-DATA-MATCH-LIST)
  (GENERATE-LIST)
  ))
  (DOLIST (DATA-RECORD SCENARIO-DATA-MATCH-LIST)
    (COND ((EQUAL SCENARIO-NUMBER (CAR DATA-RECORD))
    (LET ((TEMP1-LIST (CADR DATA-RECORD))
          )
      (SETQ TEMP1-LIST
            (CONS ASSOCIATED-PARAMETER TEMP1-LIST))

      (SETF (CADR DATA-RECORD)            ;change old param list to
            TEMP1-LIST)                   ;the new consed list
      ))
   )))

;;;This ends the checking of parameter/scenario expectancies and updating
;;;the appropriate data-lists.

(DEFFLAVOR SCENARIO-PARAMETER-MATCH (SCENARIO-NUM
                              MATCH-PARAMETER)
    ()
  :READABLE-INSTANCE-VARIABLES
  :WRITABLE-INSTANCE-VARIABLES
  :INITABLE-INSTANCE-VARIABLES)



;;;============================================================
;;; Determine the qualitative match for the lookahead scenarios
;;;     eg Major, Minor, etc.
;;;============================================================

;; This function will take the scenario-expectancy list and
;; determine the number of parameters that are expected to
;; match under ideal conditions for each scenario. It then inserts
;; the results in the global list parameters-per-scenario-expect.
;
(DEFUN MAKE-LIST-OF-NUM-PARAMS-EXPECTED ()
  (DOLIST (J SCENARIO-EXPECTANCY)
    (LET* ((TEMP1 (CAR J))
     (TEMP2 (CADR J))
     (TEMP3 (LENGTH TEMP2))
     )
      (SETQ PARAMETERS-PER-SCENARIO-EXPECT
      (CONS (LIST TEMP1 TEMP3)
            PARAMETERS-PER-SCENARIO-EXPECT))
      ))
  (SETQ PARAMETERS-PER-SCENARIO-EXPECT
  (SORT PARAMETERS-PER-SCENARIO-EXPECT #'< :KEY #'CAR))
  )

;; Function to generate the raw list scenario-match-ratio
;;  set up for 12 scenarios at the moment.
(DEFUN GENERATE-RATIO-LIST ()
  (SETQ SCENARIO-RATIO-MATCH-LIST
  (LIST (LIST 1)  (LIST 2)
        (LIST 3)  (LIST 4)
        (LIST 5)  (LIST 6)
        (LIST 7)  (LIST 8)
        (LIST 9)  (LIST 10)
        (LIST 11) (LIST 12))
  )
  )
```

```
;; Now make the qualitative match for the scenario.
;; At the moment there are 3 levels of match.
;;     scenario-match ratio       qualitative-match
;;         0.0 <= x < 0.3             improbable
;;         0.3 <= x < 0.75            minor
;;        0.75 <= x <= 1              major
;; the results will then be put into another list
;; scenario-ratio-match-list
;;
(DEFUN SCENARIO-QUAL-MATCH ()
  (SETQ SCENARIO-DATA-MATCH-LIST
 (SORT SCENARIO-DATA-MATCH-LIST #'< :KEY #'CAR))
  (IF (NULL SCENARIO-RATIO-MATCH-LIST)
      (GENERATE-RATIO-LIST))
  (DOLIST (I SCENARIO-DATA-MATCH-LIST)
    (LET* ((TEMP-DATA-LENGTH (LENGTH (CADR I)))
    (TEMP-EXPECT-LENGTH                      ;assume it is sorted
      (CADR (NTH (1- (CAR I)) PARAMETERS-PER-SCENARIO-EXPECT)))
    ;figure percentage match with expected
    (TEMP-RATIO (/ TEMP-DATA-LENGTH TEMP-EXPECT-LENGTH))
    (TEMP-QUAL-VALUE)
    )
      ;set qualitative match value for the scenario under evaluation
      (COND ((< TEMP-RATIO 0.3)
      (SETQ TEMP-QUAL-VALUE 'IMPROBABLE))
     ((< TEMP-RATIO 0.75)
      (SETQ TEMP-QUAL-VALUE 'MINOR))
     ((<= TEMP-RATIO 1.0)
      (SETQ TEMP-QUAL-VALUE 'MAJOR)))
      ;modify the list to include the new data of the ratio and the
      ;qualitative value of the match.
      (SETF (NTH (1- (CAR I)) SCENARIO-RATIO-MATCH-LIST)
     (APPEND (NTH (1- (CAR I)) SCENARIO-RATIO-MATCH-LIST)
            (,TEMP-RATIO ,TEMP-QUAL-VALUE)))
      )  ;end let*
    )
  )

;;;================================================================
;;; Sort the scenarios into their appropriate list depending on
;;; what their qualitative values are for the scenario/parameter
;;; matching.
;;;================================================================

;; This function will create three new global lists:
;;       scenario-major
;;       scenario-minor
;;       scenario-improbable
;; which will contain all the scenario #'s and are sorted with
;; the largest ratio'd scenarios first.  It will be used later
;; to determine parameter priorities.
;;
(DEFUN SPLIT-INTO-MAJ-MINOR ()
  (DOLIST (I SCENARIO-RATIO-MATCH-LIST)
    (LET ((TEMP-SCEN (CAR I))
   (TEMP-RATIO (NTH 1 I))
   (TEMP-QUAL  (NTH 2 I))
   )
      (COND ((EQUAL TEMP-QUAL 'IMPROBABLE)
      (SETQ SCENARIO-IMPROBABLE (CONS (LIST TEMP-SCEN TEMP-RATIO)
                                      SCENARIO-IMPROBABLE)))
     ((EQUAL TEMP-QUAL 'MINOR)
      (SETQ SCENARIO-MINOR (CONS (LIST TEMP-SCEN TEMP-RATIO)
                                 SCENARIO-MINOR)))
     ((EQUAL TEMP-QUAL 'MAJOR)
      (SETQ SCENARIO-MAJOR (CONS (LIST TEMP-SCEN TEMP-RATIO)
                                 SCENARIO-MAJOR)))
     ) ;end cond
      ))
  (SETQ SCENARIO-IMPROBABLE (SORT SCENARIO-IMPROBABLE #'> :KEY #'CADR)
 SCENARIO-MINOR      (SORT SCENARIO-MINOR      #'> :KEY #'CADR)
 SCENARIO-MAJOR      (SORT SCENARIO-MAJOR      #'> :KEY #'CADR))

