NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

ISBN 0-315-53295-5

Canada

A HIERARCHICAL EXPERT SYSTEM FOR COMPUTER PROCESS CONTROL

by

IOAN TRIF

AUGUST 1988

A thesis
submitted to the Graduate Studies and Research
in partial fulfillment of the
requirements for the degree of
Master of Applied Sciences
in Electrical Engineering

OTTAWA-CARLETON INSTITUTE FOR ELECTRICAL ENGINEERING

Department of Electrical Engineering
Faculty of Engineering

University of Ottawa, Ontario, CANADA

CONTENTS

	Page
LIST OF FIGURES	vii
LIST OF TERMS	viii
ACKNOWLEDGEMENTS	x
ABSTRACT	xi
1. INTRODUCTION	1
1.1 Thesis Research Areas	1
1.2 Artificial Intelligence and Process Control	2
1.3 Process Control Domain Characteristics	8
2. EXPERT SYSTEMS FOR PROCESS CONTROL	9
2.1 Expert Systems for Process Operator	11
2.1.1 Duties of Process Operator	11
2.1.2 Demands of Expert Systems for Process Operator	13
2.1.3 Unique Problems in Real-Time Expert Systems	13
2.1.4 Types of Process Models	15
2.1.5 How Models Can Be Used With Expert Systems	17
2.2 Expert Systems for Control System Design	20
2.2.1 Introduction and Motivation	21
2.2.2 The Interactive Design Process	22
2.2.3 Traits of A Human Expert	24
2.2.4 Traits of An Expert in Control System Design	27
2.2.5 Comments	27
2.3 Expert Systems for Process Management	28
2.3.1 Manager's Role in a Plant	28
2.3.2 How Expert Systems Could Help Process Managers	29
2.4 Some Examples of Expert Systems Used in Process Control	30
2.4.1 FALCON	30

2.4.2 Heat Exchanger 31
2.4.3 EXACT 31
2.4.4 Flow Meter Consultant 31
2.4.5 DELTA 32
2.4.6 ACE 32
2.4.7 MAID 33
2.4.8 CACE-III 35

3. DESIGN AND IMPLEMENTATION OF THE HESCPC SYSTEM 36

3.1 Tools Used for the HESCPC Implementation 36

3.1.1 A brief Overview of Reshell 38
3.1.2 Rules vs. Frames 42
3.1.3 Comments 45

3.2 Architecture of the HESCPC Expert System 46

3.2.1 Declarative Knowledge Block (DKB) 49
3.2.2 Declarative-Procedural Knowledge Interface Block (DPKIB) 55

3.2.2.1 How the DPKIB Block Works 58

3.2.3 Procedural Knowledge Block (PKB) 59

3.3 Implementation Considerations of the HESCPC Expert System 59

3.3.1 The <expert_name>. RBS File 62
3.3.2 The <expert_name>_ CA.PRO File 66
3.3.3 State Space Feedback Design Heuristics 70
3.3.4 Why an Expert System ? 72
3.3.5 Comments 74

4. COMMENTS ON THE SAMPLES RUN OF THE HESCPC EXPERT SYSTEM

4.1 Control System Design with HESCPC 76

4.1.1 Multivariable Linear System Design Examples 76
4.1.2 Monovariable Linear System Design Example 80

4.2 Operator-Advisor Samples Run 81

5. CONCLUSION 83

5.1 What Has Been Achieved 83
5.2 What Has Not Been Achieved 84
5.3 Future Work 85

6. RELATED REFERENCES 86

7. APPENDIX 89

7.1 Test 1 89
7.2 Test 2 107
7.3 Test 3 111
7.4 Test 4 120
7.5 Test 5 124

LIST OF FIGURES

		Page
Figure 1.1	General Architecture of a Process Controlled by Conventional and Expert System Automation Tools	6
Figure 1.2	General Architecture of an Expert System for Process Control	7
Figure 2.1	A Process Control System	10
Figure 2.2	Interactive Design Process	23
Figure 2.3	Design as a Feedback Process	25
Figure 2.4	MAID Expert System	34
Figure 2.5	HESCPC Expert System	34
Figure 3.1	The HESCPC expert as a Three-Block System	37
Figure 3.2	General Block Diagram of an Expert System	39
Figure 3.3	Architecture of Reshell	41
Figure 3.4	Hierarchical Organization of a Reshell-based Expert System	43
Figure 3.5	Architecture of the HESCPC Expert System	47
Figure 3.6	Declarative Knowledge Block (DKB)	50
Figure 3.7	An Operator-Adviser Expert System for a Nuclear Power Plant	52
Figure 3.8	Declarative-Procedural Knowledge Interface Block (DPKIB)	56
Figure 3.9	How the DPKIB Block Works	60
Figure 3.10	Fortran Routines used to Solve Subproblem 5	61
Figure 3.11	State Space Feedback Design Heuristics	71
Figure 3.12	General Flowchart of the HESCPC Expert System	75

LIST OF TERMS

AI	Artificial Intelligence
CA	Condition and Action file
CAD	Computer Aided Design
CCRS	Canada Center for Remote Sensing
CLADP	Cambridge Linear Analysis Design Program
CONTPACK	Control Package
DKB	Declarative Knowledge Block
DPKIB	Declarative-Procedural Knowledge Interface Block
ES	Expert System
HESCPC	Hierarchical Expert System for Computer Process Control
LDIAS	LANDSAT-D Image Analysis System
LTI	LDIAS Task Interface
OKB	Operator Knowledge Base
PC	Process Control
PKB	Procedural Knowledge Block
RBS	Rule Base System file
SSFE	State Space Feedback Expert

To my parents ...

ACKNOWLEDGMENTS

First, I would like to thank Dr. Dan Ionescu, my thesis supervisor, for his direction, guidance and advice. His time and patience commited to communicate his knowledge, judgement, and experience in Computer Process Control has been rewarded with the results of the research presented in this thesis. I would also like to thank him for introducing me to the new field of Artificial Intelligence.

Next, I would like to thank Dr. Morris Goldberg, for making Reshell available to the purposes of this research.

Next, I would like to thank Dr. Gerald Karam, one of the originators of Reshell, for clarifying to me some problems encountered with the integration of declarative-procedural knowledge.

Next, I would like to thank Mr. Mohammed Master for providing good working conditions of the computing facilities during this research. Also, I would like to thank Mr. Jim Burgess, consultant-analyst in the Electrical Engineering Department, for helping me to port some routines written in Assiris to VAX/VMS.

Finally, I would like to thank my wife Elena and my daughter Silvana for their patience and understanding throughout the course of this thesis elaboration.

ABSTRACT

This thesis is concerned with the design and implementation of a Hierarchical Expert System for Computer Process Control (HESCPC). The hierarchical architecture integrates four classes of expert systems: operator/manager_companion expert class, control_system_design algorithms expert class, hardware_design expert class, and software_design expert class. The thesis emphasizes the integration of declarative knowledge represented by M-Prolog rules with procedural knowledge embedded in a specialized Fortran library.

At this stage of the HESCPC development, the declarative knowledge represented by 123 meta rules and 261 rules is distributed on a hierarchical structure among 20 experts on different levels of the hierarchy which are able to communicate among themselves to solve difficult control problems. The design and implementation of the general expert system structure, an operator-adviser expert, and a control system design expert has been accomplished.

Examples of control system design sessions of *linear mono and multivariable systems* using *feedback state space approach* are given. A sample run of an *operator-adviser* data driven expert for a nuclear power plant is also presented.

PART 1

INTRODUCTION

This thesis considers the design and implementation process of an intelligent piece of software which can aid either the control design engineer or the human operator to solve complex and difficult control problems. The goal of *Part 1* is to present the thesis research areas and its organization, the role of Artificial Intelligence (AI) in Process Control (PC), and some process control domain characteristics.

1.1 THESIS RESEARCH AREAS

The results reported in this thesis constitute a beginning effort at the University of Ottawa to build an intelligent tool - *an expert system* - for Process Control. Theoretical studies regarding the possibility of constructing such a system started in the summer of 1986 and the implementation process started in the summer of 1987.

This thesis covers the following major topics:

- Definition, design and implementation of a hierarchical structure which includes all the areas of application of expert system technology in the field of Process Control,
- Design and implementation of a declarative-procedural knowledge interface that makes possible the integration of control system design rules, represented

by M-Prolog statements, with control system design algorithms embedded in a specialized Fortran library, and

- Design and implementation of a small operator-adviser data driven expert for a nuclear power plant operator.

The thesis is organized as follows. Following this introductory part, *Part 2* presents the various areas where expert systems can be found in the field of Process Control. Some examples of expert systems used in Process Control are also given. *Part 3* discusses the architecture of the HESCPC expert system and some of the important issues related to its implementation. *Part 4* outlines the kind of problems the HESCPC expert is able to solve and analyses the experimental results obtained. *Part 5* provides some lessons learned from this experiment and proposes some future work that should be done for further development of the HESCPC expert system. *Part 6* gives Related References and finally, an *Appendix* presents the HESCPC system at work. This includes, design sessions of linear mono and multivariable systems using the *feedback state space approach* as well as a sample run of an operator-adviser expert for a nuclear power plant.

1.2 ARTIFICIAL INTELLIGENCE AND PROCESS CONTROL

Interest in Artificial Intelligence (AI) has been growing dramatically in the last few years, and many corporations have set up AI groups or are in the process of doing so. One of the prime areas of corporate interest is Expert Systems (ES). Though the number of expert systems actually functioning in a corporate environment is still relatively small [39], the number of projects looking into expert system development is growing rapidly.

Whether expert systems are viewed as a fundamental principle for organizing and animating expert knowledge, a challenge to create much smarter computer software, a concept for organizing otherwise ill-structured systems, or a wildly fraudulent fad, their goal of capturing commonsense reasoning in software correlates well with the structure of process control knowledge [15] . Long ago, the process control field realized the limited value of formal analytical methods in favor of a combination of recognized, simplified, analytical techniques, designer judgements, and tuning procedures.

Intelligent software should improve all aspects of process automation design and function. However, the implementation of this software faces serious difficulties because of the large spectrum of knowledge to be acquired, the size of the rule base to be created, and the complexity of the human and system interfaces. The potential benefit is to allow rapid application of defined expertise to dynamic problems involving thousands of variables. The system approach to this, requires the consideration of the challenges of real-time inference, in particular, the dynamic nature of the domain, the large knowledge base and the requirements for efficient execution.

The present digital control systems do not provide means for intelligent interpretation of sensor data, diagnosis of problems, coping with process disturbancies and predicting consequence of actions. In a large processing plant, such as a steelmill, a nuclear power plant or an oil refinery, there are several thousand measurements and alarms provided to the human operator. In a major process upset, the operator may be confronted with a very large number of individual alarms, but with very little intelligence concerning the underlying plant condition. The plant status may change significantly within minutes. The large size and dynamic nature of the domain requires new approaches to the inference, since exhaustive search procedures are not possible in real-time. The knowledge based system offers a natural vechicle to perform inference

as would a human expert who is confronted with the same problem of limited time to respond to a complex situation. The key concepts are to quickly recognize process conditions which are potentially significant and to invoke relevant sets of rules and focus on these problem areas for diagnosis and procedural advice [10]. On the other hand, expert systems could not replace the standard control algorithms, the use of advanced control techniques, optimization, etc. but will augment the control power of the above.

The important requirement for integrating an expert system with a distributed control system is an effective communication between the two. The process data base must be accessible to the expert system in as much detail as it is accessible to the human expert (i.e. the expert system should be able to access current data as well as historical data files). To prevent overloading the communication network, the expert system should be able to vary the scan rates on the severity of abnormal conditions in a selected process area. The current process data is used to update the knowledge base.

Typically, the vast and complex process operating knowledge [10] is shared between the process control engineer and process operator. Control engineers tend to analyse process performance, whereas operators use heuristic rules to operate the plant. The engineer should be provided with user friendly interactive tools to enter his process knowledge and operator heuristics into the knowledge base. The expert system should develop inference results based on various types of evaluations, and should also provide capabilities to invoke logic rules and procedures to diagnose a process problem and explain to the operator the recommended course of action.

The state variable representation is a basic construct for dealing with the time varying quality of the environment. For this reason an expert sytem used in process control must be able to gather data, construct the generalized state variable of the process, arrive at conclusions, decide upon actions, and carry them out in an

environment of continuously changing circumstances. It must be capable of changing its plans when circumstances change, and it must be able to operarte as best it can with incomplete data. Furthermore, it must be able to operate in a situation in which not all information is up-to-date [10]. This dynamic aspect of process control imposes the expert sytem to also include a wide procedural knowledge-base with already known or learned control cases. Process models, which will be discussed in Section 2.1.4 can be an important resource for an expert system to use during execution.

The implementation of an expert system satisfying the above mentioned requirements, as proposed by K. Gidwani [10], should have the general architecture as presented in figure 1.1.

The main idea following in this thesis is to design and implement a complex expert system that groups together the hierarchical structures of four expert classes as depicted in figure 1.2. Each expert class of this structure will satisfy some of the objectives mentioned above, combining at all stages of the hierarchically distributed architecture, procedural and declarative knowledge, therefore different software technologies. This mixing of numeric-processing languages, and languages that are fundamentally symbolic, needs enhanced computing resources. To satisfy the real-time objective of both the computational complexity and symbolical programs, a multiprocessor organization with high speed CPU's at competitive prices is required.

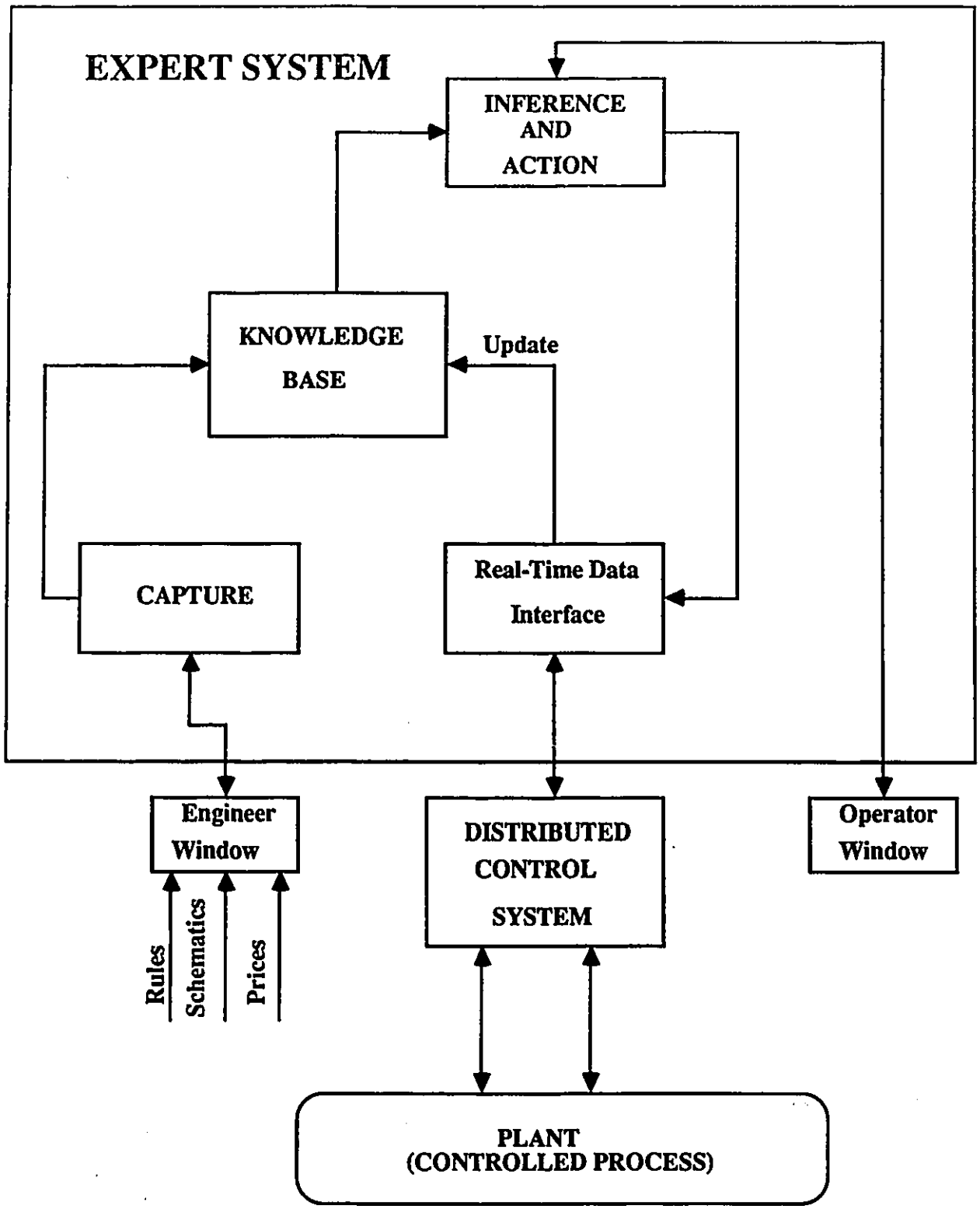


Figure 1.1 : General Architecture of a Process Controlled by Conventional and Expert System Automation Tools [from K. Kumar Gidwani 1985]

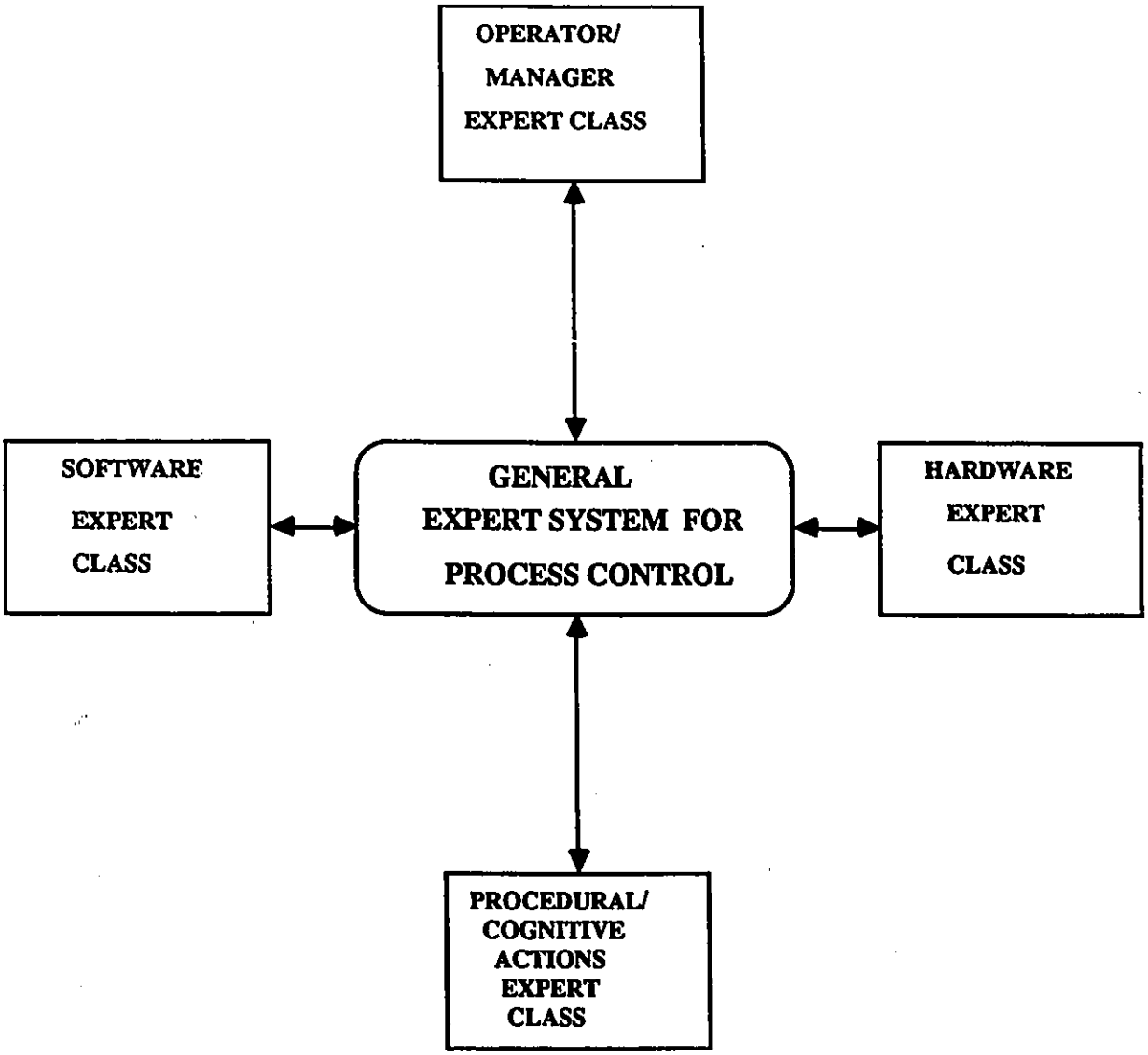


Figure 1.2 : General Architecture of an Expert System for Process Control

1.3 PROCESS CONTROL DOMAIN CHARACTERISTICS

This short section intends to point out some of the most important process control domain characteristics which make the implementation of expert systems for this type of applications a difficult job. Many of these characteristics have been already discussed in Section 1.2 and no further comments are needed at this time.

A. A process control domain description requires:

- o a large spectrum of knowledge,
- o different types of knowledge: declarative and procedural which must be integrated to work as a whole,
- o a large knowledge base.

B. An expert system for process control must be able to deal with:

- o real-time constraints,
- o limited time to respond to very complex situations,
- o a domain which is dynamic in nature,
- o many disturbances that take place in a short period of time,
- o complex human-machine interfaces.

PART 2

EXPERT SYSTEMS FOR PROCESS CONTROL

Process control represents one of the most complex and challenging control software activities. Applications of expert systems in the field of process control can be found in the following areas:

- o Expert Systems for Process Operator
- o Expert Systems for Control Systems Design
- o Expert Systems for Process Management

As shown in figure 2.1, (from Freeman 1985 [9]) the heart of a modern process control system is a process computer which receives information from both the process model and the data acquisition system, interprets the data and instructs the process controller how to control the process. The process computer also communicates with display systems, data storage and retrieval systems, and off-line analysis functions. The proper control of such a complex system can require considerable expertise.

Expert system technology brings some significant help to the plant operator, control system designers, as well as process management staff. The idea is to make available the expertise of master operators, control system designers or process managers as a tool for the average person in the specific area.

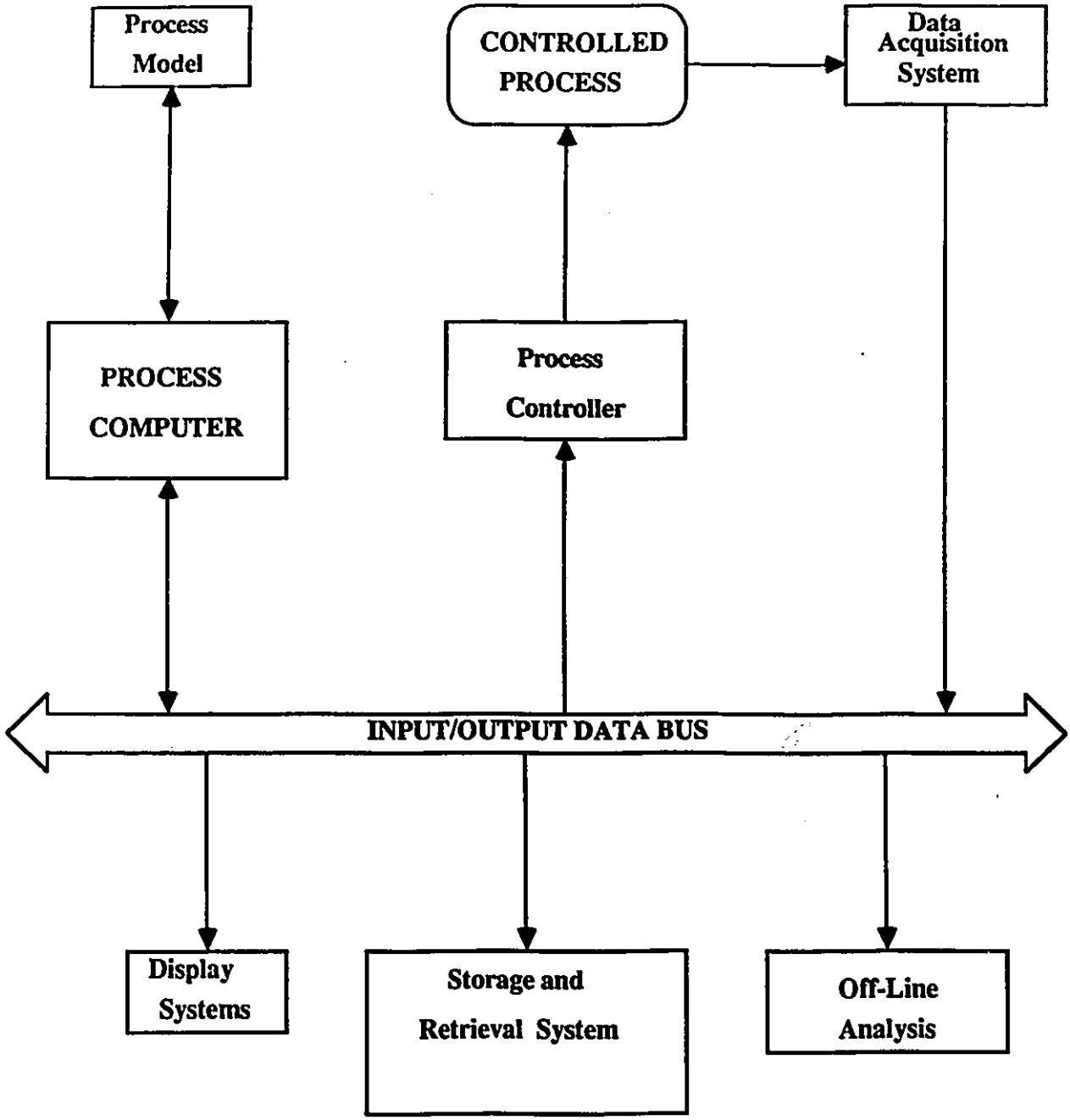


Figure 2.1 : A Process Control System [from Freeman 1985]

2.1 EXPERT SYSTEMS FOR PROCESS OPERATOR

In a control room there may be thousands of pieces of information available to the operator at all times. There are alarms, gauges, recorders, indicators, lights, horns, computer displays, etc. One of the problems is how to transform this vast amount of data currently available into more valuable and useful information [3].