    )            ;end function
```

```
;;;============================================================
;;; Functions to determine the system's Parameters Priority
;;; It will dump the results into the parameter's database.
;;;============================================================

;; This function is for importing the parameter's ranking data file.
;; What it is is a file with each of the system parameters, and
;; the ranking of importance [1..10] 10 being most important, for
;; all the given combinations of scenarios.  For example, if for
;; PZR-P all the for it (1, 2, 3, 4) all had a major qual. match
;; value (ie all the parameters matched well with the scenarios),
;; then the prioity given to PZR-P would be 10, since all data
;; correlates "perfectly" with the expected on both the scenario
;; side, and the parameter side.
;;
;;
(DEFUN MAKE-PARAMETER-EXPECTANCY (INPUT-FILE)
  ; input file h:>srn>thesis>parameter-expect.data
    (SETQ INPUT-FILE (FS:PARSE-PATHNAME INPUT-FILE))
    (WITH-OPEN-FILE (PARAM-INPUT INPUT-FILE
                           :DIRECTION :INPUT
                           :CHARACTERS T)
      ;; Note in the data file the whole thing is one big
      ;; Lisp form... this case a list.
      ;;
      ;; The set up of parameter-expectancy is as follows:
      ;;  ( (pzr-p  ( (10 (1 2 3 4))
      ;;              (8  (2 3 4))
      ;             )
      ;             ( (6 3)     ;for the hybrid version
      ;               (3 2)
      ;               (1 1)))
      ;;              etc ) )
      ;     (pzr-l  ( (10 (3 4 6 8)
      ;               (6  (3 8)
      ;               ... ))
      ;    ....etc ...)
      ;;
      (SETQ PARAMETER-EXPECTANCY
      (READ PARAM-INPUT))
      )
    )             ;End fun

;; This function initializes the list parameter-ratio-match
;;
(DEFUN GENERATE-PARAM-RATIO-MATCH-LIST ()
  (SETQ PARAMETER-RATIO-MATCH
  (LIST (LIST 'PZR-P NIL)  (LIST 'PZR-L NIL)
        (LIST 'HL1-T NIL)  (LIST 'SG1-P NIL)
        (LIST 'SG1-L NIL)  (LIST 'QNT-P NIL)
        (LIST 'SG2-P NIL)  (LIST 'SG2-L NIL)
        (LIST 'CL1-T NIL))
  )
  )

;; This function updates the list parameter-ratio-data and inputs
;; the lists of rank-10 scenarios, major scen., etc.
;; while looping through, it updates upon match of parameter key
;; using the setf function.
;;
(DEFUN MARK-PARAMETER-RATIO-DATA (PARAMETER RATIO-LIST)
  (COND ((NULL PARAMETER-RATIO-MATCH)
   (GENERATE-PARAM-RATIO-MATCH-LIST)
   ))     ;end cond
  (DOLIST (DATA-RECORD PARAMETER-RATIO-MATCH)
    (COND ((EQUAL PARAMETER (CAR DATA-RECORD))
     (LET ((TEMP1-LIST)
           )
       (SETQ TEMP1-LIST
             (APPEND  TEMP1-LIST RATIO-LIST))
       (SETF (CADR DATA-RECORD)           ;change old param list to
             TEMP1-LIST)                  ;the new appended list
       ))
    )
    )
  )
```

```
;; Now take each parameter [eg pzr-p] and compare its expected
;; scenarios with the scenarios of valid matching from
;; scenario-ratio-match data.
;; Put the results in parameter-ratio-match database. (Similar
;; to scenario-ratio-match). Ratio is # scenarios that are good /
;; # scenarios expected by parameter-expectancy.
;;   This is much like the Scenario-Qual-Match function.

(DEFUN MAKE-PARAMETER-COMPARISON ()
  (GENERATE-PARAM-RATIO-MATCH-LIST)
    (DOLIST (I OOB-PARAMETERS)
      (DOLIST (K PARAMETER-EXPECTANCY)
 (COND ((EQUAL (CAR K) I)
        (LET ((TEMP-10-SCAN (CADR (CAADR K)))
              (TEMP-SC-MAJ)
              (TEMP-SC-MIN)
              (TEMP-SC-IMP)
              (TEMP-LIST)
              )
 ;; loop thru the rank 10 scenarios and compare what qual.
 ;; match they have. Add the scenario to the approp. list.
 ;;
 (DOLIST (J TEMP-10-SCAN)
   (DOLIST (M SCENARIO-MAJOR)
     (IF (EQUAL J (CAR M)) (SETQ TEMP-SC-MAJ (CONS J TEMP-SC-MAJ))
         ))
   (DOLIST (M SCENARIO-MINOR)
     (IF (EQUAL J (CAR M)) (SETQ TEMP-SC-MIN (CONS J TEMP-SC-MIN))
         ))
   )              ;end dolist j
 (SETQ TEMP-LIST
       (LIST (LIST (LENGTH TEMP-10-SCAN))
             TEMP-SC-MAJ
             TEMP-SC-MIN
             TEMP-SC-IMP))
 ;; Update the list parameter-ratio-match using setf by adding
 ;; in the values in temp-list to the data field of the parameter
 ;; -ratio-match list. The mark-parameter-ratio-data function
 ;; performs this update.
 ;;
 (MARK-PARAMETER-RATIO-DATA I TEMP-LIST)
 )                ;end let
 ))               ;end cond
 )                ;end dolist K
     )                    ;end dolist I
    ;; Sorts out the parameters by rank
    ;;
    (MAKE-PARAMETER-RANKING)
    )    ;end