An expert system can facilitate the rapid acquisition of relevant data and on-line analysis supporting the decision making. Various types of gauges may be displayed and assigned variables to monitor. Cross plotting of variables may be specified and displayed in multiple windows. Multiple activities may be simultaneously monitored while control action is taken. Models may be run and predictions compared with data to test control actions. Rules may be invoked to temporarily modify the control while a diagram is completed. In this way the operator can be helped toward a conclusion that is consistent with all of the available knowledge and information about the process and its current state [9].

2.1.1 Duties of Process Operators

As described in [21], [7], [14] the process operator's duties fall into four categories:

A. Process start up or change over,

B. Process regulation and optimization,

C. Dynamic avoidance of and compensation for breakdowns, and

D. Diagnosis of failures.

A. A human operator performing the *start up* or *change over* of a process can be helped by an expert system functioning as an adviser or coach to plan and time his/her

control actions. Thus the process is guided through various state transitions toward the desired steady-state operation. The expert system must also be able to re-plan the route to the desired state in the face of unforseen events. The benefit sought from an expert system of this type is the increased ability to quickly achieve the desired operating state, minimizing scrap but not compromising safety [21].

B. Substantial prescriptive as well as diagnostic expertise can be provided to the human operator by an expert system in *regulation and optimization* duties once normal operation of the process is attained. This type of expert system must detect when the process moves away from the desired state and recommend proper actions. In this way the operator is guided in moving the process towards a more stable state or states producing the desired output.

C. An expert system which is able to predict future states of the process and recommend actions based on assessment of the current state will help operators in *dynamically avoiding* or *compensating* for breakdowns. Also, in compensating for unavoidable failures, it would recommend corrective actions based on an assessment of the alternatives presently available, rather than on pre-scripted or "canned" advice, which might not be applicable in a crisis situation [35].

D. Finally, the last category of expert systems meant to aid the operator in *diagnosis of failures* is not difficult to imagine. Such an expert system making reasoning on data from a set of sensors will be presented in Section 3.2.1 as a component of the HESCPC system.

2.1.2 *Demands of Expert Systems for Process Operator*

From the above discussion one can conclude that expert systems for process operator applications *must* meet the requirements of real-time process control software. With the exception of diagnosis of upsets that have led to a plant shutdown, expert systems for process operators will necessarily be a real-time piece of code. Therefore the system must be capable of scanning time varying and possibly rapidly changing data, handling interrupts and tasks of diverse priorities, and initiating actions depending on events or on a schedule [27]. For coping with such complex problems, Sauers and Walsh [41] have proposed the approach of changing data by means of shared memory, while Moore and others [34] have implemented techniques for interfacing the inference engine of an expert system to a data scanning module which runs its own code.

2.1.3 *Unique Problems in Real-time Expert Systems*

As was stated above expert systems for process operator are in fact real-time expert systems. Two unique problems arise in such a system that do not necessarily occur in other real-time software.

A. How *to re-use the system's memory space,* and

B. How *to provide real-time advice* concerning a system of complex causal relationships using information from variables that are generally functions of time.

A. In order for a perpetually running expert system to never run out of memory, obsolete facts must eventually be removed and space must be made available for new data through a "garbage collection" mechanism [21]. This solution can lead to spurious

application of rules in the expert system's knowledge base if there are rules whose applicability depends not only on the presence of some facts, but also on the absence of others. Griesmer and others [12] have proposed a solution to this problem by giving to the garbage collection task a higher priority than other tasks, but this may "lock" other tasks out at inopportune times. Other solutions, such as preventing the marking of facts as "obsolete" by one rule when their absence is meaningful to other rules, require further research to determine their practicability [21].

B. An expert system for process operator must be able to perform a variety of dynamic, "temporal reasoning". It is not enough for this type of intelligent piece of software to reason about data that is transient and changing, it must also be able to reason about transient conditions of the process as a whole, about the time required for initiating an action, and the time needed before the effects of an action will occur. An approach to make possible this type of reasoning is to encode temporal information as part of specific rules. Moore and others [34] have proposed that the elapsed time before the effects on an action are expected to be detected and coded in a triplet structure in the expert system's knowledge base as:

[<action>, <time-delay>, <expected-result>].

As stated in [21], this approach has the advange of making the temporal information explicit, and keeping the information lexically located with the rules that use it. However, it has the disadvantage of requiring the information to be determined ahead of the real-time situation, raising the possibility that the approach will lead to numerous ad hoc rules that eventually become unmanageable. As an alternative to the above approach, Kaemmerer and Christopherson [21] have proposed the use of a dynamic model of the process which has to be controlled as a part of the expert system's reasoning process. Their approach will be discussed in Section 2.1.5.

2.1.4 Types of Process Models

The report made by Crossman in 1974 in [7] shows that use of some type of process model is necessary for process control operators to become expert at their job. It seems that expert operators as opposed to average ones apparently do not use a "rule-of-thumb" approach in controlling a process, nor do they use analytical reasoning of general engineering principles; rather they use an *intuitive mental process model.* Common factors seem to underline individual skills in terms of both the speed of learning new control jobs, and the individual's final level of ability in controlling a process. These underlying factors include the ability:

- to predict what is likely to happen in a situation if the process controls are left alone,
- to know what means can be used to influence the process in the current situation, and
- to select the proper control action most likely to achieve the desired result.

From the above discussion one may hypothesize that the operator was helped by an *intuitive mental process model* to achieve expert performance by organizing his/her knowledge about the process (reducing cognitive load), and by facilitating casual and temporal reasoning about it. This, in turn, may support the ability of the operator to forecast the likely future states of the process, to determine what actions are effectively available in the current situation, and to distinguish transient conditions requiring no action from enduring ones needing attention [21]. The question is: *can we expect similar benefits from the use of process models with expert systems?*. The answer cannot be given here. What is less clear is *how* such process models are to be used in conjunction with expert systems. Kaemmerer and Christopherson [21] suggest some

approaches to this problem which will be presented shortly.

From an abstract point of view, all models can be seen as *information resources* that predict the consequences of changes to the state of the system being modelled. During the course of this research the following types of models were found in the literature:

1. Functional model
2. Physical model
3. Economic model
4. Procedural model
5. Cognitive model

The first four types of models are defined by Lee, Addams and Gaines in [28] as follows:

- *A functional model* is a description of the functions performed by the major portions of a plant, and the relationships among them, with respect to the flow of materials through the process;

- *Physical process model* defines the mathematical relationships among the physical variables involved in the process;

- *Economic model* defines the realtionships among the resource costs, the constraints on the mix of materials permissible and the objective function describing the aim of process operation (e.g. to minimize cost);

- *Procedural model* describes how the plant should be operated, especially under start up, shutdown, and load changes;

- *Cognitive model* was introduced by Kaemmerer and Christopherson and described at the begining of this section. A precise definition of this type of model is impossible to be given, but studies of operators have provided information on the nature of cognitive models. For example, Bainbridge and others [1] have studied operators performing proportional control of one process variable in a task in which many parameters had to be taken into account in the process control. They found that experienced operators perform the task by predicting the significant events expected to happen next, and the control action to take, if any, in response. In contrast, inexperienced operators use error-correcting actions and feedback rather than predictive control, and organize their data monitoring efforts less efficiently. The experienced operators' performance suggests that *their cognitive models* include means for *identifying the present* and *predicting the future states* of the process, as well as some knowledge of the dynamics of the process, and a repertoire of control actions that can be carried out on it [21]. Furthermore, related studies indicate that an operator's cognitive model includes a running record of important system variables, which are kept in mind and updated regularly only if the operator is more or less continuously involved in the control activity [2].

2.1.5 How Models Can be Used with Expert Systems

Kaemmerer and Chritopherson [21] suggest the following use of models

with expert systems:

A. Using models in *knowledge engineering,*

B. Using models for *automatic knowledge acquisition,*

C. Using models during *expert system execution,* and

D. Using information from models for *efficient implementation.*

A. It is known that *knowledge engineering* is the process of acquiring knowledge from human experts and encoding it for system use. When a functional model defining the components of the process and their interrelationship is acquired, the knoweldge engineer establishes the entities and variables of interest to the system, and the physical paths by which components of the system can effect each other. A precise functional model of the process obtained from designers is needed to define the measurements by which the expert system will interface to the actual process. The informal description of the functional model of the process drawn from operators is valuable for identifying the salient feature of the process in the operators' minds.

Another use of functional models in knoweldge engineering is to generate *sample problems* to be posed to the expert operators, in building expert systems as diagnostic aids. In this technique, a component of a functional model is selected as a locus of an imaginary process upset, and an expert operator is asked to predict how that upset would be manifested to the plant operator. Because the mental process by which one recognizes a situation from the data is likely to be different from that by which one generates a description and data given a situation, the technique involves the use of two experts. The second expert is presented with the data imagined by the first, and asked to describe the data collection and reasoning steps to be used in diagnosing the upset.

B. To build an expert system is not an easy job even if the *four key resources* - an expert, a knowledge engineer, a computer and a software tool - are available. Currently, many person-years of effort are required to capture the knowledge that underlies an expert's performance from the expert, through person to person interviews, observations, and problem solving exercises. To reduce this labor Kaemmerer and Christopherson proposed [21] to design an expert system capable of augmenting its own knowledge base, by "observing" an expert operator in action. The observations would consist of a record of variables describing the state of the process at selected time intervals, while an expert operator runs the controls. The rules to be derived by the system from these observations would be a triplet:

[<state_k>, <action_k>, <state_k+1>],

encoding the information that "when the process is in state_k, the expert operator takes action_k, bringing the process into the state_k+1". Furthermore, general rules can be obtained by using generalization techniques from individual situation sequences. In order for these techniques to be applicable, they *must* be assigned with *finite state-space* descriptions of the situation. It is also efficient for the system to have the knowledge needed to recognize pertinent analytical relations between corresponding elements of the states [38], [45]. Physical process models can provide some of the latter information.

C. As was stated in Section 1.2, process models can be an important resource for an expert system to use during execution. For example, an expert system performing a prescriptive or advisory function may itself use a physical process model as part of its reasoning process (e.g., to generate the initial recommendations for set points in supervisory control). In a *diagnostic system*, a simulator can be used on a time availability basis to generate predictions from hypothesized causes of the failure in

order to determine how well the hypothesized cause also accounts for other observed data. To the extent that other data are consistent with the predictions of the simulator, the sytem confidence in the hypothesis may be increased, (assuming the simulator itself has previously been validated). In *prescriptive systems* calls to a simulator may be used to verify that the results expected from the recommended actions will be desirable, and to generate expectations against which future incoming information from the process can be compared. The latter provides a means to "ignore" transient effects, by designing the rules in the knowledge base to suppress alarms that would otherwise occur, as long as the incoming data do not deviate significantly from simulated transients previously judged to be acceptable [21].

D. Finally, process models can provide useful information for an *efficient implementation* of an expert system. As was discussed in Section 2.1.4, the expert operator's cognitive model of the process is related to the means he/she uses to organize the scanning of incoming data from the process, making his/her cognitive load more manageable. Thus, the operator's cognitive model should aid in grouping the data points to be scanned by the expert system, and in setting the relative priorities for the scanning tasks under different situations. Functional models can provide a means to organize rules according to the functional components of the process to which they might pertain, leading to faster search for relevant rules [21].

2.2 EXPERT SYSTEMS FOR CONTROL SYSTEM DESIGN

With the advent of increasingly more powerful computers and increasingly more capable software, engineers have been given the opportunity to apply an expanding number of computationally demanding approaches to the solution of design problems.

Indeed, conferences devoted exclusively to the exchange of information on the status of computer-aided control systems design methods and programs are now held regularly in the United States as well as in Great Britain [20].

2.2.1 *Introduction and Motivation*

Research in feedback control proceeds on two levels. The goal of the first level is to invent, analyze, and describe control schemes that allow the construction of new physical processes which perform better. The goal of the second level is to improve the productivity of the control sytem designer who specifies how these new processes will be built. This productivity goal is enhanced with the emergence of new design methodologies and with computer-aided control engineering software that supports the design process. The availability of the first generation computer-aided control engineering software has vastly accelerated the applications of sophisticated feedback control algorithms through their implementation in digital controllers. However, the diversity and complexity of the system classes and consequently the pertinent control algorithms and the conditions under which these control algorithms are valid, makes the building of a general valid software package which is completely reliable and easy to work with practically impossible. It is also obvious that a perfect human expert in all the areas of control system doesn't actually exist. For this reason it has been recommended to capture the existing expertise from different experts in the related fields and to include it in a knowledge base of an expert system.

One advantage of the expert system is its ability to explain the internal workings of the program to the user while the program is executing. Another advantage of the expert system is the consistency with which it responds to situations since the rules do not change from one design session to another. Thus, it is believed that expert system

programming techniques provide a reasonable approach to deal with the complexity of the design process. This belief is based partially on the experience gained from investigating the application of expert systems techniques to design problems and partially on the extrapolation of this experience [20]. At this point it is evident that :

- o design heuristics (rules of thumb) can be placed in a knowledge base and incrementally expanded or amended,
- o the applicability of a design method to a given plant can be automatically analysed and advice provided to the user,
- o the nuances of using a particular approach can be placed in the knowledge base and are not overlooked by the system during execution,
- o parameter variations can be analysed in a straightforward manner,
- o the results of sets of parameter values can be automatically determined and combinations of programming methods (modular and rule-based) can be employed in building an expert system.

2.2.2 *The Interactive Design Process*

The relationship between man and machine in the interactive design process has been discussed by Pang and MacFarlane [36]. A large part of the material in this section has been adapted from their considerations. They consider data passed from machine to man in terms of *indicators* and data passed from man to machine in terms of *drivers* as depicted in figure 2.2. The man works in terms of a high level conceptual framework and accesses in the machine a powerful manipulative framework. The basic task in creating a satisfactory interactive computing system is to get these two

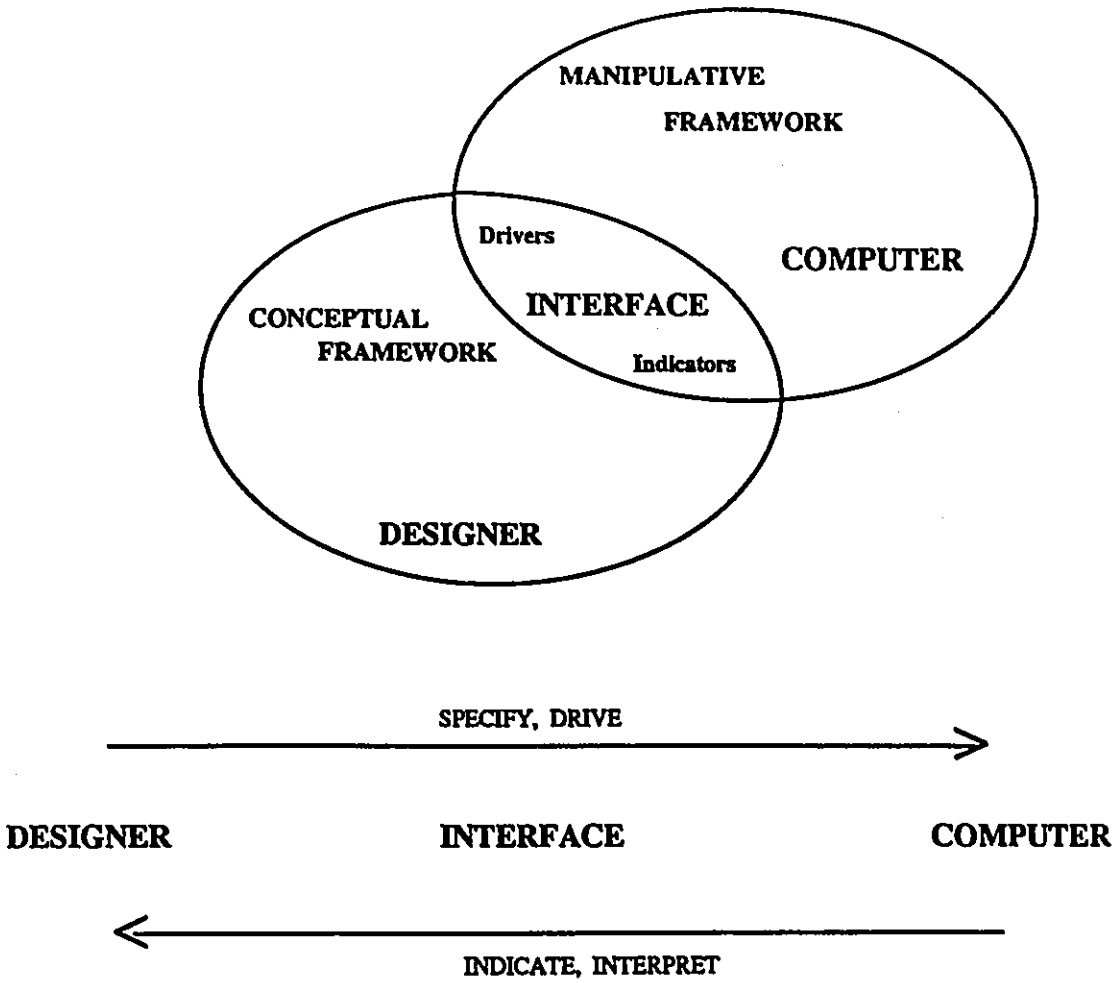


Figure 2.2 : Interactive Design Process [from Pang and MacFarlane 1986]

frameworks to mesh together satisfactorily via an appropriate set of indicators and drivers. Thus, the design process can be described as:

- *A feedback process,* where both the object being created and the specification against which it is being manipulated are being iteratively adjusted in a feedback cycle of dependence as the design proceeds. This is illustrated in figure 2.3.
- *A process of instantiation:* the progressive generation of a specific fully defined object from an initial incomplete general description. In creating a specific instance of the general class of object desired, the designer is grappling with both uncertainty and complexity.

When developing an interactive computing environment, we have to take proper account of the man as well as the machine. In discussing this it is useful to talk in terms of ***principles*** and ***procedures***. ***Principles*** are the organizers of high-level declarative knowledge, and ***procedures*** are the implementors of low-level imperative or procedural knowledge. In an interactive computing environment both the declarative knowledge and procedural knowledge have to be integrated in an effective and efficient way. Such an integration will be presented in *Part 3* of this thesis.

2.3.3 Traits of a Human Expert

The traits of a human expert have been considered by Michaelson, Michie and Boulanger [33]. They asset that human experts can be characterized by the following traits:

- Collect information,

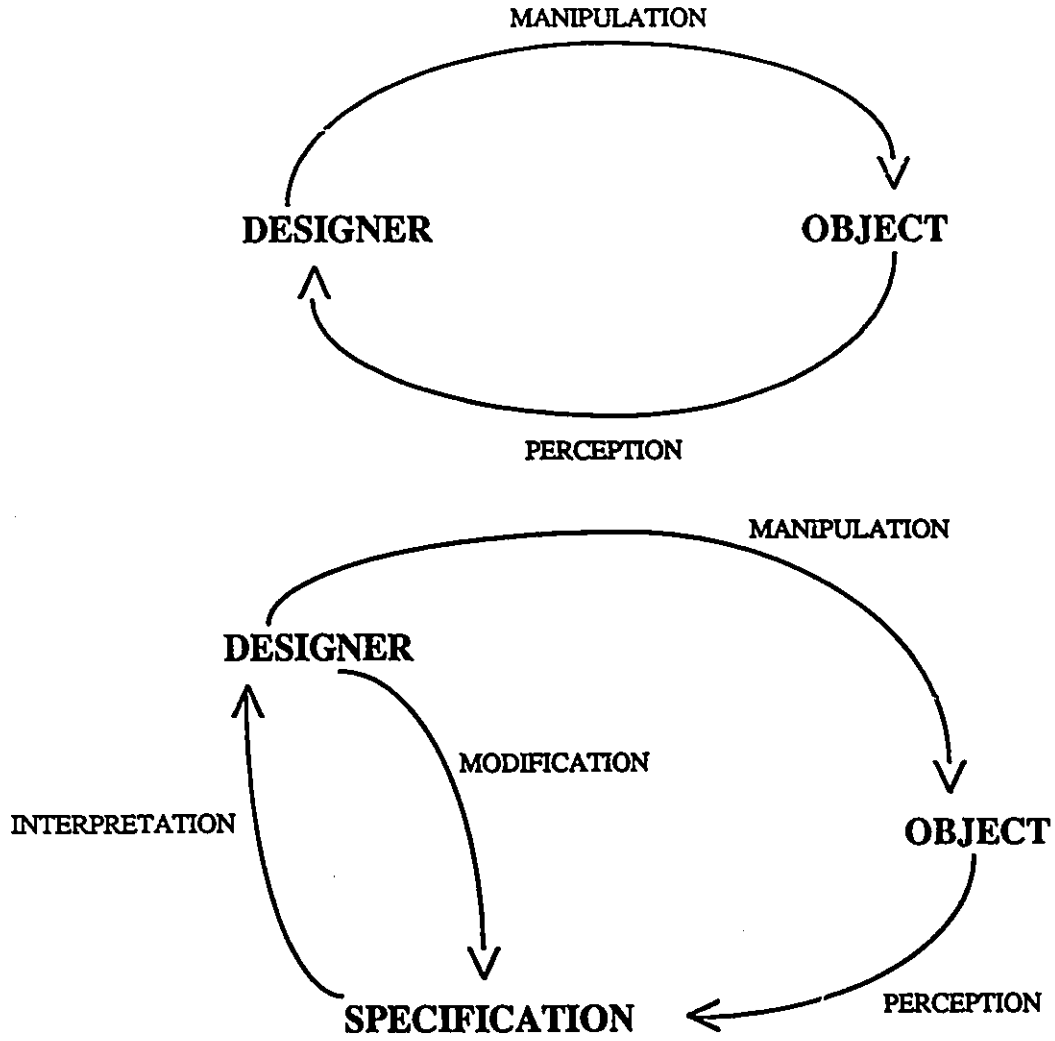


Figure 2.3 : Design as a Feedback Process [from Pang and MacFarlane 1986]

- Identify problem,
- Group and differentiate,
- Solve problems that are unstructured and ill-defined,
- Use heuristics (rule of thumb) to solve problems,
- Engage in different problem solving activities,
- Process data,
- Generate questions,
- Make decisions,
- Explore and refine,
- Pursue and test hypothesis,
- Establish hypothesis space.

In addition to these traits, they observed the following capabilities exhibited by human experts:

- Apply their expertise to the solution of problems in an efficient manner,
- Restructure and reorganize the knowledge,
- Employ plausible inference,
- Reason from incomplete or uncertain data,
- Explain and justify what they do,
- Communicate well with other experts and acquire new knowledge,
- Break rules,
- Determine relevance,
- Degrade gracefully.

2.2.4 *Traits of An Expert in Control System Design*

Considering the traits and capabilities outlined above, the following specific traits of an expert in control systems are offered as being among the most fundamental [20] :

- o Obtains a realistic model of the dynamical system,
- o Chooses the design method based on system diagnosis, constraints and specifications,
- o Designs to specifications and constraints,
- o Understands the limitations of the chosen method,
- o Implements trade offs, and
- o Justifies and documents his design.

2.2.5 *Comments*

As was presented above, the basic problem of a control system designer is to create or modify a given dynamical system. The complexity of the design process can be transfered from the shoulders of the design engineer to the knowledge sources of an expert system. This is achieved by integrating *declarative knowledge* with *procedural knowledge* in an interactive computing environment. *Part 3* of this thesis presents the design and implementation of a hierarchical expert system which integrates both declarative knowledge and procedural knowledge required for designing linear mono and multivariable systems using feedback state space approach.