;; Now, that the lists are sorted, DECA will put the parameters
;; into priority  according to the conclusions derived from the
;; data thus far.
;;
(DEFUN MAKE-PARAMETER-RANKING ()
  ;; Sort thru the parameter-ratio-match list
  (DOLIST (I PARAMETER-RATIO-MATCH)
    (LET ((TEMP-FLAG-FOR-MATCH)
   (TEMP-PARAM (CAR I))
   (TEMP-SCENARIOS (RETRIEVE-SCENARIOS I))    ;do not want 10-rank in it
    )
    ;; note that temp-scenarios contents are already sorted.
    ;; So they are ready for the match with the parameter-expectancy
    (DOLIST (PARAM-EXPT PARAMETER-EXPECTANCY)
      (COND ((EQUAL (CAR PARAM-EXPT) TEMP-PARAM)
       (LET ((PARAMETER-TEMPLATE (CADR PARAM-EXPT)) )
         (DOLIST (J PARAMETER-TEMPLATE)
           (LET ((TEMP-RANK (CAR J))
                 (TEMP-RECORD (CADR J))
                 )
             (IF (EQUAL TEMP-SCENARIOS TEMP-RECORD)
                 (SETQ TEMP-FLAG-FOR-MATCH T
                       PARAMETER-RANK-LIST
                       (CONS (LIST TEMP-PARAM TEMP-RANK)
                             PARAMETER-RANK-LIST))
                 ) ) ) )
      ;Code for the hybrid system.
      ;Now if DECA hasn't received a rank yet, (ie fail at the most
      ;significant matches), then the function switches to a mode
      ;which isn't combinatorially explosive. [not n factorial
      ;combinations to search thru & impossible for real time.]
      ;The hybrid looses some of the subtle knowledge of parameter
      ;scenario interrelationships, but not all. It only covers the
      ;most significant ones as dictated by the knowledge base.
      ;    To do the next part use a third sublist in
      ;parameter-expectancy which contains, a rank x and the number
      ;of scenarios needed to fit (ie 3 of the 7 in 10-rank). This
      ;eliminates the need to search all the combinations.
      ;
      (COND ((NULL TEMP-FLAG-FOR-MATCH)
             (LET ((SCEN-NUM-TEMPLATE (CADDR PARAM-EXPT)))
                (DOLIST (K SCEN-NUM-TEMPLATE)
                  (LET ((TEMP-RANK (CAR K))
                        (TEMP-SC-NUM (CADR K)))
                    (IF (EQUAL TEMP-SC-NUM (LENGTH TEMP-SCENARIOS))
                        (SETQ PARAMETER-RANK-LIST
                              (CONS (LIST TEMP-PARAM TEMP-RANK)
                                    PARAMETER-RANK-LIST)))
                    )))
              ; Just in case there was not any match via hybrid
              (IF (AND (MEMBER TEMP-PARAM OOB-PARAMETERS)
                       (NOT (EQUAL (CAAR PARAMETER-RANK-LIST)
                                   TEMP-PARAM)))
                  (SETQ PARAMETER-RANK-LIST
                        (CONS (LIST TEMP-PARAM 1) PARAMETER-RANK-LIST)))
              ))
      ))
    )   )
    )
  ;; At this point the oob-parameters have all been assigned a
  ;; rank. Now DECA can assign the priority according to the rank.
  ;; In the list parameter-rank-list it contains a bunch of conses of
  ;; (parameter . rank). Now sort thru these sublists and put in
  ;; descending order according to rank.
  (SETQ PARAMETER-RANK-LIST (SORT PARAMETER-RANK-LIST #'> :KEY #'CADR))

  )          ;end make-parameter-ranking


;; Used to get all the scenarios of the parameter under consideration.
;;
(DEFUN RETRIEVE-SCENARIOS (I)
  (LET* ((TEMP-DATA (CDADR I))
 (TEMP-MAJ (FIRST TEMP-DATA))
 (TEMP-MIN (SECOND TEMP-DATA))
 (TEMP-IMP (THIRD  TEMP-DATA))
 (TEMP-ALL)
 )
    (SETQ TEMP-ALL
   (SORT (APPEND TEMP-MAJ TEMP-MIN TEMP-IMP) #'<))
    )
  )

;;;===========================================================
;;; Qualitatively refine parameter sort
;;;===========================================================

;; Using the list parameter-rank-list DECA will see if the rank is
;; is appropriate given the levels of confidence in the likelyhood
;; of the scenario occurring.
;;
(DEFUN REFINE-PARAMETER-RANK-TOP ()
  (SETQ PARAMETER-RANK-LIST
 (REFINE-PARAMETER-RANK PARAMETER-RANK-LIST))
  (SETQ PARAMETER-RANK-LIST
 (SORT PARAMETER-RANK-LIST #'> :KEY #'GET-RANK-INDEX))
  )

;;
(DEFUN GET-RANK-INDEX (X)
  (CAR (LAST X))
  )
```

```
;; Refine-parameter-rank function will evaluate the list of
;; parameters and their associated ranks and refines the ranking
;; of parameters which have the same rank by checking their
;; severity. The worse the severity the more priority given to
;; the parameter for a given rank.
;;
(DEFUN REFINE-PARAMETER-RANK (RANK &OPTIONAL (NEW-LIST NIL))
  (LET ((TEMP1 (CAR RANK))
 (TEMP2 (CADR RANK))
 (TEMP3 (CDDR RANK))
 )
    (COND ((NULL TEMP2)
    (SETQ NEW-LIST (CONS TEMP1 NEW-LIST))
    (SETQ NEW-LIST (REVERSE NEW-LIST))
    NEW-LIST)
   (T (COND ((EQUAL (LAST TEMP1) (LAST TEMP2))
             (SETQ TEMP1
                   (CONS (CAR TEMP2) TEMP1))
             (SETQ RANK
                   (CONS TEMP1 TEMP3))
             ;;(FORMAT T "~%~a "RANK)
             (REFINE-PARAMETER-RANK RANK NEW-LIST))

            ; Now if 2 ranks not the same.
            (T
             (SETQ NEW-LIST (CONS TEMP1 NEW-LIST)
                   RANK     (CONS TEMP2 TEMP3))
             ;;(FORMAT T "~% new-list ~a")
             ;;(FORMAT T "~%     ~a"RANK)
             (REFINE-PARAMETER-RANK RANK NEW-LIST))
            )                            ;end cond
       ))                                ;end cond
    )
  )

;; Now have the new list of parameters which are grouped by common
;; rank. In order-multiple it will loop through the list to see if
;; length is > 2 (ie more than 1 param with a given rank). It then
;; calls change-order which will compare these rank's severity and
;; order them accordingly.  Then the function loose-parentheses is
;; called to clean up the list and return back to the original list
;; except that the parameters have been refined for each rank.
;;
(DEFUN ORDER-MULTIPLES ()
  (DOLIST (I PARAMETER-RANK-LIST)
    (COND ((> (LENGTH I) 2)
     (LET ((TEMP-1 (CHANGE-ORDER I))
           )
       (SETQ PARAMETER-RANK-LIST          ;don't use setf here
             (SUBST TEMP-1 I PARAMETER-RANK-LIST))
       ;;(SETF I TEMP-1)
       )))
    )
  ; remove redundant ()'s
  (SETQ PARAMETER-RANK-LIST
 (LOOSE-PARENTHESES PARAMETER-RANK-LIST))
  )

(DEFUN CHANGE-ORDER (I)
  (LET ((TEMP-LIST (ZL:FIRSTN (1- (LENGTH I)) I))
 (TEMP-RANK (CAR (LAST I)))
 )
    (DOLIST (J TEMP-LIST)
      (LET* ((TEMP-PLACE (- (LENGTH OOB-PARAMETERS)
                     (LENGTH (MEMBER J OOB-PARAMETERS))))
       (TEMP-SEVERITY (NTH TEMP-PLACE OOB-SEVERITY))
       )
 (COND ((OR (EQUAL TEMP-SEVERITY 'LLL)
             (EQUAL TEMP-SEVERITY 'HHH))
        (SETQ TEMP-LIST (SUBST (LIST J 3) J TEMP-LIST)))
        ;; old (SETF J (LIST J 3)))
       ((OR (EQUAL TEMP-SEVERITY 'LL)
             (EQUAL TEMP-SEVERITY 'HH))
        (SETQ TEMP-LIST (SUBST (LIST J 2) J TEMP-LIST)))
       ((OR (EQUAL TEMP-SEVERITY 'L)
             (EQUAL TEMP-SEVERITY 'H))
        (SETQ TEMP-LIST (SUBST (LIST J 1) J TEMP-LIST)))
       )
 ))                      ;end dolist
```

```
    (SETQ TEMP-LIST (SORT TEMP-LIST #'> :KEY #'CADR))

    (DOLIST (K TEMP-LIST)
      (SETF (CADR K) TEMP-RANK)

      )
    ;return the new and improved sublist to function order-multiple.
    TEMP-LIST
    )
  )

;; Now the variable would have the following format:
;;     ((a 10) ((d 9)(b 9)(c 9)) (e 3)...)
;; Need to remove the redundant () on params w/same ranks
;;
(DEFUN LOOSE-PARENTHESES (RANK)
  (LET ((TEMP-LIST))
    (DOLIST (I RANK)
      (COND ((LISTP (CAR I))
       (DOLIST (J I)
         (SETQ TEMP-LIST (CONS J TEMP-LIST))
         ))
     (T
      (SETQ TEMP-LIST (CONS I TEMP-LIST))
       ))
      )
    (SETQ RANK (REVERSE TEMP-LIST))
    )
  RANK
  )


;;;================================================================
;;; Qualitatively sort scenarios
;;;================================================================

;; Set up the list possible-scenario-for-situation, which will
;; contain the scenarios which have a reasonable possibility to
;; be the actual scenario. (ie maj or min scenarios). More
;; refinment may be necessary if there is less than great match
;; (agreement) with params and scenarios.