2.3 EXPERT SYSTEMS FOR PROCESS MANAGEMENT

A potential application of expert system technology in the field of process control is in the area of high level process management. Plant management staff are now being asked to consider yet another new technology. However, it is not clear whether this new option is an evolutionary change from present control activities or yet another revolutionary path.

The ideas reported in this section are based on the paper entitled "Expert Systems as a Stimulus to Improved Process Control" by Beaverstock, Bristol and Fortin 1985 [3].

2.3.1 Manager's Role in a Plant

It is well known that plants operate on a concept of responsibility. Today's popular management concepts regarding productivity improvement are based on establishing areas of responsibility for workers and managers alike. This new philosophy has been given the name *"distributed management"*. While hierarchy still exists, individuals in lower levels are expected to develop operational knowledge and show innovation in improving their unit's operation, as well as being flexible in handling unexpected problems. Toward the top of the hierarchy, individuals are measured by their:

- o experience in strategic planning,
- o openness for risk taking,
- o interpretation of information,
- o success in providing a proper work environment, and
- o ability to delegate authority and responsibility.

2.3.2 *How Expert Systems Could Help Process Managers*

Expert systems certainly possess the potential to take on more of the attributes associated with the managerial function, such as equipment scheduling, production planning, or even determining the best operating conditions. Data bases exist today to cover plant operations. The information is available as either scanned, *periodic time-based* (second, minute, hour, shift, month, or year) or *event-based* (batch finished or pallet shipped). Unfortunately, there is no way to relate event and periodic information unless some extensive time stamping methods are used. The standardization and coordination of data bases are key process control problems necessary to effective distributed control. Knowledge bases can replace the rigidity of such conventional data bases and provide a much more robust description of the plant. They can organize the system in a clear and much more useful form. Eventually, this new approach may allow automatic translation between different application and marketing perspectives, thus eliminating the need for a central data base altogether.

The impact of expert systems on plant managment personnel will depend on the scope and source of the expert knowledge. Studies have shown [3] that the user receives no credit even though he is given the responsibility to accept or reject the system proposal. If the expert system is correct, credit is given to the system. If recommendation is wrong and it is rejected by the manager, emphasis will focus on why a wrong conclusion was reached, rather than on the user's insight in correcting the situation. Such possibilities must be considered very closely for expert systems to be accepted and successful on the managerial level. While such systems can provide inference, managers must be given the opportunity to provide perception. Additionally, interpretation or operational information and value judgements are considered managerial prerogatives that will not be easily relinguished. Expert systems will

certainly improve in their ability to take over managerial type functions and will be used where they can be viewed as noncompetitive co-workers, prepared to serve in either a private advisory role or totally replace particular, well-defined functions.

2.4 SOME EXAMPLES OF EXPERT SYSTEMS FOR PROCESS CONTROL

Since the thesis is concerned with building an expert system for computer process control, it is appropriate to give some examples of such systems which have been already developed or are in the process of being developed. Therefore, this section intends to make a short review of several expert systems used in process control.

2.4.1 FALCON

FALCON, an acronym for Fault Analyser Consultant, is an expert system to aid an operator in detecting and identifying process faults. It is being developed as a joint project by the University of Delaware, E.I. du Pont de Nemours and Company, and The Foxboro Company to do fault detection and analysis on an adipic acid reactor. The simulation package used for FALCON to simulate the adipic acid reactor is DESLSIM and covers a wide range of operating conditions. It has over 200 differential equations and over 1100 variables. Even on a VAX-11/780, FALCON could not run in real time. The problem was solved by running the simulation over the weekend, saving the resulting time histories in files, and then using those files to provide data to FALCON. At present FALCON has been installed, running on the Micro-VAX-II using Lisp and Fortran 77, and testing on the process is under way. More information about this expert system can be found in [4] and [43].

2.4.2 *Heat Exchanger*

Another expert system for fault detection and analysis is being developed to run on a variety of processes (as opposed to a single, specific process). Emphasis is being placed on development of tools to greatly reduce the effort required for knowledge-base engineering. Initial tests have been on a pilot plant heat exchanger. The heat exchanger demonstration has been implemented on a Xerox 1186 Lisp machine, using the LOOPS software environment. Unfortunately, the only source of information concerning this expert system is the IEEE Control Systems Magazine, December 1987 [44].

2.4.3 *EXACT*

EXACT is an expert system application to tune proportional-integral-derivative (PID) controllers [25], [26], and uses heuristics to tune a PID controller, with no need for process modelling. It has been developed in the form of an expert system, then delivered on a microprocessor as an assembly language program with a widely acceptance.

2.4.4 *Flow Meter Consultant*

The Flow Meter Consultant is planned for implementation as an expert system. It will be a sales support tool, helping sales personnel to translate customer flow-metering requirements, into recommendations for specific flow meters, including prices and a completed order form. There are two special requirements for this expert system: first that it ties into a data base of flow meter products currently being defined independently, and second that the human interface matches the general appearance and

function of the human interface for an associated control system product line. The Flow Meter Consultant will be developed using an expert system shell - Personal Consultant Plus - on a personal computer, and eventually delivered as a C program [44]. (Personal Consultant Plus is a trademark of Texas Instruments).

2.4.5 DELTA

DELTA stands for Diesel Electic Locomotive Troubleshooting Aid and is being used in railroad repair shops to assist maintenance personnel in isolating and repairing a large variety of diesel electric locomotive faults. DELTA asks the user a series of questions about a locomotive that is not working correctly. Using the answers to these questions along with the knowledge the system has about different types of locomotive problems, DELTA draws a conclusion about what the fault is and recommends corrective action. On request, this system can display detailed drawings of various components of the locomotive at any time during a diagnosis to assist the user in understanding how and why a certain fault is being considered. Repair procedures are also displayed on the screen to help in the repairs recommended by DELTA. More about Delta in [32] and [3].

2.4.6 ACE

The ACE expert system (Automated Cable Expertise) is being developed at Bell Laboratories. It handles hundreds of telephone cable maintenance reports daily. These reports come from a data management and report generation system. Each night, this expert system examines these reports to determine what repairs are needed and where, a job that would take a week for a human. The main sources of ACE's knowledge are a

large data base system and primers on maintenance analysis strategies. Based on its analysis, ACE suggests the proper response to immediately resolve a problem which ACE retrieves from a data base of recommended repair procedures. This type of maintenance system could also be useful in any process environment where routine problems are encountered on a regular basis ([32] and [3]).

2.4.7 MAID

MAID stands for Multivariable Analytical and Interactive Design. This expert system is a prototype for the investigation of applying expert system techniques in control system design. It was implemented using an expert system shell called X_i (Expertech, 1985) and runs on an IBM PC/G terminal which acts as a stand-alone personal computer. In the initial development stage, the declarative knowledge was represented by IF-THEN rules and the control system analysis and design facilities were provided by the $MATRIX_X$ package (Integrated Systems 1984).

MAID can aid a designer by guiding him through the design process of *multivariable systems* acting as a designer's assistant. As shown in figure 2.4, the approach used for implementing this expert system differs from the approach used for the HESCPC system implementation. That is, in the case of MAID, the declarative and procedural knowledge are not integrated to work as a whole. The user acts as an interface between the two. In the case of HESCPC expert system the declarative knowledge represented by M-Prolog rules and the procedural knowledge embedded in a Fortran library talk together acting as one system with control at the task level also provided. Thus, the HESCPC expert could redirect the execution of a process control CAD task depending on the obtained results. This is illustrated in figure 2.5 and

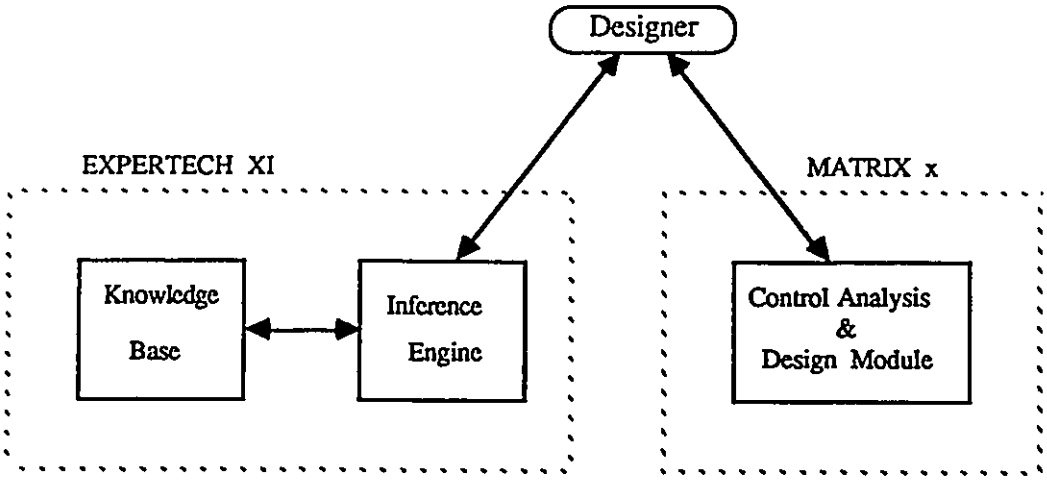


Figure 2.4 : MAID Expert System [from Pang and MacFarlane 1986]

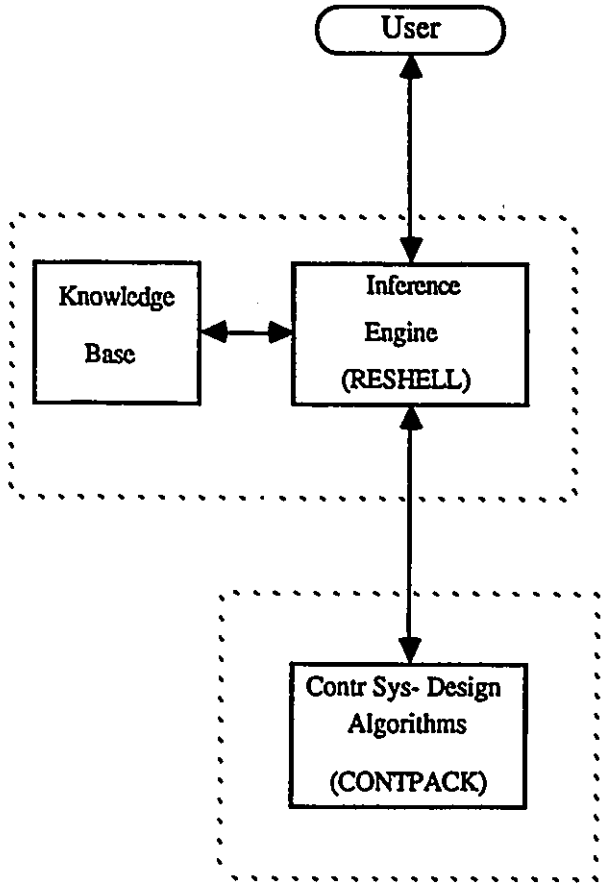


Figure 2.5 : HESCPC Expert System

presented in detail in the rest of this thesis. For the next development stage, MAID is planned to be reimplemented using KEE (Knowledge Engineering Environment), an expert system development tool marked by Intellicorp, and its knowledge represented by frames. More information about MAID can be found in [36].

2.4.8 CACE-III

Computer-Aided Control Engineering for third generation (CACE-III) is an expert system developed at Rensselare Polytechnical Institute, as a research tool, for the United States Army. It has demonstrated a high level of competence in automatically designing lead-lag compensation for a single-input, single-output linear plant and has been extended to the compensation of sample data and nonlinear plants. CACE-III runs on VAX-11/785 under the VAX/VMS operating system. Its inference engine is provided by DELPHI, a proprietary rule-based system shell developed at General Electric from DELTA's original design.

Like the HESCPC expert system presented in this thesis, CACE-III integrates the control system design rules, written in the VAXLISP version of Common Lisp, with procedural routines contained in two packages:

- The Cambridge Linear Analysis and Design Program (CLADP) which contains more than thirty separate routines to perform design, analysis and simulation functions for both *single-input, single-output* and *multiple-input, multiple-output* systems in either the time or frequency domain, and
- The SIMNON package which supports simulation of nonlinear systems.

A detailed presentation of CACE-III expert system can be found in [20].

PART 3

DESIGN AND IMPLEMENTATION OF THE HESCPC SYSTEM

The HESCPC expert system is a hierarchical structure of specialized experts which incorporates two kinds of knowledge:

- o Declarative knowledge represented by M-Prolog rules, and
- o Procedural knowledge embedded in a control system design algorithms Fortran library called CONTPACK.

From this point of view the HESCPC system is a three-block system:

- o Declarative Knowledge Block (DKB),
- o Declarative-Procedural Knowledge Interface Block (DPKIB), and
- o Procedural Knowledge Block (PKB).

This is shown in figure 3.1.

Before discussing the architecture of the HESCPC system and its implementation, a short review of the tool that made possible the building of this expert system and some remarks concerning knowledge representation are presented.

3.1 TOOLS USED FOR THE HESCPC IMPLEMENTATION

As was stated in Section 2.1.5, four key resources are necessary for developing an expert system:

- o a human expert,
- o a knowledge engineer,

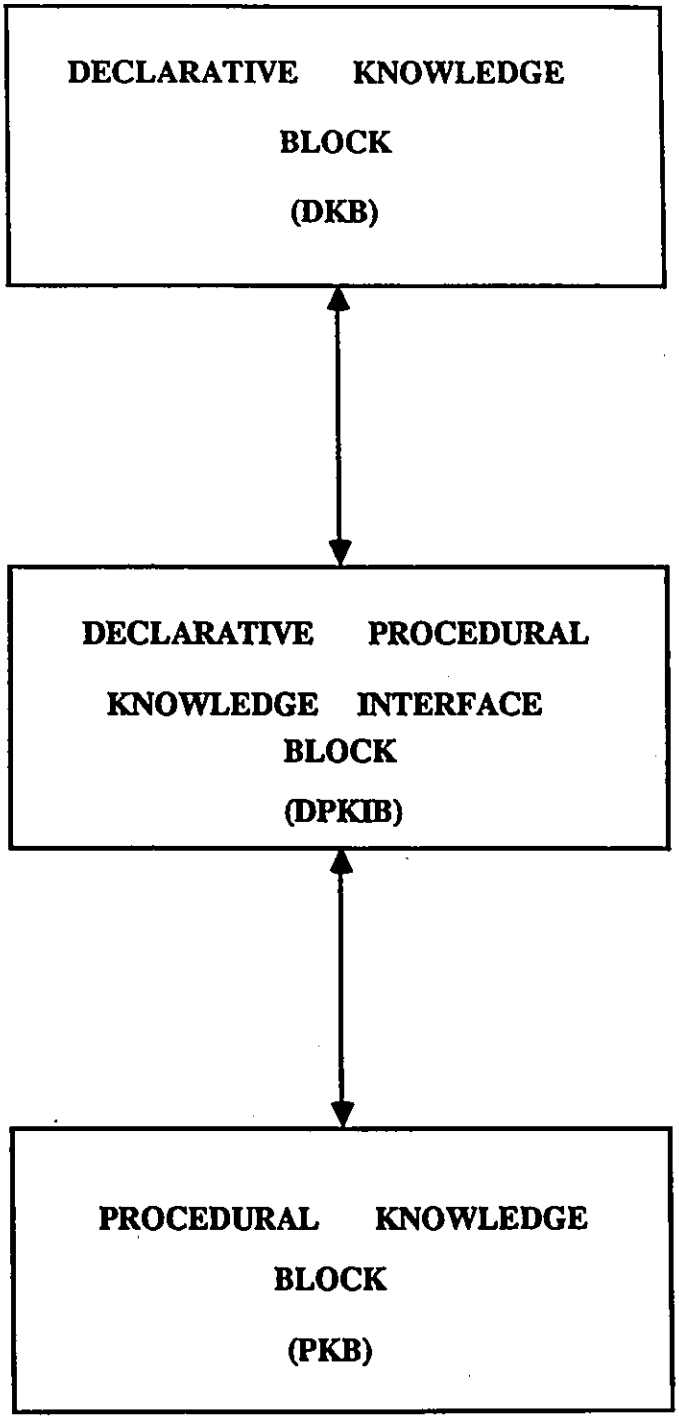


Figure 3.1 : The HESCPC expert as a Three-Block System

- a computer, and
- an expert system tool called a shell.

The HESCPC expert system was implemented on a VAX/VMS using a tool called Reshell.

3.1.1 A brief overview of Reshell

It is known that expert systems are computer programs which are built to represent and apply actual expert knowledge of specific areas. As shown in figure 3.2, they consist of three major components: a *knowledge base*, an *inference engine* and a *user interface*. They differ from traditional data processing systems because they:

- use symbolic representation, symbolic inference and heuristic search,
- have the ability to handle explanations, noisy or fuzzy data,
- acquire knowledge through interaction with the user and maintain consistency between the newly acquired knowledge and the "past experience".

Such an expert system with an empty knowledge base is called a *shell*.

Reshell is a *blackboard system* developed by the Canada Centre for Remote Sensing (CCRS) and it is written in M-Prolog. The organization of the Reshell expert system shell is shown in figure 3.3. Its key functional components are:

- the data interface,
- the scheduler,
- the meta rule interpreter,
- the object rule interpreter,
- the frame processor,

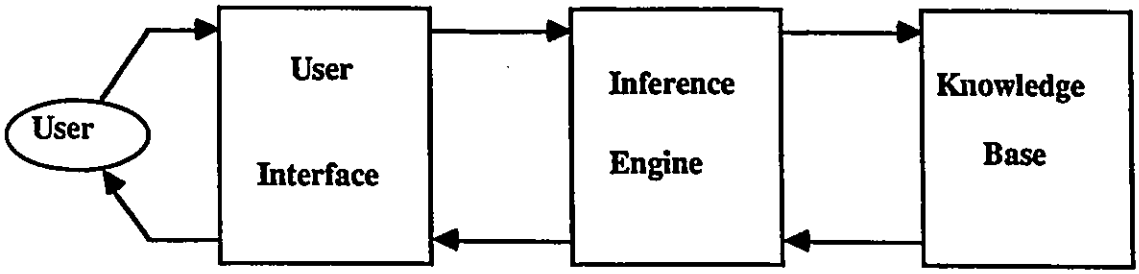


Figure. 3.2 : General block diagram of an Expert System
[from Louis E. Frenzel, Jr., 1987,
Crash Course in Artificial Intelligence and
Expert Systems]

- the arbitrator, and
- the operator.

The fundamental data structures in Reshell are:

- object values,
- frames,
- object rules,
- meta rules,
- operator knowledge, and
- LDIAS Task Interface (LTI).

The primary data storage elements of this shell are:

- the blackboard,
- the rule bases,
- the frame data base, and
- the operator knowledge bases (OKB).

Other components, not shown in figure 3.3 are:

- the expert system controller, which transparently switches control of the Reshell components between different experts,
- the procedures for accessing and controlling Reshell components from applications, and
- the human interface and development components.

Reshell was designed in such a way to permit the implementation of hierarchical structures of expert systems placed on different levels of this hierarchy. The

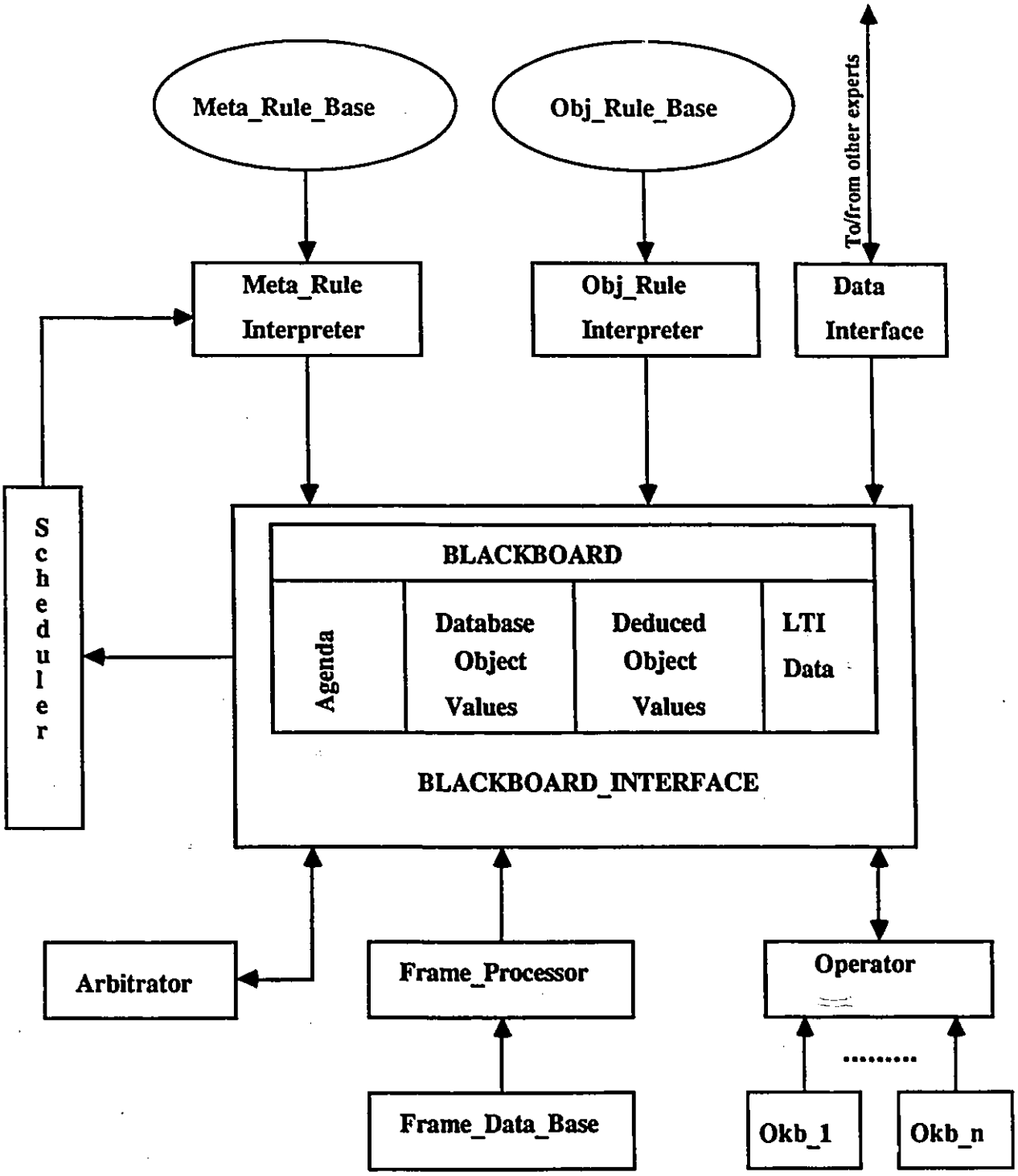


Fig. 3.3 : Architecture of RESHELL
[from G. M. Karam, A. -M. Cardill, F. Braun 1987 Knowledge Engineering with Reshell]

components of this structure can communicate among themselves through their blackboards via the Data Interface as shown in figure 3.3. Figure 3.4 illustrates the hierarchy concept which can be implemented using the Reshell system. Reshell was successfuly used for the implementation of hierarchical structures of expert systems for updating forestry maps with LANDSAT data [11].

Using Reshell the knowledge can be represented either by frames or production rules and the inference can be made in one of the two search modes: *depth-first backward chaining* or *breadth-first forward chaining*. Both these modes were used in the HESCPC system.

A detailed presentation of this powerful tool is given in Reshell Beginner's Guide [22], Knowledge Engineering with Reshell [23], and Reshell Software Reference [24].

3.1.2 Rules vs. Frames

Knowledge representation is still a central topic of research in Artificial Intelligence. This section intends to present some remarks concerning how to represent the expert's knowledge in an expert system.

Two main approaches are currently used for knowledge representation:

- o Production rules: IF [condition(s)] THEN [action(s)], and
- o Structured objects.

A. *Production rules* approach was used in most early and conventional expert systems and it is still used today. The *advantages* often claimed for the use of this kind of knowledge representation, as reported by Pang and MacFarlane [36], are:

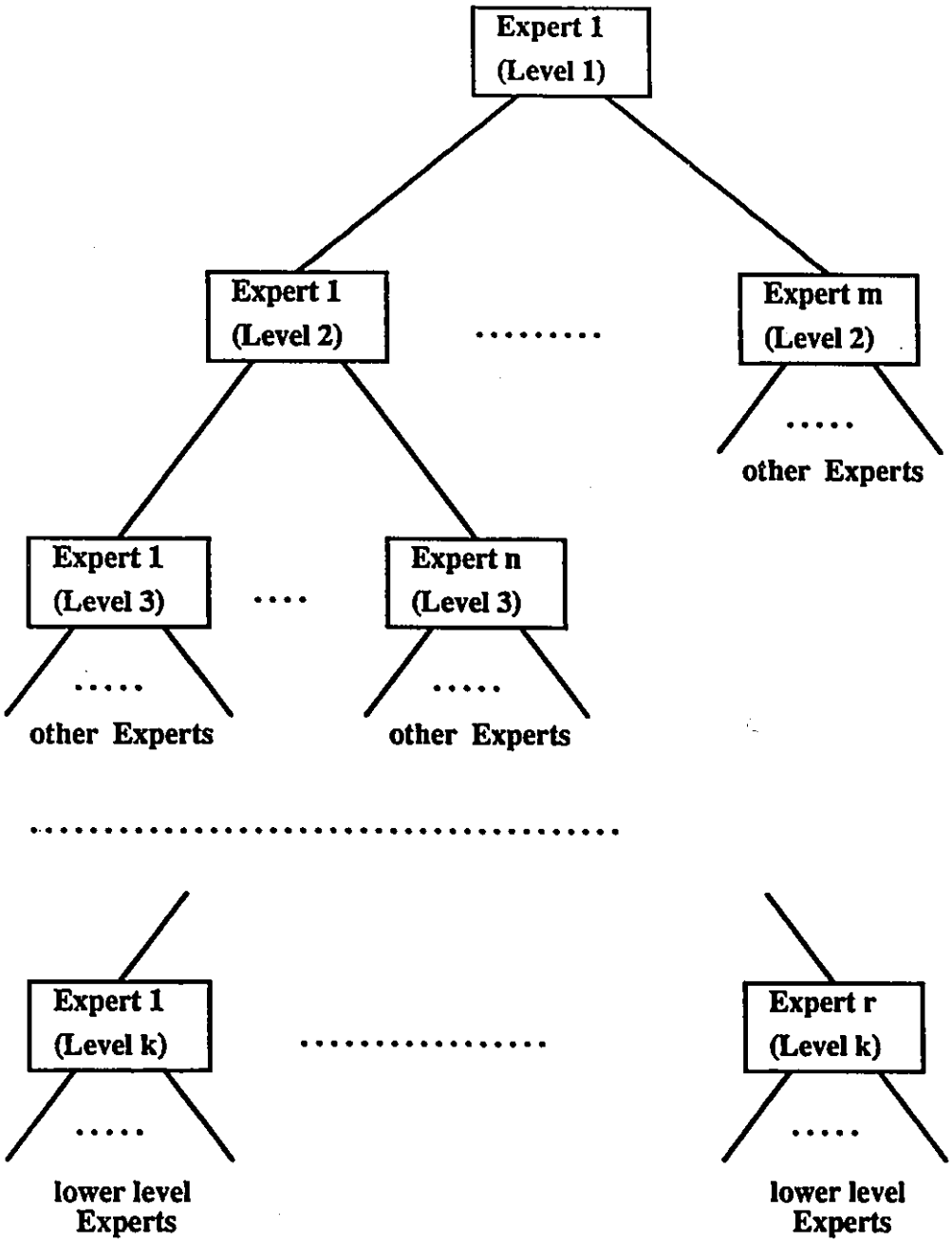


Figure 3.4 : Hierarchical Organization of a Reshell-based Expert System

- It represents knowledge in a declarative manner. Interaction among the rules may lead to unplanned and interesting results.
- It allows for a uniform representation of knowledge.
- It allows for incremental growth of the knowledge base. New rules can be added easily.
- It represents naturally occurring chunks of knowledge.

However they outline the following *disadvantages* of production rules:

- It is an inefficient way of representing knowledge and results with little flexibility to handle the search especially if the knowledge base becomes large.
- The expressive power is inadequate for representing concepts and relationships among objects.
- It is difficult to see the consequences of adding a new rule to the system. It may lead to an undersirable interaction and result in the knowledge base containing contradictory and circular rules.

B. Examples of representing knowledge by *structured objects* are *semantic networks* and *frames*. Using this approach the knowledge base is typically a hierarchical network of objects, concepts or events. Their interrelations are represented by a framelike or node-and-link structure. *A frame* is a generic data structure containing any number of categories of information called *slots*, where this information is associated with the subject of the frame [36]. Each frame, which has a name similar to the concept of a header in a list, defines a semi-independent body of knowledge which can be both procedural and declarative. Frames may be linked together to form a hierarchical classification of domain knowledge and allow for inheritance. Pang who has evolved

the concept of frame, first developed by Marvin Minsky in 1975, outlines the following *advantages* of the frame-based approach [36] :

- Frames unify both the procedural and declarative expressions of knowledge.
- The knowledge base of a frame-based system is extremely modular. This provides a natural way of representing components of expertise. Also, the knowledge base can be maintained more easily.
- Each frame can represent appropriate knowledge as default values.
- Each frame is an object which represents an independent body of knowledge.
- Once the domain expertise was organized into frames, it is relatively simple to represent the procedural aspects as rules within the knowledge base.
- Frames can be defined as specializations of more general frames, leading to a hierarchical classification of the domain knowledge.
- Frames can be linked together to have inheritance relationships.
- Uncertainty in the design process calls for flexibility and this can be provided if the frames are semi-independent.
- The flexibility of the frames allows the experienced designer to vary the sequence in which the frames are used and therefore provides him with a more powerful structure for handling unanticipated types of design problem.

3.1.3 Comments

The hierarchical concept presented in figure 3.4, which can be implemented in a Reshell-based system, is in fact a *semantic network* where each element is a *production rule expert system* which belongs to a class, or sub class of experts. As will be presented in the next section, all the experts composing the HESCPC system are related among themselves through very well defined relationships. By partitioning the *global*

knowledge base into small knowledge bases distributed on a hierarchical structure the main disadvantages of production rule systems outlined above have been eliminated. For example:

1. The exhaustive search has been eliminated by keeping small knowledge bases. At a given time, the search is done in *only one expert knowledge base.*
2. The consequences of adding a new rule to the system is not so difficult to see because the *knowledge bases are not large.*
3. The relationships between concepts are very well defined in the *semantic network* representation.

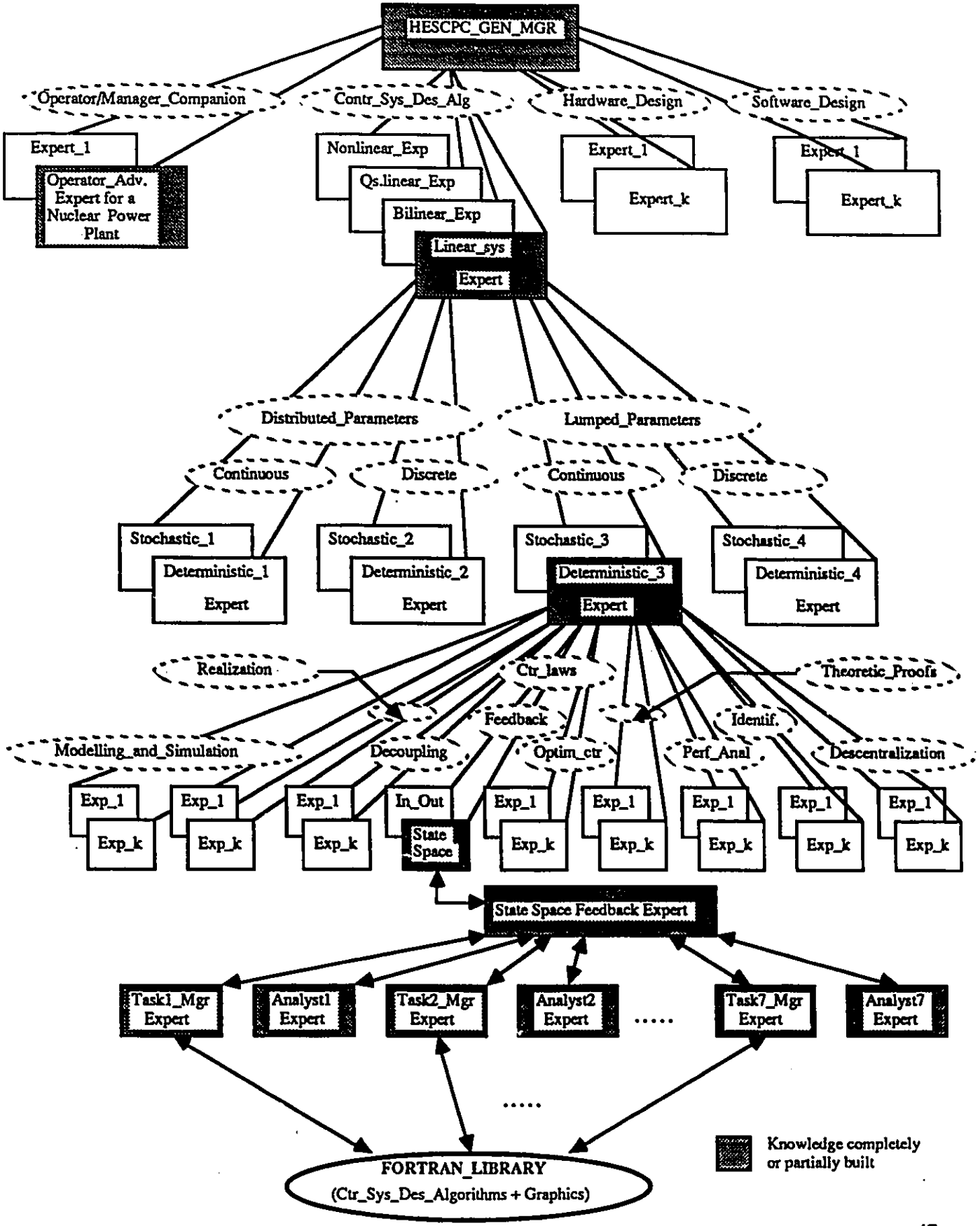
For the HESCPC's implementation the *"semantic network-production rule"* approach has been considered in the naive belief that this representation is much closer to the human way of thinking. As a future work, the HESCPC system will be implemented using frames approach to explore the possibilities of this kind of knowledge representation for process control domain and verify the advantages of frames outlined above concerning the integration of declarative-procedural knowledge.

3.2 ARCHITECTURE OF THE HESCPC EXPERT SYSTEM

The HESCPC expert system was designed following a vertical hierarchical architecture. The global system was decomposed into a number of experts able to exchange information and controls through their blackboards. Thus, this society of experts could cooperatively work in solving specific control system problems.

As shown in figure 3.5, at the top of the hierarchy, a *General Manager Expert* having knowledge about which *expert-class* and *problem-class* (in the selected

Figure 3.5 : Architecture of the Hierarchical Expert System for Computer Process Control



expert-class) can be selected directs the user to the next level of the hierarchy activating a certain expert depending on the user input. On the next level, the user is querried by the activated expert about the specific parameters that describe the *target problem* to be solved. These parameters are needed to further go down on the hierarchy for locating a *specialized problem expert* which has the required knowledge in solving the user's *target problem*. At this stage of development, the HESCPC system has *two specialized problem experts:*

1. Operator-Adviser Expert for a Nuclear Power Plant, and
2. State Space Feedback Expert

which will be discussed later on.

This process of gathering parameters may continue on more than one level (depending on the *target problem* description) until the desired *specialized problem expert* is reached. At this point this expert asks the user for the *input data* necessary to solve the *target problem*. After collecting the input data, the *specialized problem expert* starts the reasoning process using the *solving problem knowledge* embedded into its knowledge base.

Usually, for control system design the *target problem* is so complex that it has to be *broken down* into *subproblems*. Thus, the *specialized problem expert* must also contain knowledge about this *broken down* process and which expert in the hierarchy is able to solve each *subproblem*. In this way, the *specialized problem expert* activates the proper expert with proper data for solving a certain *subproblem*. In turn, the latest expert could contain knowledge about how to further break down the subproblem into *sub-subproblems* and which experts in the hierarchy can solve these *sub-subproblems*. This process can continue until a reasonable complexity of the *sub-subproblems* is reached.

After doing their job, the *subproblem experts* report the results to the *specialized problem expert* which in turn reports the global conclusion (solution) of the *target problem* to the user.

Because for process control design the last *sub-subproblems* resulted from the broken down process are solved by specialized algorithms, at the lowest level of the hierarchy resides a Control System Design Fortran library (CONTPACK) which will be briefly discussed in Section 3.2.3.

3.2.1 Declarative Knowledge Block (DKB)

The experts defined in the Declarative Knowledge Block (DKB) are shown in figure 3.6. In fact, each expert composing this block (except the Operator-Adviser expert) acts as a manager to the user guidance in reaching the proper *specialized expert* for solving the *target problem*. At this stage of development, only 5 of these experts have their knowledge bases completely or partially built. A short functional description of the 5 experts follows.

A. The HESCPC_GEN_MGR Expert is the top expert of the implemented hierarchy and first activated when a HESCPC session starts up. Being the general manager of the system, it knows that four classes of experts can be selected, and that under each expert class a proper expert can be activated depending on the problem class the user is interested in.

- *The Operator/Manager-Companion expert class* intends to include experts for process operators and process managers as described in Sections 2.1, and 2.3 respectively of this thesis.

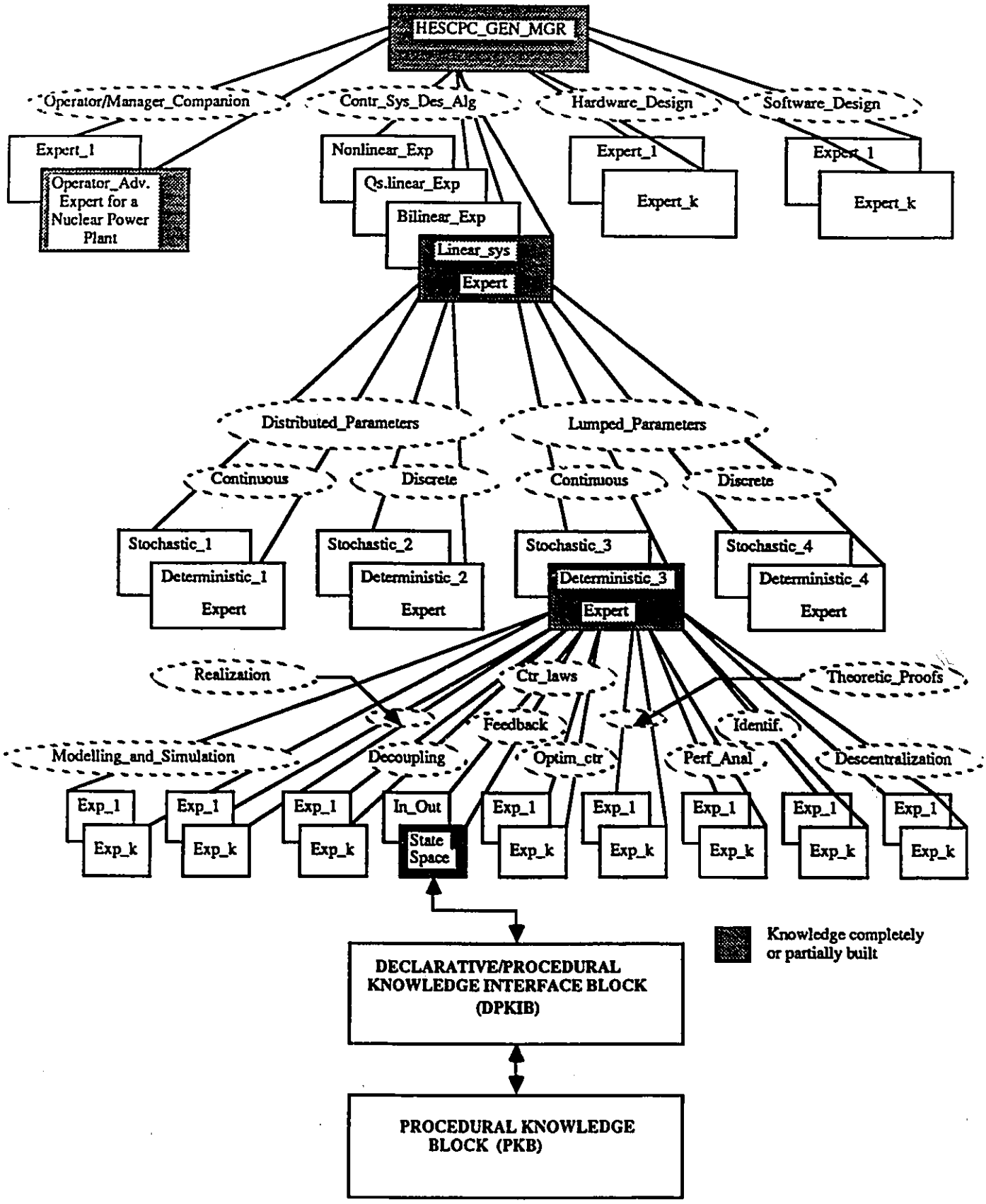


Figure 3.6 : Declarative Knowledge Block (DKB)

- o *The Control_System_Design_Algorithms expert class* includes experts dealing with four problem classes:
  - linear systems,
  - bilinear systems,
  - quasi-linear systems, and
  - nonlinear systems.
- o Under *the Hardware_Design expert class,* R1 like experts will be build for computerized process control system hardware configuration. (R1 is an expert system for configuring DEC VAX computer systems to meet customer needs. See McDermott 1980, [31]).
- o *The Software_Design expert class* will include the same kind of experts as the Hardware_Design expert class but for the software configuration for computer process control.

At this time the HESCPC_GEN_MGR expert has an incomplete knowledge base.

B. **The Operator_Adviser Expert for a Nuclear Plant** was implemented in the summer of 1987 as a demonstration for the Atomic Energy of Canada Ltd. It is a small data-driven expert with an incomplete knowledge base. The idea was to explore the possibilities of implementing such an expert system using Reshell. This expert is one of the two *specialized problem experts* implemented into the HESCPC structure at this stage.

As shown in figure 3.7, the Operator_Adviser expert system for a nuclear power plant has an inference engine which is part of Reshell and a knowledge base containing the knowledge captured from Parcy 1982 [38]. The *Data File* is created and updated at a fixed rate by the *Data Acquisition System* which is not part of the expert system. For this demonstration the following *ten sensors* which collect relevant data from a nuclear

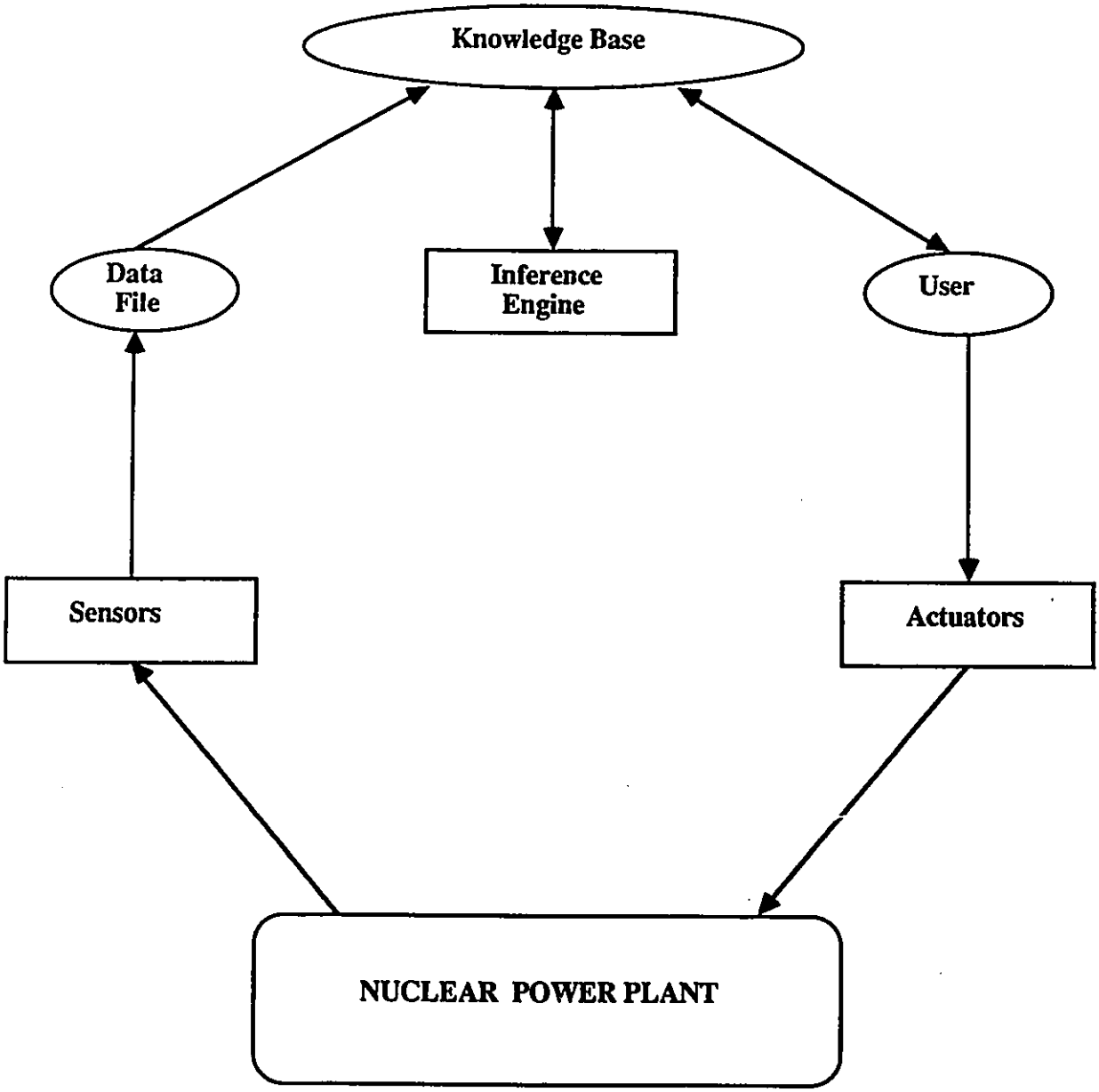


Figure 3. 7 : An Operator-Adviser Expert System for a Nuclear Power Plant

power plant were considered:

- reactivity metre,
- reactivity balance,
- drift speed,
- temperature sensor trtc_1,
- temperature sensor trtc_2,
- fuel bar variation,
- primary flow drift variation,
- secondary flow drift variation,
- steam generator, and
- gain breaking.

When the user wants to check the state of the reactor the Data File is loaded into the expert's blackboard. The goal of the expert is to find *"what is the state of the reactor"*. At this point the inference engine is invoked and, using the knowledge base, the inference process starts. Depending on the sensor data, the expert will come up with a diagnosis of the reactor and an advice (if necessary) to the user. The format of this message is:

"The nuclear reactor is in:

state ---> (reactivity, ..., breakdown)

condition ---> (acceptable-time, ..., inacceptable)

cause ---> (where the trouble is located)

action ---> (action/no action to be taken)."

For the next step, the human operator takes or initiates the recommended actions by the expert. Sample runs of this expert for different sensor data are given in *Appendix*. Note that two classes of breakdowns were considered: *internal* and *external breakdowns*.

C. **The Linear Systems Expert** acts as a manager to the user's guidance for locating the specialized expert on the *target problem*. By querring the user it collects:

- the parameter types of the linear system *(distributed or lumped)*,
- the linear system class *(continuous or discrete)*, and
- the linear system type *(deterministic or stochastic)*.

At this stage, this expert has an incomplete knowledge base.

D. **The Deterministic_3 Expert** continues to gather the parameters that describe the *target problem:*

- to find the area in which the problem to be solved fits *(realization, modeling and simulation, control laws, identification descentralization, theoretic proofs* or *performance analysis)*,
- if the user chooses the control laws area, this expert will ask him what kind of design he is faced with *(optimal control, feedback,* or *decoupling)*,
- if the feedback design is selected, the expert querries the user about the method he intends to use for this design *(input-output* or *state space)*.

If, for exampie, the state space method has been selected the control will be passed over to the State Space Expert before reaching the *specialized problem expert*. The Deterministic_3 Expert has an incomplete knowledge base at the present time.

E. **The State Space Expert** is located on the last level of the hierarchy in the Declarative Knowledge Block (DKB). It also acts as a manager and collects the last parameter that describes the *target problem:*

- what kind of state space problem has to be solved *(state feedback, output feedback, canonical forms* or *property analysis)*.

The State Space Expert has an incomplete knowledge base and the user must select

"state feedback" to reach the State Space Feedback Expert (part of DPKIB) which is the *second specialized problem expert* implemented at this stage into the HESCPC system.

3.2.2 Declarative-Procedural Knowledge Interface Block (DPKIB)

The components of the Declarative-Procedural Knowledge Interface Block (DPKIB) are presented in figure 3.8. All the experts defined in this block have their knowledge bases completely built. A functional description of each component of the DPKIB and how this block works follows.