(DEFUN PUT-SCENARIOS-TOGETHER ()
  (SETQ POSSIBLE-SCENARIO-FOR-SITUATION
 (APPEND SCENARIO-MAJOR SCENARIO-MINOR))
  (SETQ POSSIBLE-SCENARIO-FOR-SITUATION
 (SORT POSSIBLE-SCENARIO-FOR-SITUATION #'> :KEY #'CADR))
  )

;;;================================================================
;;; Run context solution searches and output results,
;;;   1 - Scenario
;;;   2 - Parameters and their priority
;;;   3 - Suggested action to alleviate the situation
;;;================================================================


;; Read in the data from disk for the scenario number and its
;; associated description.
;; file - h:>srn>thesis>scenario-descriptions.data
;;
(DEFUN MAKE-SCENARIO-DESCRIPTION (INPUT-FILE)
    (SETQ INPUT-FILE (FS:PARSE-PATHNAME INPUT-FILE))
    (WITH-OPEN-FILE (DESCRIPTION-INPUT INPUT-FILE
                              :DIRECTION :INPUT
                              :CHARACTERS T)
      ;; Note in the data file the whole thing is one big
      ;; Lisp form... this case a list.
      ;;
      (SETQ SCENARIO-DESCRIPTION
     (READ DESCRIPTION-INPUT))
      )
    )             ;End fun
```

```
;; This function will control all the output to the user
;;
(DEFUN OUTPUT-CONTROL ()
  (COND ((NULL SCENARIO-MAJOR)
   (FORMAT T "~%DECA's conclusions for system time: ~a ~%" SYS-TIME)
   (OUTPUT-RESULTS-WITHOUT-SCENARIO)
   (FORMAT T "~%End of data evaluation for system time: ~a ~%" SYS-TIME)
   )
  (T
   (FORMAT T "~%DECA's conclusions for system time: ~a ~%" SYS-TIME)
   (OUTPUT-RESULTS-WITH-SCENARIO)
   (FORMAT T "~%End of data evaluation for system time: ~a ~%" SYS-TIME)
   ))
  )

(DEFUN OUTPUT-RESULTS-WITHOUT-SCENARIO ()
  ;first the scenario
    ;output
    (FORMAT T "~% No scenario selected, not confident enough. ~%")
    (FORMAT T "~% The parameter and priorities are as follows: ~2%")
    (DOLIST (I PARAMETER-RANK-LIST)
      (LET ((PARAM (CAR I))
     (RANK  (CADR I))
      )
  (FORMAT T "~3T~a ~16T~a ~%" PARAM RANK)
  ))
    ;;(WRITE PARAMETER-RANK-LIST :PRETTY :ALIST)
    (FORMAT T "~2% Scenarios that were considered as possible choices ~%")
    (FORMAT T " but not selected are: ~2%")
    (FORMAT T " Scenario   Ratio    Description  ~2%")
    (DOLIST (SCENARIO-NUM POSSIBLE-SCENARIO-FOR-SITUATION)
      (LET ((TEMP-SCEN-NUM (CAR SCENARIO-NUM))
     (TEMP-RATIO    (CADR SCENARIO-NUM))
     (TEMP-DESCRIPTION )
      )
  (DOLIST (J SCENARIO-DESCRIPTION)
    (IF (EQUAL (CAR J) TEMP-SCEN-NUM)
        (SETQ TEMP-DESCRIPTION (CADR J)))
    )
  (FORMAT T "~3T ~a ~13T~A ~20T ~a~%"
          TEMP-SCEN-NUM
          TEMP-RATIO
          TEMP-DESCRIPTION)
      ))
    (FORMAT T "~2%")
    ;;(WRITE POSSIBLE-SCENARIO-FOR-SITUATION :PRETTY :ALIST)
  )

(DEFUN OUTPUT-RESULTS-WITH-SCENARIO ()
  ;first the scenario
  (LET ((TEMP-SCENARIO-SPECIFICS )
  )
    ;output
    (FORMAT T "~% Scenario selected is; ~%")
    (SETQ TEMP-SCENARIO-SPECIFICS (GET-GUESS))
    (FORMAT T " Scenario Number ~a ~%" (FIRST TEMP-SCENARIO-SPECIFICS))
    (FORMAT T " Scenario Description ~d ~%" (LAST TEMP-SCENARIO-SPECIFICS))
    (FORMAT T " Confidence  ~a ~2%" (SECOND TEMP-SCENARIO-SPECIFICS))
    (FORMAT T "~% The parameter and priorities are as follows: ~2%")
    (DOLIST (I PARAMETER-RANK-LIST)
      (LET ((PARAM (CAR I))
     (RANK  (CADR I))
      )
  (FORMAT T "~3T~a ~16T~a ~%" PARAM RANK)
  ))
    ;;(WRITE PARAMETER-RANK-LIST :PRETTY :ALIST)
    (FORMAT T "~2% Scenarios that were considered as possible choices ~%")
    (FORMAT T " but not selected are: ~2%")
    (FORMAT T " Scenario   Ratio    Description  ~2%")
    (DOLIST (SCENARIO-NUM POSSIBLE-SCENARIO-FOR-SITUATION)
      (LET ((TEMP-SCEN-NUM (CAR SCENARIO-NUM))
     (TEMP-RATIO    (CADR SCENARIO-NUM))
     (TEMP-DESCRIPTION )
      )
  (DOLIST (J SCENARIO-DESCRIPTION)
    (IF (EQUAL (CAR J) TEMP-SCEN-NUM)
        (SETQ TEMP-DESCRIPTION (CADR J)))
    )
```

```
 (FORMAT T "~3T ~a ~13T~A ~20T ~a~%"
         TEMP-SCEN-NUM
         TEMP-RATIO
         TEMP-DESCRIPTION)
     ))
    (FORMAT T "~2%" )
    ;(WRITE (CDR POSSIBLE-SCENARIO-FOR-SITUATION) :PRETTY :ALIST)
    )
  )