A. **The State Space Feedback Expert (SSFE)** is the second *specialized problem expert* implemented into the HESCPC system and can be used by a control system designer who has as a *target problem* the design of *mono* or *multivariable linear systems* using the *feedback state space approach.* It is activated by the State Space Expert (part of the Declarative Knowledge Block) and together with its 14 children and 51 Fortran routines is able to solve the following kind of *target problem:*

Given a linear time invariant system described by the equation:

$$dx/dt = Ax + Bu, \quad \text{where}$$

A is a n x n *matrix* called the *state space matrix,*

B is a m x n *matrix* called the *input matrix,*

x is the n x 1 *state space vector,*

u is the m x 1 *input vector,*

it is required: to design a feedback upon the state space variable x

$$u = Hx + Gv, \quad \text{where}$$

v is the m x 1 *reference variable,*

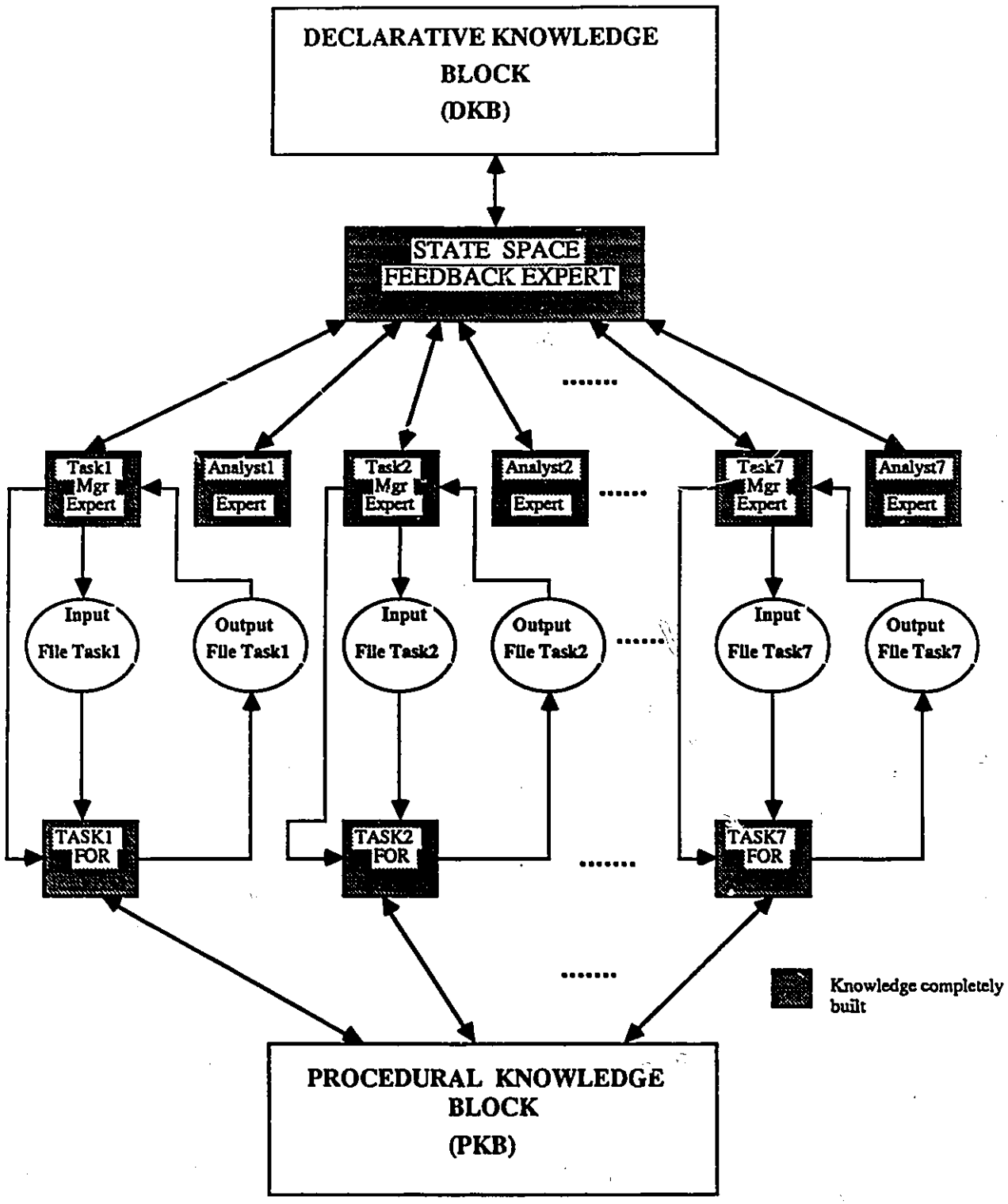


Figure 3.8 : Declarative-Procedural Knowledge Interface Block (DPKIB)

G is a given *m* x *m* *matrix*, and

H is the *desired* *m* x *n* *feedback matrix*.

For the first step, this *target problem* was decomposed into *seven subproblems*. The knowledge needed to solve these *subproblems* is embedded into *seven experts* (Task1_Mgr, Task2_Mgr, ..., Task7_Mgr) which are the children of the State Space Feedback Expert. For the next step, *each subproblem* was further broken down into a different number of *sub-subproblems*. The total number of *sub-subproblems*, for the seven *subproblems*, is 51. *Each sub-subproblem is a Fortran routine* part of the *Procedural Knowledge Block (PKB)*, which is a Fortran library specialized on control system design algorithms.

Refering to figure 3.8, the Task1_Mgr, ..., Task7_Mgr experts have the same functionality. They only differ by their knowledge bases. The same statement can be made about Analyst1, ..., Analyst7 experts and Task1_For, ..., Task7_For components.

B. **The Task*n*_Mgr Expert** is a *specialized subproblem expert*. It has an intimate knowledge about a specific *subproblem* (subproblem *n*).
All these experts, Task1_Mgr, Task2_Mgr, ..., Task7_Mgr, have their knowledge bases completely built.

C. **The Analyst*n* Expert** is a child of the State Space Feedback Expert and is attached to *subproblem n*. It knows how to analyse the output data produced by Fortran routines which solve the *subproblem n*. All these experts, Analyst1, Analyst2, ..., Analyst7, have their knowledge bases completely built.

D. **The Task*n*_For Component** represents a Fortran code which was written for *subproblem n*. It is activated as a logical function by Task*n*_Mgr expert. Each component, Task1_For, Task2_For, ..., Task7_For, contains procedural knowledge about the *sub-subproblems* needed for solving the corresponding *subproblem*. These components make the necessary calls to the Fortran routines which reside in Procedural Knowledge Block (PKB).

3.2.2.1 How the Declarative-Procedural Knowledge Interface Block Works

When the specialized problem expert (State Space Feedback Expert) is reached, the user is asked for the *target problem input data*. After collecting this data the Task1_Mgr Expert, which is a *specialized subproblem expert*, is always activated first with the proper data. This expert has an intimate knowledge about *subproblem 1*. It knows how to:

- o create an Input Data File for subproblem Task1_For with the specific data (see figure 3.8),
- o activate the subproblem Task1_For as a logical function,
- o load the Output Data File created by subproblem Task1_For into its blackboard,
- o display this data to the user, and
- o send this data to its parent - State Space Feedback Expert - .

Thus, at this point the output data generated by the first subproblem (Task1_For) is available into the blackboard of the State Space Feedback Expert. Next, this expert sends this data to the child, Analyst1 expert, which starts to analyse it. The results of this process are displayed to the user and, at the same time, sent to the parent - State Space Feedback Expert - . Depending on these results the parent expert decides which

subproblem has to be solved next and activates the corresponding child expert - another Task*n*_Mgr Expert - with the proper data. This process continues until the *target problem* is solved. Figure 3.9 depicts the sequence of actions described above. Note that a subproblem could be activated more than once, or not activated at all during a session, depending on the input data of the *target problem*.

3.2.3 Procedural Knowledge Block (PKB)

The Procedural Knowledge Block (PKB) is a Fortran library, called CONTPACK. It contains over 700 Fortran routines specialized on control system design [16].

After a careful study of this library, 51 routines were found necessary for the designing process of mono and multivariable linear systems using feedback state space approach. As an example, figure 3.10 shows the Fortran routines used by Task5_For component for solving the *subproblem 5*. Note that the nodes represent routine names and the links represent routine calls.

3.3 IMPLEMENTATION CONSIDERATIONS OF THE HESCPC SYSTEM

The HESCPC expert system was implemented using Reshell. Its highly structured knowledge is distributed among a number of experts hierarchically organized as presented in figure 3.5. The knowledge embedded in each expert is represented as IF_THEN production rules and is shared between two files:

- o <expert_name>.RBS file, and
- o <expert_name>_CA.PRO file.

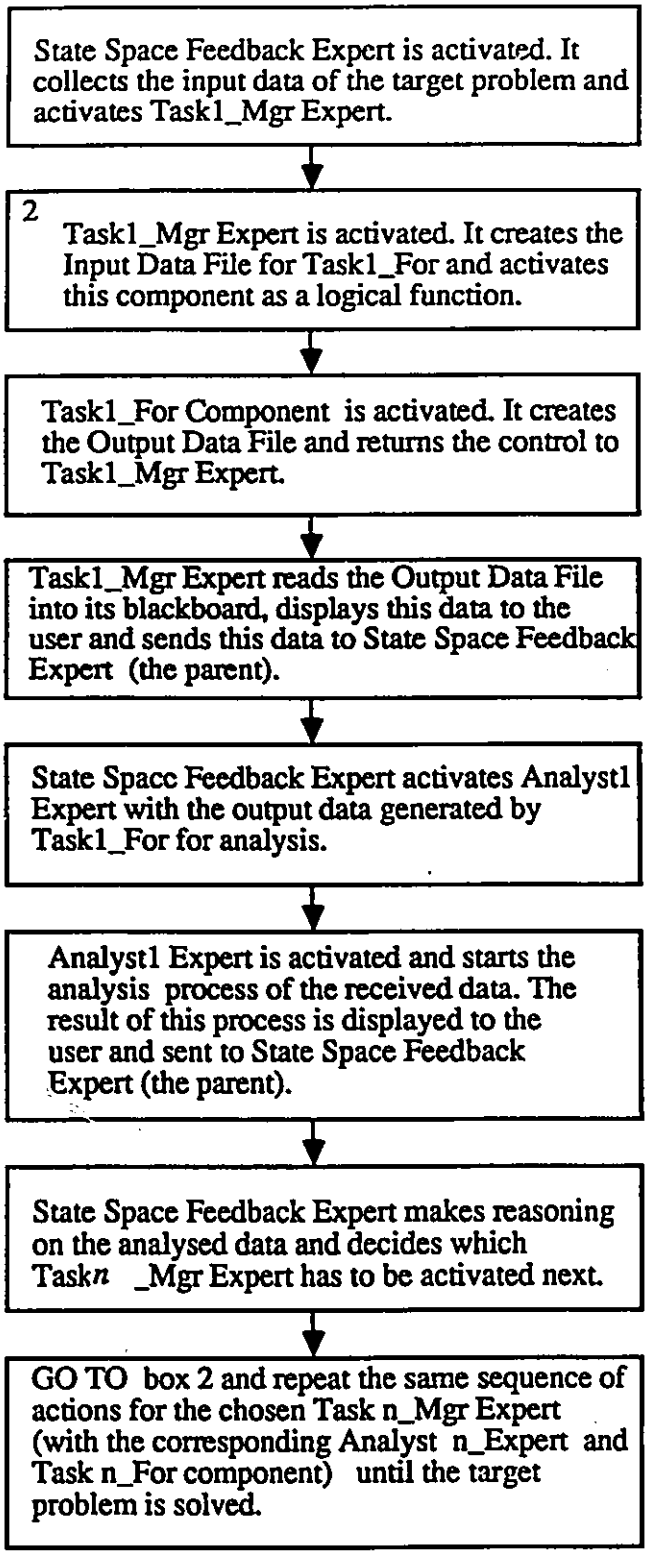


Figure 3.9 : How the Declarative-Procedural Knowledge Interface Block works

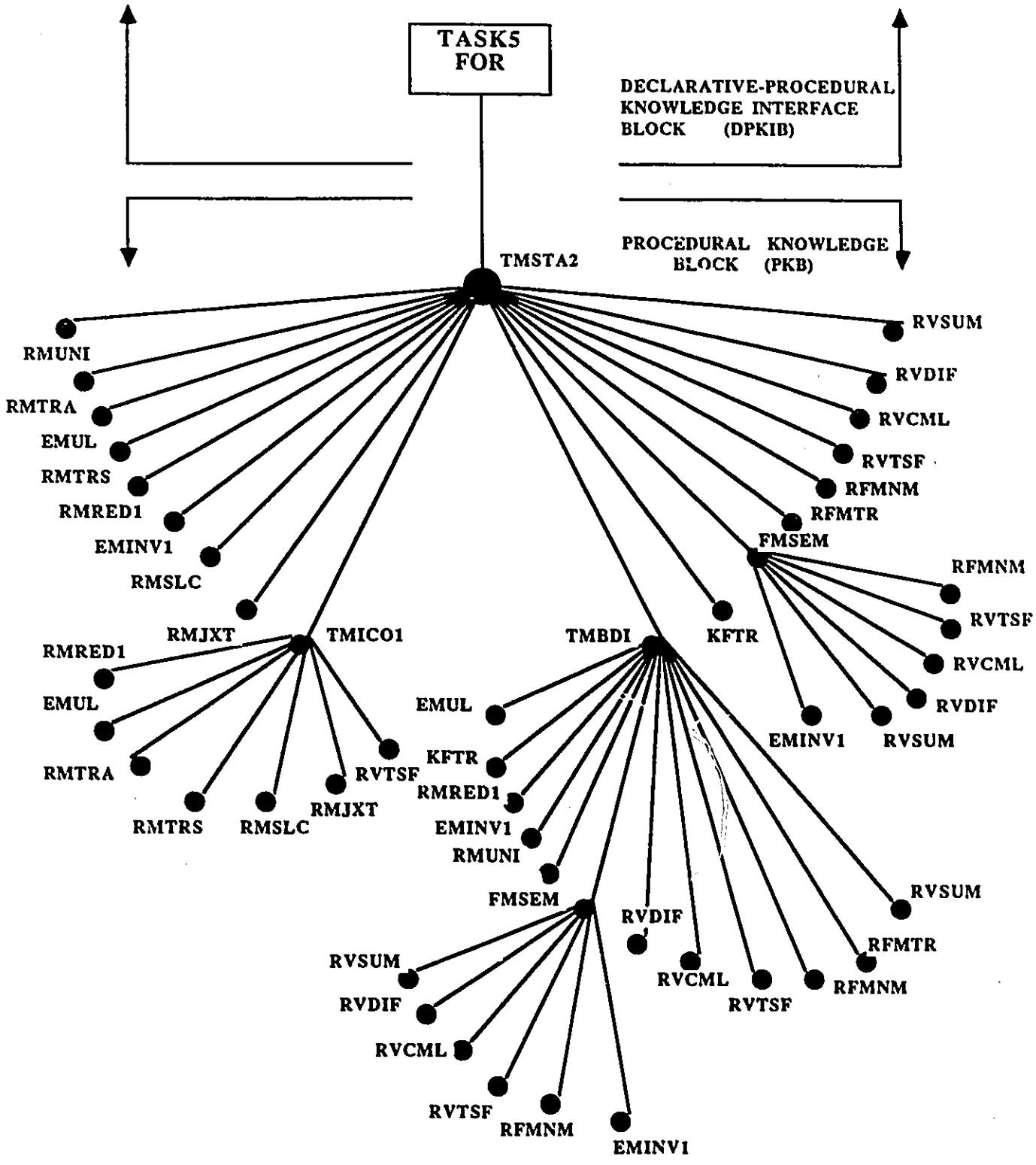


Figure 3.10 : Fortran Routines used to solve Subproblem 5

3.3.1 The <expert_name> RBS File

This file contains two kinds of rules represented in a Reshell format:

1. The *Object Rules* which are the inference rules that manipulate the *domain objects*. The *domain objects* are the entities an expert knows about and has expert knowledge of. They are represented as follows:

   obj([[, <object_name>,*], <object_attribute>, <attribute_value>, [<measure_of_belief>, <measure_of_disbelief>]]).*

   For example:

   obj([[, sensor, *], reactivity_meter, 15, [100,0]]).*

   An *Object Rule* has the following format:

   IF [<list_of_conditions>]

   THEN [<list_of_actions]

   with associated: <rule_number>, <certainty_factor>.

   The <certainty_factor> represents the degree of confidence the system has that this rule is true or false. These certainty factors are combined during the search procedure and are used to arrive at a certainty value for the final solution [23].

   Rule number 2 of State Space Feedback Expert.

   IF the state space vector dimension is known *and*

   the state space matrix is known *and*

the input matrix is known *and*

the desired new eigenvalue vector is known *and*

the calculation precision for task 4 is known

THEN the input data for task4 is OK.

This object rule is true 100% and it is represented in the Reshell format as follows:

```
knb(
    [
        obj([[*, task4_input,*], data, ok])
    ],
    [
        obj([[*, tasks_input,*], state_space_vect_dim, N]),
        obj([[*, tasks_input,*], state_space_matrix, A]),
        obj([[*, tasks_input,*], input_matrix, B]),
        obj([[*,task3_output,*],desired_new_eigenvalues_vect,R]),
        obj([[*, task4_input,*], calc_precision_task4, CP])
    ],
        100, 2
).
```

2. The *Meta Rules* are the "control program" of the expert system. They define the strategies to be used to deduce or prove the goal of the expert. To be more specific, the *meta rules* are those rules that govern the ways in which the *object rules* are controlled and used to solve a problem. Like object rules, they are IF-THEN structures with conditions and actions. Unlike the *object*

rules, the actions of *meta rules* are not deductions, they are control operations for the expert system. The rule conditions are used to select the appropriate control actions [23].

A *Meta Rule* has the following format:

IF [<list_of_conditions>]

THEN [<list_of_actions]

<phase_name>, <meta_rule_number>.

Meta rule number 15 of State Space Feedback Expert.

IF the message from Analyst3 expert was received *and*
the system is multivariable *and*
the method selected by the user is Jordan *and*
the goal "input data for task6 collected" was proven *and*
the state space vector dimension is known *and*
the input vector dimension is known *and*
the state space matrix is known *and*
the input matrix is known *and*
the desired new eigenvalue vector is known *and*
the calculation precision for task6 is known *and*
the phase is "check method"

THEN activate the expert "Task6_Mgr" in the "start" mode with
command "task6_mgr" and the following data:
"state space matrix,
input matrix,
desired new eigenvalue vector,

calculation precision for task6", *and*

set the phase to "get results m Task6_For".

This meta rule is represented in the Reshell format as follows:

```
mknb(
[
activate_the_expert__in_the__mode_with_command__and_data(task6_mgr,
start, task6_mgr,
[obj([[*, tasks_input, *], state_space_vect_dim, N, [100, 0]]),
obj([[*, tasks_input, *], input_vect_dim, M, [100, 0]]),
obj([[*, tasks_input, *], state_space_matrix, A, [100, 0]]),
obj([[*, tasks_input, *], input_matrix, B, [100, 0]]),
obj([[*, task3_output, *], desired_new_eigenvalues_vect, R, [100, 0]]),
obj([[*, task6_input, *], calc_precision_taks6, CP, [100, 0]])
]),
set_the_phase_to (get_results_from_task6)
],
[
message_from_analyst3 (received),
the_system_is (multivariable),
find__in_the_blackboard(obj([[*, method,*], jordan_wolovich, jordan]),
                                                    anywhere),
prove_the_goal(obj([[*, task6_input, *], input_data_task6, collected])),
find__in_the_blackboard(obj([[*,tasks_input,*],state_space_vect_dim,N])),
                                                    anywhere),
find__in_the_blackboard(obj([[*, tasks_input, *], input_vect_dim, M]),
```

anywhere),

find__in_the_blackboard(obj([[,tasks_input,*],state_space_matrix,A]),*

anywhere),

find__in_the_blackboard(obj([[,tasks_input,*],input_matrix,B]),anywhere),*

find__in_the_blackboard(obj([[,task3_output, *],*

desired_new_eigenvalues_vect, R]), anywhere),

find__in_the_blackboard (obj([[, task6_input, *], calc_precision_task6,*

CP]), anywhere)

],

[check_method],

15

).

3.3.2 The <expert_name>_CA.PRO File

This file contains the condition predicates and action procedures. They are M-Prolog statements that implement specific functions in *object* and *meta rules*. Some examples of conditions and actions represented in M-Prolog rules are given below:

1. Nuclear_Reactor_CA.PRO

IF the reactivity balance is abnormal *and*
the drift speed is not fast *or*
the reactivity balance is uncertain *and*
the observation of temperature sensor trtc_1 is greater
or equal to 10

THEN the big transient regime is abnormal

which was implemented as:

```
big_transient_regime(abnormal)   :-
(reactivity_balance (abnormal),
drift_speed (not_fast));
(reactivity_balance (uncertain),
observation_of (trtc_1, X),
X > = 10).
```

2. Task5_Mgr_CA.PRO

IF	the error index of Task5_For is known *and*
	the transposed matrix is known
THEN	Task5_Mgr sends this data to the parent expert

which was implemented as:

```
task5_mgr_sends_output_data_to_parent   :-
        find_in_bb(anywhere, obj([[*, task5_output, *],
                                  error_index_task5, IER])),
        put_high(return_data, obj([[*, taks5_output,*],
                  error_index_task5, IER, [100, 0]])),
        find_in_bb(anywhere, obj([[*, task5_output, *],
                  transposed_matrix_task5, [H/T]])),
        put_high(return_data, obj([[*, task5_output, *],
            transposed_matrix_task5, [H/T], [100, 0]])).
```

3. Analyst1_CA.PRO

IF the state space vector dimension is known *and*
the input vector dimension is known *and*
the control matrix rank is known *and*
the decision on the branch was made
THEN make analysis on output data from Taski_For

which was implemented as:

```
make_analysis(BRANCH)  :-
        find_in_bb(anywhere, obj([[*, tasks_input, *],
                                  state_space_vect_dim, N])),
        find_in_bb(anywhere, obj([[*, tasks_input, *],
                                  input_vect_dim, M])),
        find_in_bb(anywhere, obj([[*, task1_output, *],
                                  contr_matrix_rank_task1, IR])),
        decide_branch(N, M, IR, BRANCH), !.

decide_branch(N, M, IR, BRANCH)  :-
        N > 0,
        M > 0,
        IR >= 0,
        M = 1,
        IR = N,
        BRANCH = one,
```

```
        put_in_bb_(data_base, obj([[*, analyst, *],
                branch, BRANCH, [100, 0]])), !.

decide_branch(N, M, IR, BRACH)  :-
        N > 0,
        M > 0,
        IR >= 0,
        M = 1,
        IR < N,
        BRANCH = two,
        put_in_bb(data_base, obj([[*, analyst, *],
                branch, BRANCH, [100, 0]])), !.

decide_branch(N, M, IR, BRANCH)  :-
        N > 0,
        M > 0,
        IR >= 0,
        M > 1,
        IR = N,
        BRANCH = three,
        put_in_bb(data_base, obj([[*, analyst, *],
                branch, BRANCH, [100, 0]])), !.

decide_branch(N, M, IR, BRANCH)  :-
        N > 0,
        M > 0,
```

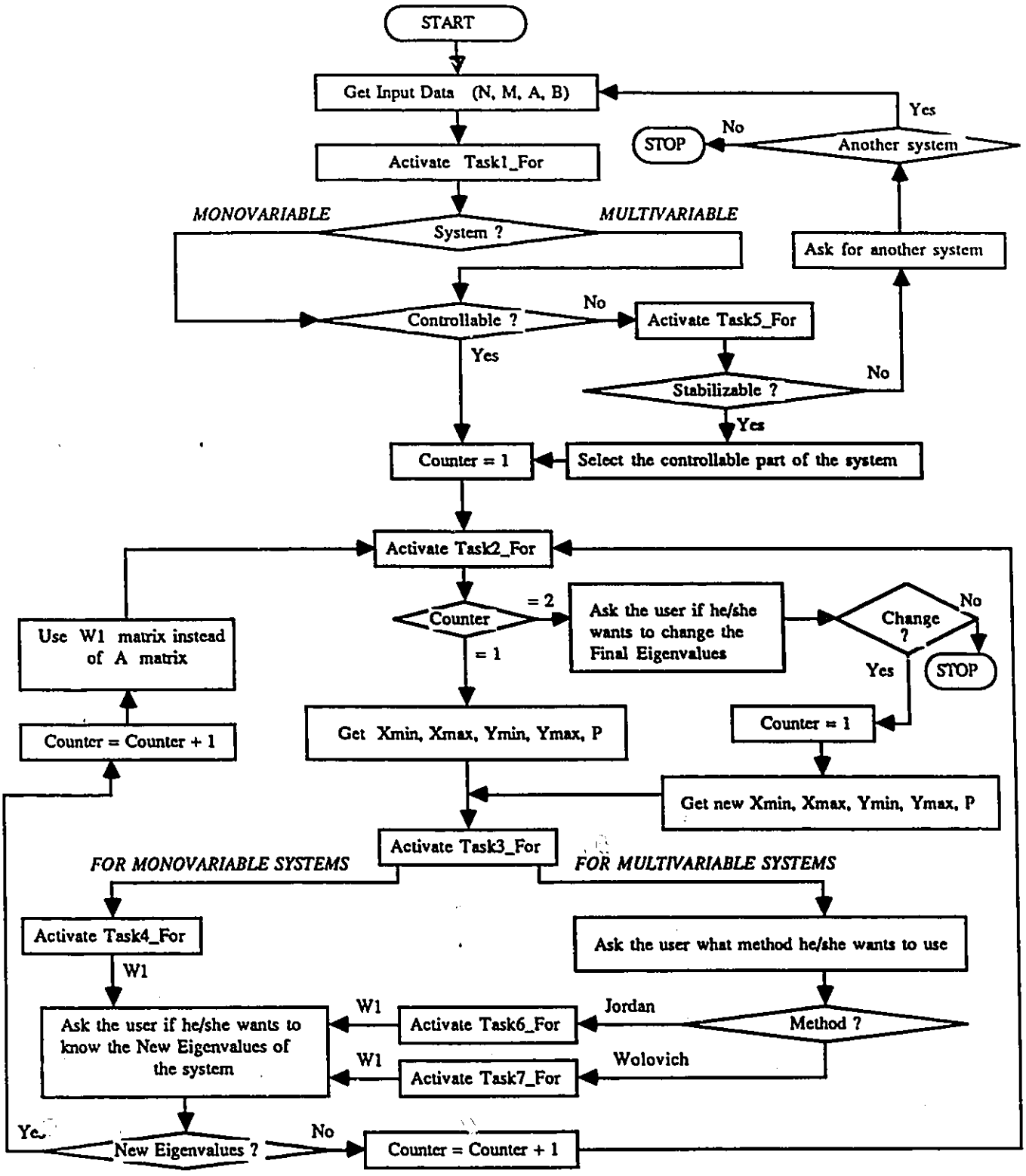
```
IR >= 1,
M > 1,
IR < N,
BRANCH = four,
put_in_bb(data_base, obj([[*, analyst, *],
        branch, BRANCH, [100, 0]])), !.
```

3.3.3 *State Space Feedback Design Heuristics*

This subsection presents the state space feedback heuristics embedded in the State Space Feedback Expert (SSFE). In fact each expert composing the HESCPC system contains a certain amount of heuristics but the most complex is the one presented below.

As depicted in figure 3.11, after the State Space Feedback Expert was reached the user is asked for the *input data* of the *target problem* , in this case, a *linear system with lumped parameters*. This data is passed to the *subproblem* Task1_For (via Task1_Mgr expert as shown in figure 3.8) which in turn makes calls to *sub-subproblems* in the Procedural Knowledge Block (PKB). The resulting data is passed back to the SSFE (via Task1_Mgr expert, see figure 3.8) which in turn passes it to the Analist1 expert for analysis. The results of this process are reported to SSFE. Thus at this point the State Space Feedback Expert knows if the linear system is mono or multivariable and if it is *controllable* or not.

1. If the linear system is <u>not</u> *controllable* the SSFE sends the required data to the *subproblem* Task5_For (via Task5_Mgr expert, see figure 3.8) which in turn



LEGEND

N ---> State Space Vector Dimension
M ---> Input Vector Dimension
A ---> State Space Matrix
B ----> Input Matrix
W1 ----> Closed Loop System Matrix

Xmin ---> Lower Limit of Real Part of Eigenvalues
Xmax ---> Upper Limit of Real Part of Eigenvalues
Ymin ---> Lower Limit of Imag. Part of Eigenvalues
Ymax ---> Upper Limit of Imag. Part of Eigenvalues
P ---> Percentage of Complex Eigenvalues of total number of Eigenvalues

Figure 3.11 : State Space Feedback Design Heuristics

makes calls to a large number of *sub-subproblems* in PKB as shown in figure 3.10. Again the output data from Task5_For is passed to Analyst5 expert (via Task5_Mgr and SSFE) for analysis. At the end of this process the State Space Feedback Expert knows if the linear system can be *stabilized* or not. If not, the user is informed and the SSFE asks for another input data for another system. If the system can be *stabilized* the user must select the controllable part of the given system and the design session continues as shown in figure 3.11.

2. If the given linear system is *controllable* the State Space Feedback Expert passes the required data to *subproblem* Task2_For (via Task2_Mgr, see figure 3.8) which *computes the system's eigenvalues*. Again, the analysing process follows, and then the subproblem Task3_For will generate the *desired new eigenvalues* for the given linear system. Next, the *system's feedback matrix* will be calculated by the subproblem Task4_For for monovariable systems, and by subproblems Task6_For or Task7_For - user choice - for multivariable systems.

3.3.4 Why an Expert System ?

Looking at the feedback state space design heuristics depicted in figure 3.11 one could ask: *why not implement this logical flow by using conventional programming techniques with Pascal, Fortran C, or some other high-level language?*. The answer is that figure 3.11 could be implemented using a conventional, high-level language, but:

1. For Process Control, such an implementation would suffer from *complexity problems* mentioned before. The more interesting and more productive approach is to consider the descriptive, *declarative-oriented programming approach* of rules to represent *what one wants the expert system to know* versus considering

the imperative, *procedure-oriented approach* of if-then-else constructs to represent *what one wants the program to do.*

2. Using an expert system approach the user is equipped with explanation facilities:
   - o *Why* - the user can ask the expert system why a question is being asked, and
   - o *Explain* - the user can ask the expert to explain its reasoning process: *how* the solution path was created and how the object values were deduced.

   Implementing figure 3.11 using a procedure-oriented language the user would not have been able to query the system to check on the logical flow of the design heuristic.

3. Using an expert system approach the knowledge base can be *incrementally increased without restructuring the code already written.* For example, the knowledge base for State Space Feedback Expert was first built for design of linear monovariable systems, therefore only a part of figure 3.11 was implemented. After a few weeks new rules (for linear multivariable system design) were added to its knowledge base *without* making any changes to the existing rules. The only requirement was that the premises of the new rules *had to be satisfied.*

The above three differences between an expert system and conventional programming techniques are reasons to prefer the use of a production rules expert system over the use of a procedural programming approach.

3.3.5 Comments

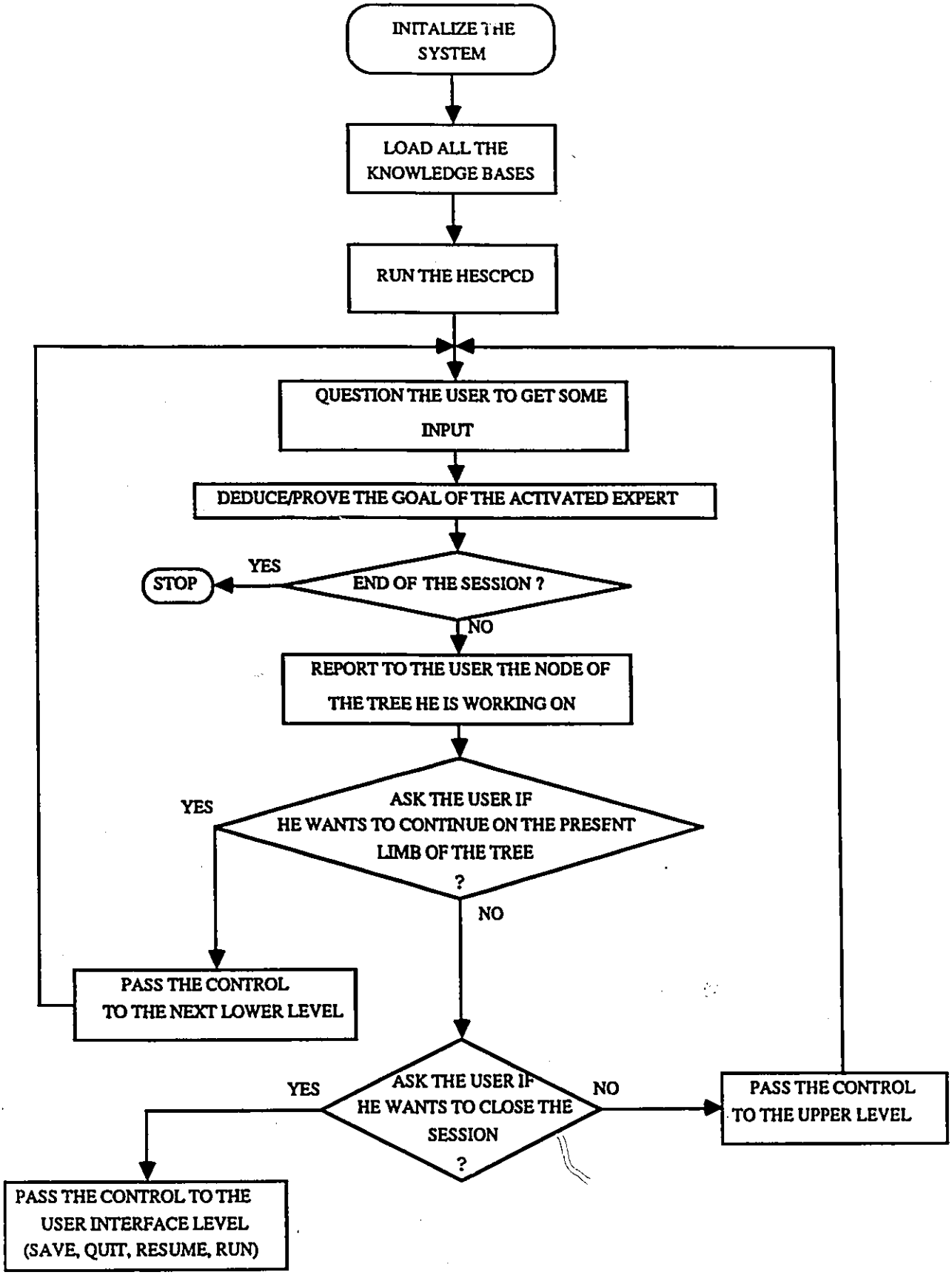
- The HESCPC expert system has been designed and implemented in such a way as to provide the user with the capability of returning to the previous level of the hierarchy if he/she has chosen a wrong limb of the tree. This has been achieved by implementing specific *meta rules* into the experts' knowledge base. Thus, before passing the control to the next lower level, the activated expert reports to the user the node of the tree he/she is working on. Next, the expert asks the user if he/she wants to continue on that limb of the tree. If the answer is "yes" the control will be passed on to the next lower level. If the answer is "no", the expert asks the user if he/she wants to close that session. If the answer is "yes" the control will be given to the *User Interface* level (the highest level in the hierarchy). If the answer is "no" the control will be passed on to the previous upper level. This is illustrated in figure 3.12.

- Although the inference engine used in the HESCPC expert system is part of Reshell, a lot of code has been written to build its knowledge base - 5349 M-Prolog lines and 519 Fortran lines - .

  At this stage of its development the HESCPC's knowledge base includes 123 *meta rules* and 261 *rules*. More details about the HESCPC's implementation can be found in its source code.

- The Reshell inference engine has proven tc be a very capable tool for developing an expert system for an engineering application.

Figure 3.12 : General Flowchart of the HESCPC System



PART 4

COMMENTS ON THE SAMPLES RUN OF THE HESCPC EXPERT SYSTEM

The HESCPC expert system has been exercised on a variety of continuous-time linear mono and multivariable systems (lumped parameters) using the *feedback state space* approach design method. It has also been used as an operator-adviser for a nuclear power plant.

The HESCPC's tests discussed below and presented in *Appendix* were chosen in such a way as to have each expert in the hierarchy activated at least once.

4.1 CONTROL SYSTEM DESIGN WITH HESCPC

This section presents a short discussion of three tests: two for *multivariable linear systems* and one for *a monovariable linear system* (refer to figures 3.8 and 3.11).

4.1.1 Multivariable Linear Sytem Design Examples

The following design example of a multivariable linear system achieved by the HESCPC expert in *Appendix - TEST 1* is provided:

Given a linear time invariant system described by the equation:

$$dx/dt = Ax + Bu, \quad \text{where:}$$

A is a $n \times n$ *matrix* called the *state space matrix*

B is a $m \times n$ *matrix* called the *input matrix*

x is the n x 1 *state space vector*

u is the m x 1 *input vector.*

It is required to design a feedback upon the state variable x

$$u = Hx + Gv, \text{ where:}$$

v is the m x 1 *reference variable*

G is a given m x m *matrix*

H is the *desired* m x n *feedback matrix.*

Solution:

Step 1.

As presented in *Appendix - TEST 1*, the user initializes the HESCPC system by issuing the "new_reshell run hescpc" command. After the *specialized problem expert,* State Space Feedback Expert, is reached the user is asked for the *input data* .

Step 2.

The State Space Feedback Expert collects the *input data* from the user:

$$A = \begin{matrix} 9.6 & -5.7 \\ 0.9 & -23.7 \end{matrix}$$ the *state space matrix*

$$B = \begin{matrix} -0.32 & -98.6 \\ 6.0 & 2.8 \end{matrix}$$ the *input matrix*

and then activates Task1_Mgr, the *specialized subproblem expert,* which in turn activates the Procedural Knowledge Block (PKB) via Task1_For . The results of these calculations:

S a matrix containing the echellon form of the composed matrix (B,A),

IR the controllable matrix rank, and

ICON the controllable index,

are displayed to the user and analysed by the Analist1 expert which concludes that *"the given linear system is multivariable and controllable"*.

Step 3.

The next *specialized subproblem* expert which is activated, Task2_Mgr, goes into the PKB block via Task2_For for solving its *subproblem* which is *compute the system's eigenvalues* . The *output data* of this *subproblem*

VALR a vector containing the Real Parts of the eigenvalues,

VALI a vector containing the Imaginary Parts of the eigenvalues, and

IER the error index of subproblem Task2_For,

is sent to the user's screen and analysed by Analyst2 expert which concludes that *" the computation was correctly completed " (see Appendix- TEST 1).*

Step 4.

As a next step, *the desired new eigenvalues* of the given linear system will be generated by the *specialized subproblem* expert Task3_Mgr (using PKB via Task3_For) after the user imposes the following parameters:

X_{min} the lower limit of the Real Part of the eigenvalues,

X_{max} the upper limit of the Real Part of the eigenvalues,

X_{min} the lower limit of the Imaginary Part of the eigenvalues,

X_{max} the upper limit of the Imaginary Part of the eigenvalues, and

P the percentage of complex eigenvalues to the total number of eigenvalues.

The result of the *subproblem* solved by Task3_Mgr expert is *the desired new eigenvalue vector* (R) of the *closed loop system* which has been correctly computed,

as the Analyst3 expert has indicated.

Step 5.

Next, the *specialized subproblem* expert, Task6_Mgr, helped by PKB block via Task6_For, using the Jordan method, computes:

M the feedback matrix,

W1 the closed loop system matrix, and

IER the error index of subproblem Task6_For,

which are displayed on the screen and analysed by the Analyst6 expert. At this point the user is asked if he/she wants to know the *new eigenvalues* of the given system. If the answer is "yes" the Task2_Mgr expert is activated again (Step3) with the *closed loop matrix* W1 as input, instead of the *state space matrix* A given at the beginning of the session. The design session continues as before (Step 4 and Step 5). As an alternative to the Jordan method, in Step 5, the user has chosen the Wolovich approach. Therefore, Task7_Mgr expert *computes the same* H and W1 as the Task6_Mgr expert but the Wolovich's method is not suitable for the given system because the *input matrix* B is not of *maximal rank* as the Analyst7 expert says.

Step 6.

After the Analyst7 expert concludes that *"computation incorrectly completed"* , the user could finish the session or try again with another system (A,B). In TEST1, the user has decided to go for another session. At this point, the State Space Feedback Expert prepares its blackboard for TEST 2 by deleting all the objects created during previous session.

Step 7.

As presented in *Appendix - TEST 2*, State Space Feedback Expert asks for the new input data of the *target problem.*

The *state space matrix* A for the new linear system is:

$$A = \begin{matrix} 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & -2 & 0 & 0 \end{matrix}$$

The *input matrix* B is:

$$B = \begin{matrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{matrix}$$

The sequence of actions for this step is similar with Step 2 but for the new input data the Analyst1 expert concludes *" the multivariable system is not controllable"*.

Step 8.

The next *specialized subproblem* expert which is activated (Task5_Mgr) checks the *system's stability* by activating the PKB block via the Task5_For component. The output data of this *subproblem* is sent to the user and analysed by the Analyst5 expert which concludes that *" computation incorrectly completed because the sign martrix A cannot be evaluated"*, so the system cannot be *stabilized.*

Step 9.

For this step, which is similar with Step 6, the State Space Feedback Expert prepares its blackboard for TEST 3.

4.1.2 Monovariable Linear System Design Example

Step 10.

Refering to *Appendix - TEST 3,* the new input data of the *target problem* is:

$$A = \begin{matrix} -5 & 0.7 \\ 2.6 & -0.2 \end{matrix}$$ the *state space matrix*

$$B = \begin{matrix} -1.8 \\ 4.3 \end{matrix}$$ the *input matrix*

The sequence of actions for this step is the same as for Step 2, but for this input data the Analyst1 expert concludes that " *the given linear system is monovariable and controllable* ".

Step 11.

At the end of this step, which is similar with Step 3, the Analyst2 expert concludes "*computation correctly completed* ".

Step 12.

This step is equivalent with Step 4.

Step 13.

This step is similar with Step 5. The only difference is that the *specialized subproblem* expert will be Task4_Mgr because the given linear system is monovariable. Therefore, the *closed loop system matrix* (W1) will be computed making the required call to the PKB block via the Task4_For. The Analyst4 expert concludes " *computation correctly completed* ". Next, the user is asked if he/she wants to know the *new eigenvalues* of the system. The answer being "yes" (see *Appendix - TEST 3*), the Task2_Mgr expert is activated again (Step 3) with the *closed loop matrix* W1 as input instead of matrix A given at Step 10. At the end of this step, the Analyst2 expert reports "*computation correctly completed* " and the user is asked if he/she wants to change the *final system eigenvalues*. Because the answer is no, the design session for state space feedback is ended and the control is given back to the *User Interface* level.

4.2 OPERATOR-ADVISER SAMPLES RUN

The following two boundary tests have been made to the Operator-Adviser

expert. For the first test, the input data represents an emergency state of the nuclear reactor. This is presented in *Appendix - TEST 4*. For the second test, the input data has been selected to represent a normal reactivity state of the reactor and is presented in *Appendix - TEST 5*.

Step 14.

Refering to *Appendix - TEST 4,* from the *User Interface* level the user issues the "run" command as in Step 1, but now the *target problem* is different. The first part of this test shows how the control is passed from one level back to the previous level in HESCPC system (the implementation of figure 3.12).

Step 15.

After going back to the previous level, the user selects a good path and the Operator-Adviser expert (*the target problem expert*) is activated. As was presented in figure 3.7, the expert loads the *Sensors Data File* into its blackboard, displays this data to the user and starts the reasoning process. The results of this process are displayed on the screen.

Step 16.

In the case of a new session, the expert loads the updated *Sensors Data File* into its blackboard and starts the reasoning process as presented in *Appendix - TEST 5*.

PART 5

CONCLUSION

This part of the thesis presents what has been achieved, as well as, what has not been achieved by this research and some future work which has to be done for further developments of the HESCPC expert system.

5.1 WHAT HAS BEEN ACHIEVED

- o A hierarchical structure has been defined, designed, and implemented using Reshell. This includes all the areas of the application of expert system technology in the field of Process Control.
- o Knowledge has been represented as a semantic network - production rules.
- o A declarative-procedural knowledge interface has been designed and implemented. This has made possible the integration of control system design rules represented by M-Prolog statements with control system design algorithms embedded in a Fortran package (CONTPACK).
- o The Fortran routines error messages are handled by M-Prolog rules in the declarative-procedural knowledge interface.This has been possible through testing the values of the indices associated with each Task*n*_For component which are returned by Fortran routines. Depending on their values a corresponding message is displayed on the user screen and a specific action is taken by the State Space Feedback Expert *(the target problem expert)* . (See Part 4 Step 5, Step 6 and Step 8).

- Control of the Fortran tasks execution has been achieved. The sequence of these tasks is established depending on the data flow and user intervention.
- A demonstration has been given on the use of an expert system approach for control system design.
- A small knowledge base of an expert system that acts as an operator companion for a nuclear power plant has been designed and implemented.

5.2 WHAT HAS NOT BEEN ACHIEVED

- The Operator-Adviser Expert is a real time expert system which considers the relevant data collected from ten sensors. It has only 2 meta rules and 57 rules, therefore a small knowledge base. In terms of performance, this expert comes up with a diagnosis and an advice in 10 to 35 seconds depending on the input data. As a result of our discussions with the Atomic Energy of Canada Ltd. representatives, a CANDU reactor has over 3000 sensors. Taking into account this number of sensors a huge knowledge base should be built for such an application. In this case, the expert system search space will be immense, therefore it is very difficult to give a guarantee that the user will receive a diagnosis and an advice which can be used in real time.
- In this implementation the Operator-Adviser expert system makes reasoning on the updated data from the sensors only. A mechanism for considering the historical data files has to be implemented to meet the requirements of a real time expert system.

5.3 FUTURE WORK

The results reported in this thesis are the first steps made at the University of Ottawa in the direction of applying expert system technology to Process Control. To continue in this direction, the following future work is proposed:

- The Procedural Knowledge Block will include a graphics package called Interactive Simulation Language (ISL) for displying a block diagram of the designed control system. ISL is a highly interactive and colorful graphics oriented language which provides the user with the capability to solve linear or very nonlinear ordinary differential equations with one independent variable.
- The CONTPACK package does not include algorithms for nonlinear systems design. As a future work, the HESCPC expert system will be integrated with the MACSYMA system. MACSYMA is a large, interactive computer system, designed to assist the user in solving mathematical problems. It has a wide range of capabilities that enables the user to apply mathematical transformations to symbolic inputs to yield either symbolic results, or Fortran programs. It also provides tools for formulating and editing new problems for creating and introducing new algorithms in the knowledge data base [29].
- Completion of the defined knowledge bases.
- Parallel execution of different experts.
- Exploring the implementation of the following mechanisms: non-monotonic reasoning, learning, and theoretic proofs.

PART 6

REFERENCES AND RELATED LITERATURE

[1] Bainbridge L.: "Analysis of Verbal Protocols from a Process Control Task", in E. Edwards, and F.P. Leeds, The Human Operator in Process Control. London: Taylor and Francis, Ltd., 1974.

[2] Bainbridge L., Beishon J., Hemming J.H., and Splaine M.: "A Sudy of Real Time Decision-making Using a Plant Simulator", in E. Edwards and F.P. Leeds, The Human Operator in Process Control, London: Taylor and Francis Ltd., 1974.

[3] Beaverstock M., Bristor E.H., Fortin D: "Expert Systems as a Stimulus to Improved Process Control. Proc. of Am. Contr. Conf., 1985 (pp. 898-903).

[4] Chester D., Lamb D., Dhurjati P.: "Rule-Based Computer Alarm Analysis in Chemical Process Plants. Proc. 7th Anual Micro-Delcon '84, pp. 22-29, IEEE Computer Society Press, May. 1984.

[5] Chesster D., Lamb D., Dhurjati P.: "An Expert System Approach to On-line Analysis in Power and Process Plants". Computer Eng., vol. 1, pp. 345-352, Aug. 1984.

[6] Cohen P.R. and Feigenbaum E.A.: "The Handbook of Artificial Intelligence, Vol. 3, William Kaufmaun, Inc., Los Altos, CA 1982.

[7] Crossman E.R.F.W.: "Automation and Skill", in E.Edwards and F.P.Lees, The human operator in process control, London: Taylor and Francis, Ltd., 1974.

[8] Fickelsherer R., Lamb D.E., Dhurjati P., and Chester D.L.: "Role of Dynamic Simulation in Developing an Expert System for Chemical Process Faults Detection". Proc. 1986 Summer Computer Simulation Conf., Society for Computer Simulation, July 28-30, 1986.

[9] Freeman D. Darrell: "Artificial Intelligence Applications in Process Control". Proc. of Am. Contr. Conf., pp. 889-891.

[10] Gidwani K. Kumar: "The Role of Artificial Intelligence Systems in Process Control". Proc. of Am. Contr. Conf., pp. 881-884.

[11] Goldberg M., Goodenough D.G., Alvo M., Karam G.: "A Hierarchical Expert System for Updating Forestry Maps with LANDSAT Data". Proc. of the IEEE, Vol. 73, No. 6, June 1985.

[12] Griesmer J.H., Hong S.J., Karnaugh M., Kastern J.K., Schor M.I., Enmnis R.L., Klein D.A., Milliken K.R., and VanWoerkom H.M.: "YES/MVS: A Continuous Real Time Expert System". Proc of the National Conference on Artif. Intell., pp. 130-136, Austin, TX, 1984.

[13] Hayes-Roth Frederick, Waterman D.A., Douglas B. Lenat: "Building Expert Systems". Addison-Wesley Publishing Company Inc., 1983.

[14] Hunt R.M." "Rule-Based Models of Human Control of Dynamic Processes". Proc. of the IEEE International Conf. on Cybernetics and Society, pp. 686-689, Seattle, WA, 1982.

[15] Ionescu Dan: "Expert Systems for Process Control". University of Iowa -Seminar in Expert Systems, December 1987.