(DEFUN GET-GUESS ()
  (LET ((TEMP-SCENARIO-INFO)
 (TEMP-NUMBER)
 (TEMP-DESCRIPTION))
    (SETQ TEMP-SCENARIO-INFO  (CAR POSSIBLE-SCENARIO-FOR-SITUATION) ;eg (1 0.75)
   TEMP-NUMBER          (CAR TEMP-SCENARIO-INFO))
    (DOLIST (I SCENARIO-DESCRIPTION)
      (IF (EQUAL (CAR I) TEMP-NUMBER)
   (SETQ TEMP-DESCRIPTION (CADR I)))
      )
    ;return the results
    (APPEND TEMP-SCENARIO-INFO (LIST TEMP-DESCRIPTION))
    )
  )

;;;=================================================================
;;; Function for resetting variables for each loop
;;;=================================================================

(DEFUN RESET-AND-REGENERATE-GLOBALS ()
  ;
  (SETQ  OOB-PARAMETERS                      NIL
  OOB-SEVERITY                      NIL
  OOB-PARAMETERS-VALUES             NIL
  LOOKAHEAD-SCENARIOS               NIL
  PARAMETERS-PER-SCENARIO-EXPECT     NIL
  SCENARIO-MAJOR                    NIL
  SCENARIO-MINOR                    NIL
  SCENARIO-IMPROBABLE               NIL
  PARAMETER-RANK-LIST               NIL
  POSSIBLE-SCENARIO-FOR-SITUATION   NIL)

  (GENERATE-LIST )                                 ; SCENARIO-DATA-MATCH-LIST
  (GENERATE-RATIO-LIST)                            ; SCENARIO-RATIO-MATCH-LIST
  (GENERATE-PARAM-RATIO-MATCH-LIST)                ; PARAMETER-RATIO-MATCH

  )   ;end reset...

(DEFUN RESET-AT-END-OF-LOOP ()
  (SETQ SDB-LIST                NIL
 SETPOINT-DATA         NIL
 SCENARIO-LIST         NIL
 SCENARIO-DESCRIPTION  NIL
 PARAMETER-EXPECTANCY  NIL
 SCENARIO-EXPECTANCY   NIL)
  )

;;;=================================================================
;;;MAIN-LOOP FOR THE DECA SYSTEM
;;;=================================================================

(DEFUN DECA ()
  (RESET-AT-END-OF-LOOP)
  ;; setup sensor-data from file
  (BRING-IN-SYSTEM-DATA "h:>srn>thesis>sensor.data")
  ;; setup setpoint database
  (MAKE-SETPOINT-DATABASE "h:>srn>thesis>setpoint.data")
  ;; setup parameter/scenario database
  (MAKE-PARAMETER-SCENARIO-DATABASE "h:>srn>thesis>scenario.data")
  ;;
  (MAKE-SCENARIO-EXPECTANCY "h:>srn>thesis>scenario-tendency.data")
  ;;
  (MAKE-PARAMETER-EXPECTANCY "h:>srn>thesis>parameter-expect.data")
  ;;
  (MAKE-SCENARIO-DESCRIPTION "h:>srn>thesis>scenario-descriptions.data")
  ;; now begin to loop for each time step that deca is evaluating
  ;;
  (IF (NOT (EQUAL SENSOR-DATA NIL))       ;check if out of data
  ;;
```

```
;; now make a run through DECA for the time record sensor-record
   (DOLIST  (I SENSOR-DATA)
      (SETQ SYS-TIME (CAR I))
      (SETQ SENSOR-RECORD (CADR I))
      (FORMAT T "~%Intermediate parameters for system time: ~a ~%" SYS-TIME)
      (FORMAT T "~%Sensor-record  ~a~%" SENSOR-RECORD)
      (RESET-AND-REGENERATE-GLOBALS)
      (COMPARE-SENSOR-DATA SENSOR-RECORD SDB-LIST)      ;sets oob
  (FORMAT T "Oob-parameters ~a~%Oob-parameters-values ~a~%Oob-severity ~A~2%"
        OOB-PARAMETERS OOB-PARAMETERS-VALUES OOB-SEVERITY)
  ;;
(GET-SCENARIOS OOB-PARAMETERS)
(FORMAT T "Lookahead-scenarios  ~a~2%" LOOKAHEAD-SCENARIOS)
  ;; match tendencies with expected
(MATCH-SCENARIO-TENDENCY)
(FORMAT T "Scenario-data-match-list ~a~2%" SCENARIO-DATA-MATCH-LIST)
(MAKE-LIST-OF-NUM-PARAMS-EXPECTED)
(FORMAT T "Parameters-per-scenario-expect ~a~2%"
        PARAMETERS-PER-SCENARIO-EXPECT)
(SCENARIO-QUAL-MATCH)
(FORMAT T "Scenario-ratio-match-list ~a~2%" SCENARIO-RATIO-MATCH-LIST)
(SPLIT-INTO-MAJ-MINOR)
  (FORMAT T "Scenario-major ~a~%Scenario-minor ~a~%Scenario-improbable ~a~2%"
        SCENARIO-MAJOR SCENARIO-MINOR SCENARIO-IMPROBABLE)
(MAKE-PARAMETER-COMPARISON)
(FORMAT T "Parameter-ratio-match ~a~%Parameter-rank-list ~a~2%"
        PARAMETER-RATIO-MATCH PARAMETER-RANK-LIST)
(REFINE-PARAMETER-RANK-TOP)
(FORMAT T "Parameter-rank-list ~A~2%" PARAMETER-RANK-LIST)
(ORDER-MULTIPLES)
(FORMAT T "Parameter-rank-list ~A~2%" PARAMETER-RANK-LIST)
(PUT-SCENARIOS-TOGETHER)
(FORMAT T "Possible-scenarios-for-situation ~a~2%"
        POSSIBLE-SCENARIO-FOR-SITUATION)
(OUTPUT-CONTROL)
)                         ;end of DO
    )
)  ;end DECA