[16] Ionescu Dan et. al: "CONTPACK - An Interactive CAD Package for Computer Process Control". Technical Report, Polyt. Inst., Timisora, Romania, 1983.

[17] Ionescu Dan, Trif Ioan, Popescu Ilie: "Simulation Expert System for Dynamical Processes". Applied Simulation and Modelling ASM"86, pp. 533-535.

[18] Ionescu D., Lethebinh Ph., Trif I.: "Expert System for Computer Process Control Design". Proceedings of the IEEE International Symposium on Intelligent Control, 1987, pp. 185-193.

[19] Ionescu D., Trif I.: "A Hierarchical Expert System for Computer Process Control". Proceedings of the Am. Contr. Conf., Atlanta, Ga., June 15 - 17/88, pp.1-7 (awarded as the best presentation).

[20] James J.R.: "Considerations Concerning the Construction of an Expert System for Control System Design". Ph.D. Thesis, Rensselaer Polytechnic Institute, 1986.

[21] Kaemmerer W.F., Christopherson P.D.: "Using Process Models with Expert Systems to Aid Process Control Operators". Proc. of Am. Contr. Conf. 1985, pp. 892-897.

[22] Karam G.M., Cardill A.M., Braun F.: "Reshell Beginer's Guide", 1987.

[23] Karam G.M., Cardill A.M., Braun F.: "Knowledge Engineering with Reshell", 1987.

[24] Karam G.M.: "Reshell Software Reference", 1987.

[25] Kraus T.W.: "Self-Tuning Control Using an Expert System Approach". Measurements and Control, June 1985.

[26] Kraus T.W. and Myron T.J.: "Self-Tuning PID Controller Uses Pattern Recognition Approach", Control Engineering June 1984.

[27] Lauber F.J.: "Software for Industrial Process Control", Computer, 17(2), February, 1984, pp. 6-8.

[28] Lee T.H., Adams G.E., and Gaines W.M.: "Computer Process Control: Modeling and Optimization". New York, Wiley, 1968.

[29] MACSYMA User's Manual, Vol. I-II

[30] McAllester, D.A.: "An Outlook on Truth Maintenance". M.I.T. AI Memo No. 551, MIT, Cambridge, MA., Aug. 1980.

[31] McDermott J.: "R1: A Rule-based Configurer of Computer Systems". Technical Rept. CMU-CS-80-119. Dept. of Computer Science, Canergie-Mellon Univ., Pittsburgh, P.A., 1980.

[32] Mokoff M.: "Artificial Intelligence Systems Make Their Mark". Computer Design, Nov. 1983, pp. 33-36.

[33] Michaelson, R.H., Michie D., and Boulanger A.: "The Technology of Expert System". Byte Vol. 10, Number 4, pp. 303-312, April, 1985.

[34] Moore R.L., Hawkinson L.B., Knickerbocker C.G., And Churchman L.M.: "A Real-time Expert System for Process Control". The First Conf. on Aritif. Intell. Applications, IEEE Computer Society, pp. 569-576, Denver, Colorado, 1984.

[35] Nelson W.R.: "REACTOR: An Expert System for Diagnosis and Treatment of Nucelar Reactor Accidents". Proc. of the National Conf. on Artif. Intell., pp. 296-301, Pittsburgh, PA, 1982.

[36] Pang G.K.H., MacFarlane A.G.J.: "An Expert Systems Approach to Computer-Aided Design of Multivariable Systems". Lecture Notes in Control and Information Sciences, Springer-Verlang Berlin, Heidelberg 1987.

[37] Parcy Jean-Pierre: "Un Système Expert En Diagnostic Sur Réacteur A'Neutrons Rapides". These Docteur 3ème Cycle, Université d'Aix-Marseille, 1982.

[38] Poter B.W., and Kibler D.F.: "Learning Operator Transformations". Proc. of the National Conf. on Artif. Intell., pp. 278-282, Austin, TX, 1984.

[39] Prerau D.S.: "Selection of Appropriate Domain for an Expert System". The AI Magazine, Summer 1985, pp. 26-30.

[40] Rowan d.A.: "Chemical Plant Fault Diagnosis Using Expert Sytems Technology: A Case Study". IFAC Kyoto Workshop on Fault Detection and Safety in Chemical Plants, Kyoto, Japan, Sept. 28-Oct. 1, 1986.

[41] Sauers R. and Walsh R.: "On the Requirements of Future Expert Systems". Proc. of the Eighth IJCAI, pp. 110-115, Karlsruhe, W. Germany, 1983.

[42] Shirley R.S. and Fortin D.A.: "Status Report: An Expert System to Aid Process Control". Proc. ISA/85, pp. 1463-1470, Oct. 21-24, 1985.

[43] Shirley R.S.: "Status Report 2: An Expert System to Aid Process Control". Proc. of the TAPPI 1986 Eng. Conf., pp. 425-430, Sept.22-25, 1986.

[44] Shirley R.S.: "Some Lessons Learned Using Expert Systems for Process Control". IEEE Control Systems Magazine, Dec. 1987, pp. 11-15.

[45] Vere S.A.: "Introduction of Relational Productions in the Presence of Background Information". Proc. of the Fifth IJCAI, pp. 349-355, Cambridge, Mass, 1977.

APPENDIX

TEST 1

```
$ new_reshell run hescpc
*** RESHELL v3.0 ***
LOADING : es_controller - 88/03/12 18:57:31


* Initializing System *
* Executing RESHELL system startup file *
* RESHELL system startup file executed *
* Executing user startup file *
* User startup file executed *
* Loading help file *
* Help file loaded *
* Loading knowledge and context for: analyst7 *
* Loading knowledge and context for: analyst6 *
* Loading knowledge and context for: analyst5 *
* Loading knowledge and context for: analyst4 *
* Loading knowledge and context for: analyst3 *
* Loading knowledge and context for: analyst2 *
* Loading knowledge and context for: analyst1 *
* Loading knowledge and context for: task7_mgr *
* Loading knowledge and context for: task6_mgr *
* Loading knowledge and context for: task5_mgr *
* Loading knowledge and context for: task4_mgr *
* Loading knowledge and context for: task3_mgr *
* Loading knowledge and context for: task2_mgr *
* Loading knowledge and context for: task1_mgr *
* Loading knowledge and context for: state_space_feedback_mgr *
* Loading knowledge and context for: bilinear_sys *
* Loading knowledge and context for: quasilinear_sys *
* Loading knowledge and context for: nonlinear_sys *
* Loading knowledge and context for: nuclear_reactor *
* Loading knowledge and context for: state_space *
* Loading knowledge and context for: stochastic4 *
* Loading knowledge and context for: stochastic3 *
* Loading knowledge and context for: stochastic2 *
* Loading knowledge and context for: stochastic1 *
* Loading knowledge and context for: deterministic4 *
* Loading knowledge and context for: deterministic3 *
* Loading knowledge and context for: deterministic2 *
* Loading knowledge and context for: deterministic1 *
* Loading knowledge and context for: linear_sys *
* Loading knowledge and context for: hescpc *
* All knowledge loaded *
* Initialized system *

*** User Interface ***
Enter one of run resume save quit exit help {help}
>run
```

```
**********************************
*   WELCOME TO THE HIERARCHICAL   *
*          EXPERT SYSTEM          *
*               FOR               *
*     COMPUTER PROCESS CONTROL    *
**********************************


*** NOTICE: This work was supported in part ***
***         by the Natural Science and      ***
***         Engineering Research of Canada  ***
***         under Grant no.A6684            ***

*** The HESCPC is a Reshell based expert    ***
*** system. Its knowledge is distributed on ***
*** an hierarchical structure among 30      ***
*** experts on different levels. At this    ***
*** stage of development 20 out of these    ***
*** experts have their knowledge bases      ***
*** completely or partially built.          ***


Which expert class would you like to activate?
Enter one of operator_companion hardware_design software_design
ctr_sys_des_alg

>ctr

What kind of control problem are you dealing with?
Enter one of linear_sys nonlinear_sys quasilinear_sys bilinear_sys

>lin


You are on the following node of the tree:

the expert_class is      *** ctr_sys_des_alg ***
and the problem_class is *** linear_sys ***


Would you like to continue on this limb of the tree?
Enter one of yes no

>y



*********************************
* START OF THE LINEAR SYSTEMS *
*            EXPERT             *
*********************************


What kind of parameters your system has ?
```

```
Enter one of distributed lumped

>lum

What class of systems would you like to select ?
Enter one of continuous discrete

>con

What type of system are you dealing with ?
Enter one of deterministic stochastic

>det


You are on the following node of the tree:

the expert_class is     *** ctr_sys_des_alg ***
the problem_class is    *** linear_sys ***
the parameter_type is   *** lumped ***
the system_class is     *** continuous ***
and the system_type is *** deterministic ***


Would you like to continue on this limb of the tree?
Enter one of yes no

>y



**********************************
*     START OF DETERMINISTIC3    *
*             EXPERT             *
**********************************


What kind of problem has to be solved?
Enter one of realization modelling_and_simulation control_laws
identification descentralization theoretic_proofs performance_analysis

>contr

What kind of design are you faced with?
Enter one of optimal_control feedback decoupling

>feed

What method do you intend to use?
Enter one of input_output state_space

>state_space
```

```
You are on the following node of the tree:

the expert_class is          *** ctr_sys_des_alg ***
the problem_class is         *** linear_sys ***
the parameter_type is        *** lumped ***
the system_class is          *** continuous ***
the system_type is           *** deterministic ***
the problem_to_be_solved is *** control_laws ***
the kind_of_design is        *** feedback ***
and the method_to_be_used    *** state_space ***


Would you like to continue on this limb of the tree?
Enter one of yes no

>y



****************************
*   START OF STATE SPACE   *
*          EXPERT          *
****************************


What kind of state space problem has to be solved?
Enter one of state_feedback output_feedback canonical_forms
property_analysis

>state_feed


You are on the following node of the tree:

the expert_class is          *** ctr_sys_des_alg ***
the problem_class is         *** linear_sys ***
the parameter_type is        *** lumped ***
the system_class is          *** continuous ***
the system_type is           *** deterministic ***
the problem_to_be_solved is *** control_laws ***
the kind_of_design is        *** feedback  ***
the method_to_be_used        *** state_space ***
kind of feedback             *** state_feedback ***


Would you like to continue on this limb of the tree?
Enter one of yes no

>y
```

```
**********************************
* START OF STATE SPACE FEEDBACK *
*            EXPERT              *
**********************************


*** The STATE SPACE FEEDBACK EXPERT is  ***
*** able to solve the following kind of ***
*** problem:                            ***

*** GIVEN a linear time invariant system***
*** described by the ecuation:          ***

***           dx/dt = Ax + Bu           ***

*** A is -----> a  n x n  matrix called ***
***             the state space matrix  ***
*** B is -----> a  m x n  matrix called ***
***             the input matrix        ***
*** x is -----> the  n x 1  state space ***
***             vector                  ***
*** u is -----> the  m x 1  input vector***

*** IT IS REQUIRED to design a feedback ***
*** upon the state space variable  x    ***

***        u = Hx + Gv,  where:         ***

*** v is -----> the  m x 1  reference   ***
***             variable                ***
*** G is -----> a given  m x m  matrix  ***
*** H is -----> the desired  m x n      ***
***             feedback matrix.        ***

*** This problem is solved by STATE     ***
*** SPACE FEEDBACK expert and its 14    ***
*** children using the declarative      ***
*** knowledge embedded in their knowledge**
*** bases along whith the procedural    ***
*** knowledge embedded into a specilized **
*** FORTRAN library called CONTPACK.    ***

*** TO THE USER: 1.  Any real number given*
***                 to this expert MUST **
***                 BE followed by E to **
***                 the desired power. ***
*** Ex.: 7.8E0, or 45.76e-3, or 0.9e5 ..***
***              2.  The maximum format ***
***                  for real number    ***
***                  accepted by the    ***
***                  FORTRAN task is F14.4*
```

```
Ready to proceed ?
Enter one of yes no

>y


What is the STATE SPACE VECTOR DIMENSION ?
Enter a value between 1 and 20

>2

What is the INPUT VECTOR DIMENSION ?
Enter a value between 1 and 20

>2

What is the STATE SPACE MATRIX ?
Enter an expression that matches [V1|V2]

.>[[9.6e0,-5.7e0],[-0.9e0,-23.7e0]].

What is the INPUT MATRIX ?
Enter an expression that matches [V1|V2]

.>[[-0.32e0,-98.6e0],[6.0e0,2.8e0]].

What is the CALCULATION PRECISION for TASK1 ?
{1.0E-5}

.>d.


************************
*  START OF TASK1_MGR  *
*        EXPERT        *
************************



*** This expert has an intimate knowledge about ***
*** TASK1.FOR. It knows how to:                  ***
***       - prepare the input data for the task ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***


*** Loading INPUT FILE for TASK1.FOR ***

*** TASK1.INPUT file loaded !!! ***

*** Activating TASK1.FOR ***
```

```
*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :      ***
*** DIRECT COMPUTATION OF CANONICAL FORMS FOR***
*** LINEAR SYSTEMS BY ELEMENTARY MATRIX       ***
*** OPERATIONS. IEEE Trans.,AC-19, April,1974***
*** pp.124-126, by I.D.APLEVICH               ***


***TASK1_EXECUTED***

*** Loading OUTPUT FILE from TASK1 ***

*** TASK1.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK1 IS : S, IR, ICON ***


*** S is  -----> a matrix containing the echellon***
***              form of the composed matrix (B,A).*

*** The value of S is -----> [[6.0E0,0,0,0],[0,0,0,0]]

*** Controllable matrix rank,IR is ---------> 2

*** Controllable index,ICON is ----------> 2



*************************
* START OF THE ANALYST1 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK1.FOR ***


*** TASK1.FOR ------> computation correctly   ***
***                   completed.              ***

*** THE MULTIVARIABLE SYSTEM IS CONTROLLABLE. ***
*** Therefore a 2 * 2 feedback matrix (H)    ***
*** could be found.                          ***


*** The STATE SPACE FEEDBACK DESIGN procedure***
*** for MULTIVARIABLE SYSTEMS will follow.   ***

Would you like to proceed ?
Enter one of yes no

>y
```

```
Do you want the SCALLING of matrix A,yes=1 ?
Enter one of 1 2

>1


What is the CALCULATION PRECISION for TASK2 ?
{5.0E-7}

.>d.


************************
*  START OF TASK2_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK2.FOR. It knows how to:                  ***
***       - create an input data file for the task**
***       - activate the task as a logical function*
***       - load the output data from the task   ***
***       - display this data to the user and    ***
***       - send this data to the parent expert. ***


*** THE SYSTEM EIGENVALUES WILL BE CALCULATED ***
*** BY TASK2.FOR                                ***

*** METHOD --> The original matrix is reduced ***
***            to the upper HESSENBERG form   ***
***            and then the QR method is applied*


*** Loading INPUT FILE for TASK2.FOR ***

*** TASK2.INPUT file loaded !!! ***

*** Activating TASK2.FOR ***


*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :      ***
*** MATRIX EIGENSYSTEM ROUTINES-EISPACK GUIDE***
*** LECTURE NOTES IN COMPUTER SCIENCE, Nr.6, ***
*** 1974, Springen Verlag, by SMITH et.       ***

***TASK2_EXECUTED***

*** Loading OUTPUT FILE from TASK2.FOR ***

*** TASK2.OUTPUT file loaded !!! ***
```

```
*** THE OUTPUT DATA FROM TASK2 IS : ***
***       VALR, VALI, IER             ***

*** The VALR is ----> a vector with 2 element(s) ***
***   containing the REAL part      ***
***   of the EIGENVALUES            ***

*** The value of VALR is ------>  [0,9.6E0]

*** The VALI is -----> a vector with 2 element(s) ***
***    containing the IMAGINARY ***
***    part  of the EIGENVALUES ***

*** The value of VALI is ------>[0,0]

*** The error index, IER is -------> 0 ***



*************************
* START OF THE ANALYST2 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK2.FOR ***


*** TASK2 ---> computation correctly completed. ***

*** The eigenvalues are not ordered; the complex***
*** conjugated eigenvalues are typed           ***
*** consecutively. The one with the positive   ***
*** imaginary being the first.                 ***


*** THE DESIRED NEW EIGENVALUES WILL BE ***
*** AUTOMATICALLY GENERATED BY TASK3.FOR***

*** THE USER MUST IMPOSE THE FOLLOWING  ***
*** PARAMETERS : XMIN,XMAX,YMIN,YMAX,P  ***


*** XMIN is -------> the lower limit of ***
***                 the REAL part of   ***
***                 the eigenvalues.   ***

*** XMAX is -------> the upper limit of ***
***                 the REAL part of   ***
***                 the eigenvalues.   ***

*** YMIN is -------> the lower limit of ***
***                 the IMAGINARY part ***
***                 of the eigenvalues.***
```

```
*** YMAX is -------> the upper limit of ***
***                 the IMAGINARY part ***
***                 of the eigenvalues.***

*** P is ----------> a real subunitary  ***
***                 number specifying  ***
***                 the percentage of  ***
***                 complex eigenvalues***
***                 of the total number***
***                 of eigenvalues.    ***

What is the LOWER LIMIT OF REAL PART of the eigenvalues ?

.>-9.8e0.

What is the UPPER LIMIT OF REAL PART of the eigenvalues ?

.>-0.1e0.

What is the LOWER LIMIT OF IMAG. PART of the eigenvalues ?

.>-1.0e0.

What is the UPPER LIMIT OF IMAG. PART of the eigenvalues ?

.>1.0e0.

What is the PERCENTAGE OF COMPLEX EIGENVALUES ?

.>0.0e0.


************************
*  START OF TASK3_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK3.FOR. It knows how to:                  ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***

*** Loading INPUT FILE for TASK3.FOR ***

*** TASK3.INPUT file loaded !!! ***

*** Activating TASK3 ***

***TASK3_EXECUTED***
```

```
*** Loading OUTPUT FILE from TASK3.FOR ***

*** TASK3.OUTPUT file loaded !!! ***

***        THE OUTPUT DATA FROM TASK3 IS : R          ***

*** R is ------> the desired new eigenvalue vector ***
***               of the closed loop system.         ***

*** The value of R is ------->[[-9.6872E0,0],[-9.6872E0,0]]


*************************
* START OF THE ANALYST3 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK3.FOR ***


*** TASK3 ---> computation correctly completed. ***

*** THE FEEDBACK MATRIX (H) WILL BE ***
*** CALCULATED BY TASK4.FOR IF THE  ***
*** SYSTEM IS MONOVARIABLE and BY   ***
*** TASK6.FOR or TASK7.FOR IF THE   ***
*** SYSTEM IS MULTIVARIABLE.        ***

*** MATRIX (H)-----> allocates the  ***
***                  poles (R) to   ***
***                  the complete   ***
***                  controllable   ***
***                  system (A,B).  ***

*** METHOD ----> The state space    ***
***              matrix (A) is      ***
***              reduced to JORDAN  ***
***              DIAGONAL FORM.     ***

*** TECHNIQUE --> The multivariable ***
***               system is reduced ***
***               to a monovariable ***
***               one. Further the  ***
***               monovariable system**
***               is reduced to the ***
***               JORDAN DIAGONAL   ***
***               FORM using one of ***
***               two methodes :    ***
*** METHODS ----> JORDAN DIAGONAL   ***
***              FORM or WOLOVICH. ***
```

```
What METHOD would you like to use ?
Enter one of jordan wolovich

>jordan


What is the CALCULATION PRECISION for TASK6 ?
{1.0E-6}

.>d.



************************
*  START OF TASK6_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK6.FOR. It knows how to:                  ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the  user and  ***
***       - send this data to the parent expert.***

*** Loading INPUT FILE for TASK6.FOR ***

*** TASK6.INPUT file loaded !!! ***

*** Activating TASK6.FOR ***

*** The procedural knowledge embedded in this **
*** FORTRAN task is based on : MODAL CONTROL ***
*** THEORY AND APPLICATIONS, Taylor and      ***
*** Francis, London, 1972, by POTER, CROSSLEY***

***TASK6_EXECUTED***

*** Loading OUTPUT FILE from TASK6.FOR

*** TASK6.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK6 IS : H, IER, W1 ***


*** H is -----> the feedback matrix            ***

*** The value of H is ------> [[-1.211554E2,0],[-1.031553E2,0]]

*** W1 is -----> the closed loop system matrix ***

*** The value of W1 is -----> [[-2.91697E1,0],[7.260323E2,0]]
```

```
*** IER is ------> the error index of TASK6     ***

*** The value of IER is ------> 0


*************************
* START OF THE ANALYST6 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK6.FOR ***


*** TASK6 ---> computation correctly completed. ***


*** The NEW EIGENVALUES will be calculated ***
*** following the previously used procedure***
*** by activating TASK2.FOR with matrix W1 ***
*** obtained from :TASK4.FOR for monovariable*
*** systems or TASK6.FOR or TASK7.FOR for  ***
*** multivariable systems, instead of      ***
*** state space matrix A given by the user ***
*** at the begining of this design session.***


Would you like to know the NEW EIGENVALUES of the system ?
Enter one of yes no

>y


*************************
*  START OF TASK2_MGR   *
*        EXPERT         *
*************************


*** This expert has an intimate knowledge about ***
*** TASK2.FOR. It knows how to:                  ***
***       - create an input data file for the task**
***       - activate the task as a logical function*
***       - load the output data from the task   ***
***       - display this data to the user and    ***
***       - send this data to the parent expert. ***


*** THE SYSTEM EIGENVALUES WILL BE CALCULATED ***
*** BY TASK2.FOR                                ***

*** METHOD --> The original matrix is reduced ***
***            to the upper HESSENBERG form   ***
***            and then the QR method is applied*
```

```
*** Loading new INPUT FILE for TASK2.FOR ***

*** TASK2.INPUT file loaded !!! ***

*** Activating TASK2.FOR ***


*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :      ***
*** MATRIX EIGENSYSTEM ROUTINES-EISPACK GUIDE***
*** LECTURE NOTES IN COMPUTER SCIENCE, Nr.6, ***
*** 1974, Springer Verlang, by SMITH et.      ***

***TASK2_EXECUTED***

*** Loading OUTPUT FILE from TASK2.FOR ***

*** TASK2.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK2 IS : ***
***       VALR, VALI, IER            ***

*** The VALR is -----> a vector with 2 element(s) ***
***   containing the REAL parts     ***
***   of the EIGENVALUES            ***

*** The value of VALR is ------>  [0,-2.917E0]

*** The VALI is -----> a vector with 2 element(s) ***
***    containing the IMAGINARY ***
***    parts of the EIGENVALUES ***

*** The value of VALI is ------>[0,0]

*** The error index, IER is -------> 0 ***



*************************
* START OF THE ANALYST2 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK2.FOR ***


*** TASK2 ---> computation correctly completed. ***

*** The eigenvalues are not ordered; the complex***
*** conjugated eigenvalues are typed            ***
*** consecutively. The one with the positive    ***
*** imaginary being the first.                  ***
```

```
Would you like to change the FINAL SYSTEM EIGANVALUES ?
Enter one of yes no

>y


What is the LOWER LIMIT OF REAL PART of the eigenvalues ?

.>-2.3e0.

What is the UPPER LIMIT OF REAL PART of the eigenvalues ?

.>-0.7e0.

What is the LOWER LIMIT OF IMAG. PART of the eigenvalues ?

.>-0.8e0.

What is the UPPER LIMIT OF IMAG. PART of the eigenvalues ?

.>1.5e0.

What is the PERCENTAGE OF COMPLEX EIGENVALUES ?

.>0.0e0.


************************
*  START OF TASK3_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK3.FOR. It knows how to:                  ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***

*** Loading INPUT FILE for TASK3.FOR ***

*** TASK3.INPUT file loaded !!! ***

*** Activating TASK3 ***

***TASK3_EXECUTED***

*** Loading OUTPUT FILE from TASK3.FOR ***

*** TASK3.OUTPUT file loaded !!! ***
```

```
***       THE OUTPUT DATA FROM TASK3 IS : R          ***

*** R is ------> the desired new eigenvalue vector ***
***               of the closed loop system.         ***

*** The value of R is ------->[[-2.2814E0,0],[-2.2814E0,0]]


*************************
* START OF THE ANALYST3 *
*         EXPERT        *
*************************


*** Analyzing output data from TASK3.FOR ***


*** TASK3 ---> computation correctly completed. ***


*** THE FEEDBACK MATRIX (H) WILL BE ***
*** CALCULATED BY TASK4.FOR IF THE  ***
*** SYSTEM IS MONOVARIABLE and BY   ***
*** TASK6.FOR or TASK7.FOR IF THE   ***
*** SYSTEM IS MULTIVARIABLE.        ***

*** MATRIX (H)-----> allocates the  ***
***                  poles (R) to   ***
***                  the complete   ***
***                  controllable   ***
***                  system (A,B).  ***

*** METHOD ----> The state space    ***
***              matrix (A) is      ***
***              reduced to JORDAN  ***
***              DIAGONAL FORM.     ***

*** TECHNIQUE --> The multivariable ***
***               system is reduced ***
***               to a monovariable ***
***               one. Further the  ***
***               monovariable system**
***               is reduced to the ***
***               JORDAN DIAGONAL   ***
***               FORM using one of ***
***               two methodes :    ***
*** METHODS ----> JORDAN DIAGONAL   ***
***              FORM or WOLOVICH. ***


What METHOD would you like to use ?
Enter one of jordan wolovich

>wolovich
```

```
*************************
*  START OF TASK7_MGR   *
*        EXPERT         *
*************************


*** This expert has an intimate knowledge about ***
*** TASK7.FOR. It knows how to:                  ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***

*** Loading INPUT FILE for TASK7.FOR ***

*** TASK7.INPUT file loaded !!! ***


*** Activating TASK7.FOR ***


*** The procedural knowledge embedded in this **
*** FORTRAN task is based on :  LINEAR       ***
*** MULTIVARIABLE SYSTEMS, Springer Verlang, ***
*** New York/1974, by WOLOVICH  W.A.         ***


*** Loading OUTPUT FILE from TASK7.FOR ***

*** TASK7.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK7 IS : H, KR, IER, W1 ***


*** H is -----> the feedback matrix             ***

*** The value of H is ------> [[0,0],[0,0]]

*** KR is ------> controllable matrix rank ***

*** The value of KR is -------> 2

*** IER is -------> error index of TASK7       ***

*** The value of IER is ------> 1

*** W1 is -------> the closed loop matrix      ***

*** The value of W1 is ------> [[-3.272E-1,0],[-3.125E0,0]]
```

```
*************************
* START OF THE ANALYST7 *
*        EXPERT         *
*************************

*** Analyzing output data from TASK7.FOR ***

*** TASK7.FOR ---> computation incorrectly completed.***

*** The matrix B is not of MAXIMAL RANK.              ***


Would you like to TRY AGAIN with another system (A,B) ?
Enter one of yes no

>y

***    Please DO NOT PANIC !!!.   ***
*** I am preparing the blackboard ***
*** for the next design session.  ***
```

```
          TEST 2


 Ready to proceed ?
 Enter one of yes no

 >y


 What is the STATE SPACE VECTOR DIMENSION ?
 Enter a value between 1 and 20

 >4

 What is the INPUT VECTOR DIMENSION ?
 Enter a value between 1 and 20

 >2

 What is the STATE SPACE MATRIX ?
 Enter an expression that matches [V1|V2]

 .>[[0.0e0,1.0e0,0.0e0,0.0e0],[3.0e0,0.0e0,0.0e0,2.0e0],
 [0.0e0,0.0e0,0.0e0,1.0e0],[0.0e0,-2.0e0,0.0e0,0.0e0]].

 What is the INPUT MATRIX ?
 Enter an expression that matches [V1|V2]

 .>[[0.0e0,0.0e0],[1.0e0,0.0e0],[0.0e0,0.0e0],[0.0e0,1.0e0]].


 *************************
 *  START OF TASK1_MGR   *
 *        EXPERT         *
 *************************


 *** This expert has an intimate knowledge about ***
 *** TASK1.FOR. It knows how to:                  ***
 ***       - prepare the input data for the task ***
 ***       - create an input data file for the task*
 ***       - activate the task as a logical function
 ***       - load the output data from the task  ***
 ***       - display this data to the user and   ***
 ***       - send this data to the parent expert.***


 *** Loading INPUT FILE for TASK1.FOR ***

 *** TASK1.INPUT file loaded !!! ***

 *** Activating TASK1.FOR ***
```

```
*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :     ***
*** DIRECT COMPUTATION OF CANONICAL FORMS FOR***
*** LINEAR SYSTEMS BY ELEMENTARY MATRIX      ***
*** OPERATIONS. IEEE Trans.,AC-19, April,1974***
*** pp.124-126, by I.D.APLEVICH              ***

***TASK1_EXECUTED***


*** Loading OUTPUT FILE from TASK1 ***

*** TASK1.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK1 IS : S, IR, ICON ***


*** S is  -----> a matrix containing the echellon***
***              form of the composed matrix (B,A).*

*** The value of S is -----> [[1.0E0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0
,0,0,0,0]]

*** Controllable matrix rank,IR is ---------> 1

*** Controllable index,ICON is ----------> 1



*************************
* START OF THE ANALYST1 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK1.FOR ***


*** TASK1.FOR ----> computation correctly ***
***                 completed.            ***

*** THE MULTIVARIABLE SYSTEM IS NOT ***
*** COMPLETE CONTROLLABLE. Perhaps  ***
*** it could be stabilizable.       ***

Would you like to check the stability ?
Enter one of yes no

>y

What is the ITERRATION NUMBER for TASK5 ?
Enter one of 30 50

>30
```

```
************************
*  START OF TASK5_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK5.FOR. It knows how to:                  ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***


*** THE METHOD FOR DETERMINING THE STABILITY ***
***         OF THE SYSTEM (A,B) FOLLOWS:      ***


*** METHOD -----> This method is based on the***
***               SIGN MATRIX of matrix A.    ***
***               The matrix A is reduced to ***
***               the JORDAN FORM and the     ***
***               stability of the            ***
***               uncontrollable part is      ***
***               tested. This job is         ***
***               performed by TASK5.         ***

*** Loading INPUT FILE for TASK5.FOR ***

*** TASK5.INPUT file loaded !!! ***


*** Activating TASK5.FOR ***


*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :      ***
*** DECOMPOSITION AND REDUCTION OF LINEAR     ***
*** SYSTEMS BY THE MATRIX SING FUNCTION.      ***
*** Revue Roumaine des Sciences Techniques,   ***
*** Serie EE, Vol.21, Nr.4/1976, by C. POPEEA***
*** and L.LUPAS.                              ***


***TASK5_EXECUTED***

*** Loading OUTPUT FILE from TASK5.FOR ***

*** TASK5.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK5 IS : W2, IER  ***


*** W2 is ------> a matrix containing the    ***
***               independent rows of the    ***
```

```
***                transposed contrllable ***
***                matrix.                ***

*** The value of W2 is ------> [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0
]]

*** IER is ----> the error index of TASK5     ***

*** The value of IER is ---------> 2


************************
* START OF THE ANALYST5 *
*        EXPERT         *
************************


*** Analyzing output data from TASK5.FOR ***


*** TASK5 --> computation incorrectly completed***

*** The METHOD used could not be completed,     ***
*** because the sign matrix SIGN(A) cannot be  ***
*** evaluated.                                  ***


Would you like to TRY AGAIN with another system (A,B) ?
Enter one of yes no

>y


***    Please DO NOT PANIC !!!.    ***
*** I am preparing the blackboard ***
*** for the next design session.  ***
```

```
          TEST 3


Ready to proceed ?
Enter one of yes no

>y

What is the STATE SPACE VECTOR DIMENSION ?
Enter a value between 1 and 20

>2

What is the INPUT VECTOR DIMENSION ?
Enter a value between 1 and 20

>1

What is the STATE SPACE MATRIX ?
Enter an expression that matches [V1|V2]

.>[[-5.0e0,0.7e0],[2.6e0,-0.2e0]].

What is the INPUT MATRIX ?
Enter an expression that matches [V1|V2]

.>[[-1.8e0],[4.3e0]].


************************
*  START OF TASK1_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK1.FOR. It knows how to:                  ***
***       - prepare the input data for the task ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***


*** Loading INPUT FILE for TASK1.FOR ***

*** TASK1.INPUT file loaded !!! ***

*** Activating TASK1.FOR ***


*** The procedural knowledge embedded in this **
```

```
*** FORTRAN task is based on the paper :      ***
*** DIRECT COMPUTATION OF CANONICAL FORMS FOR***
*** LINEAR SYSTEMS BY ELEMENTARY MATRIX       ***
*** OPERATIONS. IEEE Trans.,AC-19, April,1974***
*** pp.124-126, by I.D.APLEVICH               ***


***TASK1_EXECUTED***


*** Loading OUTPUT FILE from TASK1 ***

*** TASK1.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK1 IS : S, IR, ICON ***


*** S is  -----> a matrix containing the echellon***
***              form of the composed matrix (B,A).*

*** The value of S is -----> [[4.3E0,0,0],[0,0,0]]

*** Controllable matrix rank,IR is ---------> 2

*** Controllable index,ICON is ----------> 2



*************************
* START OF THE ANALYST1 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK1.FOR ***


*** TASK1.FOR ------> computation correctly  ***
***                   completed .            ***

*** THE MONOVARIABLE SYSTEM IS CONTROLLABLE. ***
*** Therefore a 1 * 2 feedback matrix (H)     ***
***     could be found .                    ***

*** The STATE SPACE FEEDBACK DESIGN procedure *
*** for MONOVARIABLE SYSTEMS will follow.   ***


Would you like to proceed ?
Enter one of yes no

>y
```

```
Do you want the SCALLING of matrix A,yes=1 ?
Enter one of 1 2

>1


************************
*  START OF TASK2_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK2.FOR. It knows how to:                  ***
***       - create an input data file for the task**
***       - activate the task as a logical function*
***       - load the output data from the task   ***
***       - display this data to the user and    ***
***       - send this data to the parent expert. ***


*** THE SYSTEM EIGENVALUES WILL BE CALCULATED ***
*** BY TASK2.FOR                                ***

*** METHOD --> The original matrix is reduced ***
***            to the upper HESSENBERG form   ***
***            and then the QR method is applied*


*** Loading INPUT FILE for TASK2.FOR ***

*** TASK2.INPUT file loaded !!! ***

*** Activating TASK2.FOR ***


*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :      ***
*** MATRIX EIGENSYSTEM ROUTINES-EISPACK GUIDE***
*** LECTURE NOTES IN COMPUTER SCIENCE, Nr.6, ***
*** 1974, Springen Verlag, by SMITH et.       ***

***TASK2_EXECUTED***

*** Loading OUTPUT FILE from TASK2.FOR ***

*** TASK2.OUTPUT file loaded !!! ***



*** THE OUTPUT DATA FROM TASK2 IS : ***
***       VALR, VALI, IER            ***

*** The VALR is ----> a vector with 2 element(s) ***
***   containing the REAL parts     ***
***   of the EIGENVALUES            ***
```

```
*** The value of VALR is ------>  [0,-5.0E0]

*** The VALI is -----> a vector with 2 element(s) ***
***    containing the IMAGINARY ***
***    parts of the EIGENVALUES ***

*** The value of VALI is ------>[0,0]

*** The error index, IER is -------> 0 ***



*************************
* START OF THE ANALYST2 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK2.FOR ***


*** TASK2 ---> computation correctly completed. ***

*** The eigenvalues are not ordered; the complex***
*** conjugated eigenvalues are typed            ***
*** consecutively. The one with the positive    ***
*** imaginary being the first.                  ***


*** THE DESIRED NEW EIGENVALUES WILL BE ***
*** AUTOMATICALLY GENERATED BY TASK3.FOR***

*** THE USER MUST IMPOSE THE FOLLOWING  ***
*** PARAMETERS : XMIN,XMAX,YMIN,YMAX,P  ***


*** XMIN is -------> the lower limit of ***
***                 the REAL part of   ***
***                 the eigenvalues.   ***

*** XMAX is -------> the upper limit of ***
***                 the REAL part of   ***
***                 the eigenvalues.   ***

*** YMIN is -------> the lower limit of ***
***                 the IMAGINARY part ***
***                 of the eigenvalues.***

*** YMAX is -------> the upper limit of ***
***                 the IMAGINARY part ***
***                 of the eigenvalues.***

*** P is ----------> a real subunitary  ***
***                 number specifying  ***
***                 the percentage of  ***
***                 complex eigenvalues***
***                 of the total number***
***                 of eigenvalues.    ***
```

```
What is the LOWER LIMIT OF REAL PART of the eigenvalues ?
.>-3.0e0.
What is the UPPER LIMIT OF REAL PART of the eigenvalues ?
.>-0.5e0.
What is the LOWER LIMIT OF IMAG. PART of the eigenvalues ?
.>-2.0e0.
What is the UPPER LIMIT OF IMAG. PART of the eigenvalues ?
.>2.0e0.
What is the PERCENTAGE OF COMPLEX EIGENVALUES ?
.>0.0e0.

************************
*  START OF TASK3_MGR  *
*        EXPERT        *
************************

*** This expert has an intimate knowledge about ***
*** TASK3.FOR. It knows how to:                  ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***

*** Loading INPUT FILE for TASK3.FOR ***

*** TASK3.INPUT file loaded !!! ***

*** Activating TASK3 ***

***TASK3_EXECUTED***

*** Loading OUTPUT FILE from TASK3.FOR ***

*** TASK3.OUTPUT file loaded !!! ***

***       THE OUTPUT DATA FROM TASK3 IS : R        ***

*** R is -------> the desired new eigenvalue vector ***
***               of the closed loop system.        ***

*** The value of R is ------->[[-2.9709E0,0],[-2.9709E0,0]]
```

```
************************
* START OF THE ANALYST3 *
*        EXPERT         *
************************

*** Analyzing output data from TASK3.FOR ***

*** TASK3 ---> computation correctly completed. ***

*** THE FEEDBACK MATRIX (H) WILL BE ***
*** CALCULATED BY TASK4.FOR IF THE  ***
*** SYSTEM IS MONOVARIABLE and BY   ***
*** TASK6.FOR or TASK7.FOR IF THE   ***
*** SYSTEM IS MULTIVARIABLE.        ***

*** MATRIX (H)-----> allocates the  ***
***                  poles (R) to   ***
***                  the complete   ***
***                  controllable   ***
***                  system (A,B).  ***

*** METHOD ----> The state space    ***
***              matrix (A) is      ***
***              reduced to JORDAN  ***
***              DIAGONAL FORM.     ***

What is the CALCULATION PRECISION for TASK4 ?
{1.0E-6}

.>d.


************************
*  START OF TASK4_MGR  *
*        EXPERT        *
************************

*** This expert has an intimate knowledge about ***
*** TASK4.FOR. It knows how to:                 ***
***       - create an input data file for the task*
***       - activate the task as a logical function
***       - load the output data from the task  ***
***       - display this data to the user and   ***
***       - send this data to the parent expert.***

*** Loading INPUT FILE for TASK4.FOR ***

*** TASK4.INPUT file loaded !!! ***

*** Activating TASK4 ***
```

```
*** The procedural knowledge embedded in this **
*** FORTRAN task is based on : MODAL CONTROL ***
*** THEORY AND APPLICATIONS, Taylor and      ***
*** Francis, London/1972,by POTER, CROSSLEY. ***


***TASK4_EXECUTED***

*** Loading OUTPUT FILE from TASK4.FOR ***

*** TASK4.OUTPUT file loaded !!! ***


*** THE OUTPUT DATA FROM TASK4 IS : H, IRE, W1 ***


*** H is ----> the state space feedback matrix ***

*** The value of H is ------> [5.025E-1,4.294E-1]

*** IER is -------> the error index from task4 ***

*** The value of IER is ------> 0

*** W1 is -----> the closed loop system matrix ***

*** The value of W1 is -------> [[-4.0954E0,0],[4.391E-1,0]]


*************************
* START OF THE ANALYST4 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK4.FOR ***


*** TASK4 ---> computation correctly completed. ***


*** The NEW EIGENVALUES will be calculated ***
*** following the previously used procedure***
*** by activating TASK2.FOR with matrix W1 ***
*** obtained from :TASK4.FOR for monovariable*
*** systems or TASK6.FOR or TASK7.FOR for  ***
*** multivariable systems, instead of      ***
*** state space matrix A given by the user ***
*** at the begining of this design session.***


Would you like to know the NEW EIGENVALUES of the system ?
Enter one of yes no

>y
```

```
************************
*  START OF TASK2_MGR  *
*        EXPERT        *
************************


*** This expert has an intimate knowledge about ***
*** TASK2.FOR. It knows how to:                  ***
***       - create an input data file for the task**
***       - activate the task as a logical function*
***       - load the output data from the task   ***
***       - display this data to the user and    ***
***       - send this data to the parent expert. ***


*** THE SYSTEM EIGENVALUES WILL BE CALCULATED ***
*** BY TASK2.FOR                                ***

*** METHOD --> The original matrix is reduced ***
***            to the upper HESSENBERG form   ***
***            and then the QR method is applied*


*** Loading new INPUT FILE for TASK2.FOR ***

*** TASK2.INPUT file loaded !!! ***

*** Activating TASK2.FOR ***


*** The procedural knowledge embedded in this **
*** FORTRAN task is based on the paper :     ***
*** MATRIX EIGENSYSTEM ROUTINES-EISPACK GUIDE***
*** LECTURE NOTES IN COMPUTER SCIENCE, Nr.6, ***
*** 1974, Springer Verlang, by SMITH et.     ***

***TASK2_EXECUTED***

*** Loading OUTPUT FILE from TASK2.FOR ***

*** TASK2.OUTPUT file loaded !!! ***



*** THE OUTPUT DATA FROM TASK2 IS : ***
***       VALR, VALI, IER           ***

*** The VALR is ----> a vector with 2 element(s) ***
***   containing the REAL parts    ***
***   of the EIGENVALUES           ***

*** The value of VALR is ------>  [0,-4.0954E0]

*** The VALI is -----> a vector with 2 element(s) ***
***    containing the IMAGINARY ***
***    parts of the EIGENVALUES ***
```

```
*** The value of VALI is ------>[0,0]

*** The error index, IER is -------> 0 ***


*************************
* START OF THE ANALYST2 *
*        EXPERT         *
*************************


*** Analyzing output data from TASK2.FOR ***


*** TASK2 ---> computation correctly completed. ***

*** The eigenvalues are not ordered; the complex***
*** conjugated eigenvalues are typed             ***
*** consecutively. The one with the positive     ***
*** imaginary being the first.                   ***


Would you like to change the FINAL SYSTEM EIGANVALUES ?
Enter one of yes no

>n


*** DESIGN SESSION FOR STATE FEEDBACK ENDED ***
```