;;;===========================================================
;;; Function for manual run
;;;===========================================================

(DEFUN MANUAL ()
  (RESET-AT-END-OF-LOOP)
  (BRING-IN-SYSTEM-DATA "h:>srn>thesis>sensor.data")
  (MAKE-SETPOINT-DATABASE "h:>srn>thesis>setpoint.data")
  (MAKE-PARAMETER-SCENARIO-DATABASE "h:>srn>thesis>scenario.data")
  (MAKE-SCENARIO-EXPECTANCY "h:>srn>thesis>scenario-tendency.data")
  (MAKE-PARAMETER-EXPECTANCY "h:>srn>thesis>parameter-expect.data")
  (MAKE-SCENARIO-DESCRIPTION "h:>srn>thesis>scenario-descriptions.data")
  ;; initialize some global parameters before the loops
 (SETQ SYS-TIME  (CAAR  SENSOR-DATA)
 SENSOR-RECORD (CADAR SENSOR-DATA)
 SENSOR-DATA   (CDR SENSOR-DATA))
 (RESET-AND-REGENERATE-GLOBALS)
 (TIME (LET ()
        (COMPARE-SENSOR-DATA SENSOR-RECORD SDB-LIST)     ;sets oob
        (GET-SCENARIOS OOB-PARAMETERS)
        (MATCH-SCENARIO-TENDENCY)
        (MAKE-LIST-OF-NUM-PARAMS-EXPECTED)
        (SCENARIO-QUAL-MATCH)
        (SPLIT-INTO-MAJ-MINOR)
        (MAKE-PARAMETER-COMPARISON)
        (REFINE-PARAMETER-RANK-TOP)
        (ORDER-MULTIPLES)
        (PUT-SCENARIOS-TOGETHER)
 ))                                    ;end time
 (OUTPUT-CONTROL)

   ;; Reset the parameters for next time step.

 (RESET-AT-END-OF-LOOP)                ;since manual loop
 )  ;end manual
```

```
;;;============================================================
;;; Functions for ouputting the runs to disk files
;;;============================================================

(DEFUN DECA-FILE-OUTPUT ()
 (SETQ OUTPUT-FILE-NAME "h:>srn>thesis>runtime.output")
 (RESET-AT-END-OF-LOOP)
 ;; setup data from file
 (BRING-IN-SYSTEM-DATA "h:>srn>thesis>sensor.data")
 (MAKE-SETPOINT-DATABASE "h:>srn>thesis>setpoint.data")
 (MAKE-PARAMETER-SCENARIO-DATABASE "h:>srn>thesis>scenario.data")
 (MAKE-SCENARIO-EXPECTANCY "h:>srn>thesis>scenario-tendency.data")
 (MAKE-PARAMETER-EXPECTANCY "h:>srn>thesis>parameter-expect.data")
 (MAKE-SCENARIO-DESCRIPTION "h:>srn>thesis>scenario-descriptions.data")

 (SETQ OUTPUT-FILE-NAME (FS:PARSE-PATHNAME OUTPUT-FILE-NAME))

 ;; now begin to loop for each time step that deca is evaluating
 ;;
 (IF (NOT (EQUAL SENSOR-DATA NIL))       ;check if out of data
                                         ;take return out?
    (WITH-OPEN-FILE (FILE-SPEC OUTPUT-FILE-NAME
                          :DIRECTION :OUTPUT
                          :CHARACTERS T)
 ;; now make a run through DECA for the time record sensor-record
(DOLIST  (I SENSOR-DATA)
  (SETQ SYS-TIME (CAR I))
  (SETQ SENSOR-RECORD (CADR I))
  (FORMAT file-spec
          "~%Intermediate parameters for system time: ~a ~%" SYS-TIME)
  (FORMAT file-spec "~%Sensor-record  ~a~%" SENSOR-RECORD)

  (RESET-AND-REGENERATE-GLOBALS)

  (COMPARE-SENSOR-DATA SENSOR-RECORD SDB-LIST)  ;sets oob
  (FORMAT file-spec
      "Oob-parameters ~a~%Oob-parameters-values ~a~%Oob-severity ~A~2%"
       OOB-PARAMETERS OOB-PARAMETERS-VALUES OOB-SEVERITY)
  (GET-SCENARIOS OOB-PARAMETERS)
  (FORMAT file-spec "Lookahead-scenarios  ~a~2%" LOOKAHEAD-SCENARIOS)
  (MATCH-SCENARIO-TENDENCY)
  (FORMAT file-spec
          "Scenario-data-match-list ~a~2%" SCENARIO-DATA-MATCH-LIST)
  (MAKE-LIST-OF-NUM-PARAMS-EXPECTED)
  (FORMAT file-spec "Parameters-per-scenario-expect ~a~2%"
          PARAMETERS-PER-SCENARIO-EXPECT)
  (SCENARIO-QUAL-MATCH)
  (FORMAT file-spec
          "Scenario-ratio-match-list ~a~2%" SCENARIO-RATIO-MATCH-LIST)
  (SPLIT-INTO-MAJ-MINOR)
  (FORMAT file-spec
      "Scenario-major ~a~%Scenario-minor ~a~%Scenario-improbable ~a~2%"
        SCENARIO-MAJOR SCENARIO-MINOR SCENARIO-IMPROBABLE)
  (MAKE-PARAMETER-COMPARISON)
  (FORMAT file-spec
          "Parameter-ratio-match ~a~%Parameter-rank-list ~a~2%"
          PARAMETER-RATIO-MATCH PARAMETER-RANK-LIST)
  (REFINE-PARAMETER-RANK-TOP)
  (FORMAT file-spec "Parameter-rank-list ~A~2%" PARAMETER-RANK-LIST)
  (ORDER-MULTIPLES)
  (FORMAT file-spec "Parameter-rank-list ~A~2%" PARAMETER-RANK-LIST)
  (PUT-SCENARIOS-TOGETHER)
  (FORMAT file-spec "Possible-scenarios-for-situation ~a~2%"
          POSSIBLE-SCENARIO-FOR-SITUATION)
  (OUTPUT-CONTROL-TO-FILE)

  )                          ;end of DOLIST
)                            ;end of with-open-file
    )                                 ;end if
 )  ;end DECA
```

```
;; This function will control all the output to the disk file
;;
(DEFUN OUTPUT-CONTROL-TO-FILE ()
  (COND ((NULL SCENARIO-MAJOR)
   (FORMAT file-spec
           "~%DECA's conclusions for system time: ~a ~%" SYS-TIME)
   (OUTPUT-RESULTS-WITHOUT-SCENARIO-TO-FILE)
   (FORMAT file-spec
           "~%End of data evaluation for system time: ~a ~%" SYS-TIME)
   )
  (T
   (FORMAT file-spec
           "~%DECA's conclusions for system time: ~a ~%" SYS-TIME)
   (OUTPUT-RESULTS-WITH-SCENARIO-TO-FILE)
   (FORMAT file-spec
           "~%End of data evaluation for system time: ~a ~%" SYS-TIME)
   ))
  )

(DEFUN OUTPUT-RESULTS-WITHOUT-SCENARIO-TO-FILE ()
  ;first the scenario
    ;output
    (FORMAT file-spec "~% No scenario selected, not confident enough. ~%")
    (FORMAT file-spec "~% The parameter and priorities are as follows: ~2%")
    (DOLIST (I PARAMETER-RANK-LIST)
      (LET ((PARAM (CAR I))
      (RANK  (CADR I))
      )
 (FORMAT file-spec "~3T~a ~16T~a ~%" PARAM RANK)
 ))
    ;;(WRITE PARAMETER-RANK-LIST :PRETTY :ALIST)
    (FORMAT file-spec
      "~2% Scenarios that were considered as possible choices but not~%")
    (FORMAT file-spec " selected are: ~2%")
    (FORMAT file-spec " Scenario   Ratio    Description  ~2%")
    (DOLIST (SCENARIO-NUM POSSIBLE-SCENARIO-FOR-SITUATION)
      (LET ((TEMP-SCEN-NUM (CAR SCENARIO-NUM))
      (TEMP-RATIO    (CADR SCENARIO-NUM))
      (TEMP-DESCRIPTION )
      )
 (DOLIST (J SCENARIO-DESCRIPTION)
   (IF (EQUAL (CAR J) TEMP-SCEN-NUM)
       (SETQ TEMP-DESCRIPTION (CADR J)))
   )
 (FORMAT file-spec "~3T ~a ~13T~A ~20T ~a~%"
         TEMP-SCEN-NUM
         TEMP-RATIO
         TEMP-DESCRIPTION)
      ))
    (FORMAT file-spec "~2%")
    ;;(WRITE POSSIBLE-SCENARIO-FOR-SITUATION :PRETTY :ALIST)
  )

(DEFUN OUTPUT-RESULTS-WITH-SCENARIO-TO-FILE ()
  ;first the scenario
  (LET ((TEMP-SCENARIO-SPECIFICS )
  )
    ;output
  (FORMAT file-spec "~% Scenario selected is; ~%")
  (SETQ TEMP-SCENARIO-SPECIFICS (GET-GUESS))
  (FORMAT file-spec " Scenario Number ~a ~%" (FIRST TEMP-SCENARIO-SPECIFICS))
 (FORMAT file-spec " Scenario Description ~d ~%" (LAST TEMP-SCENARIO-SPECIFICS))
  (FORMAT file-spec " Confidence  ~a ~2%" (SECOND TEMP-SCENARIO-SPECIFICS))
  (FORMAT file-spec "~% The parameter and priorities are as follows: ~2%")
    (DOLIST (I PARAMETER-RANK-LIST)
      (LET ((PARAM (CAR I))
      (RANK  (CADR I))
      )
 (FORMAT file-spec "~3T~a ~16T~a ~%" PARAM RANK)
 ))
    ;;(WRITE PARAMETER-RANK-LIST :PRETTY :ALIST)
    (FORMAT file-spec
      "~2% Scenarios that were considered as possible choices but not~%")
    (FORMAT file-spec " selected are: ~2%")
    (FORMAT file-spec " Scenario   Ratio    Description  ~2%")
    (DOLIST (SCENARIO-NUM POSSIBLE-SCENARIO-FOR-SITUATION)
      (LET ((TEMP-SCEN-NUM (CAR SCENARIO-NUM))
      (TEMP-RATIO    (CADR SCENARIO-NUM))
      (TEMP-DESCRIPTION )
      )
  (DOLIST (J SCENARIO-DESCRIPTION)
    (IF (EQUAL (CAR J) TEMP-SCEN-NUM)
        (SETQ TEMP-DESCRIPTION (CADR J)))
    )
  (FORMAT file-spec "~3T ~a ~13T~A ~20T ~a~%"
          TEMP-SCEN-NUM
          TEMP-RATIO
          TEMP-DESCRIPTION)
      ))
    (FORMAT file-spec "~2%" )
    ;(WRITE (CDR POSSIBLE-SCENARIO-FOR-SITUATION) :PRETTY :ALIST)
    )
  )

;;;================================================================
;;; End of DECA Kernel
;;;================================================================
```