TEST 4

```
*** User Interface ***
Enter one of run resume save quit exit help {help}

>run

************************************
*    WELCOME TO THE HIERARCHICAL   *
*          EXPERT SYSTEM           *
*                FOR               *
*     COMPUTER PROCESS CONTROL     *
************************************

*** NOTICE: This work was supported in part ***
***         by the Natural Science and      ***
***         Engineering Research of Canada  ***
***         under Grant no.A6684            ***

*** The HESCPC is a Reshell based expert    ***
*** system. Its knowledge is distributed on ***
*** an hierarchical structure among 30      ***
*** experts on different levels. At this    ***
*** stage of development 20 out of these    ***
*** experts have their knowledge bases      ***
*** completely or partially built.          ***

Which expert class would you like to activate?
Enter one of operator_companion hardware_design software_design
ctr_sys_des_alg

>ctr

What kind of control problem are you dealing with?
Enter one of linear_sys nonlinear_sys quasilinear_sys bilinear_sys

>lin

You are on the following node of the tree:

the expert_class is      *** ctr_sys_des_alg ***
and the problem_class is *** linear_sys ***

Would you like to continue on this limb of the tree?
Enter one of yes no
```

```
>n

Would you like to close this session?
Enter one of yes no

>n


*** Because you do not want either to ***
*** close this session or to continue ***
*** on this limb of the tree, I will  ***
*** transfer the control to the previous*
***       ***  Upper Level ***        ***


Which expert class would you like to activate?
Enter one of operator_companion hardware_design software_design
ctr_sys_des_alg

>oper

What kind of problem are you dealing with?
Enter one of nuclear_reactor

>nucl


You are on the following node of the tree:

the expert_class is       *** operator_companion ***
and the problem_class is *** nuclear_reactor ***


Would you like to continue on this limb of the tree?
Enter one of yes no

>y


****************************************
*    START OF THE OPERATOR-ADVISER     *
*           EXPERT SYSTEM              *
*       FOR A NUCLEAR POWER PLANT      *
****************************************


*** Loading DATA from sensors ***

*** DATA FILE loaded !!! ***


***       DISPLAYING DATA FROM THE SENSORS       ***
```

```
*** REACTIVITY_METRE sensor ----------------> is  ***   40    ***
*** REACTIVITY_BALANCE sensor --------------> is  ***   50    ***
*** DRIFT_SPEED sensor ---------------------> is  ***   34    ***
*** TEMPERATURE_1 sensor -------------------> is  ***   0    ***
*** TEMPERATURE_2 sensor -------------------> is  ***   0    ***
*** FUEL_BAR_VARIATION sensor --------------> is  ***   22    ***
*** PRIMARY_FLOW_DRIFT sensor --------------> is  ***   30    ***
*** SECONDARY_FLOW_DRIFT sensor ------------> is  ***   17    ***
*** STEAM_GENERATOR sensor -----------------> is  ***   57    ***
*** GAIN_BREAKING sensor -------------------> is  ***   49    ***

*** MAKING REASONING ON DATA FROM THE SENSORS ***


*****************************************
*  THE NUCLEAR REACTOR IS IN:           *
*  STATE ----> INTERNAL_BREAKDOWN       *
*  CONDITION -------> ACCETABLE         *
*                     FOR 2 MINUTES     *
*  CAUSE ----> SENSOR_TRTC_1 OR TRTC_2  *
*              IS NOT WORKING           *
*  ACTION ----> CHECK WHICH SENSOR IS   *
*               NOT WORKING AND TURN ON *
*               THE BACKUP TEMPERATURE  *
*                  SENSOR               *
*****************************************

*****************************************
*  THE NUCLEAR REACTOR IS IN:           *
*  STATE ------> INTERNAL_BREAKDOWN     *
*  CONDITION -------> ACCEPTABLE        *
*                     FOR 5 MINUTES     *
*  CAUSE -----> FUEL_BUNDLE_POSITONING  *
*               IS BAD                  *
*  ACTION ----> ACTIVATE ROBOT_1 FOR    *
*               POSITIONING CORRECTION  *
*****************************************

*****************************************
*  THE NUCLEAR REACTOR IS IN:           *
*  STATE -------> INTERNAL_BREAKDOWN    *
*  CONDITION ---------> ACCEPTABLE      *
*                       FOR 10 MINUTES  *
*  CAUSE -----> PRIMARY FLOW DRIFT      *
*               VARIATION ABNORMAL OR   *
*               SECONDARY FLOW DRIFT    *
*               VARIATION ABNORMAL      *
*  ACTION -----> CHECK WHICH ONE IS     *
*                ABNORMAL AND :         *
*      IF    PRIMARY FLOW DRIFT IS IN   *
*                           TROUBLE     *
*      THEN  ADJUST  PARAMETERS ......  *
*      IF    SECONDARY FLOW DRIFT IS IN *
*                           TROUBLE     *
```

```
*       THEN  ADJUST  PARAMETERS ......  *
*       IF    BOTH OF THEM ARE IN TROUBLE*
*       THEN  ADJUST  PARAMETERS ......  *
*****************************************

*****************************************
*  THE NUCLEAR REACTOR IS IN:           *
*  STATE --------> INTERNAL BREAKDOWN   *
*  CONDITION --------> EMERGENCY !!!    *
*  CAUSE --->TEMPERATURE CONTROL SYSTEM *
*            FAILURE                    *
*  ACTION ----------> SHUT DOWN THE     *
*           NUCLEAR REACTOR IMMEDIATLY! *
*****************************************

*****************************************
*  THE NUCLEAR REACTOR IS IN:           *
*  STATE --------> EXTERNAL BREAKDOWN   *
*  CONDITION ---------> ACCEPTABLE      *
*                       FOR 3 MINUTES   *
*  CAUSE -----> STEAM GENERATOR IS IN   *
*               DECOMPRESSION CONDITION *
*  ACTION -----> ACTIVATE ROBOT 2 TO    *
*    ADJUST THE FOLLOWING PARAMETERS... *
*****************************************


  Would you like another session ?
Enter one of yes no

>y
```

TEST 5

```
****************************************
*    START OF THE OPERATOR-ADVISER     *
*           EXPERT SYSTEM              *
*       FOR A NUCLEAR POWER PLANT      *
****************************************

*** Loading DATA from sensors ***

*** DATA FILE loaded !!! ***

***        DISPLAYING DATA FROM THE SENSORS        ***

*** REACTIVITY_METRE sensor ----------------> is  ***   6     ***
*** REACTIVITY_BALANCE sensor --------------> is  ***   8     ***
*** DRIFT_SPEED sensor ---------------------> is  ***   25     ***
*** TEMPERATURE_1 sensor -------------------> is  ***   7     ***
*** TEMPERATURE_2 sensor -------------------> is  ***   5     ***
*** FUEL_BAR_VARIATION sensor --------------> is  ***   5     ***
*** PRIMARY_FLOW_DRIFT sensor --------------> is  ***   3     ***
*** SECONDARY_FLOW_DRIFT sensor ------------> is  ***   7     ***
*** STEAM_GENERATOR sensor -----------------> is  ***   85     ***
*** GAIN_BREAKING sensor -------------------> is  ***   9     ***

*** MAKING REASONING ON DATA FROM THE SENSORS ***


****************************************
*  THE NUCLEAR REACTOR IS IN:          *
*  STATE -------> REACTIVITY_STATE     *
*  CONDITION ----------> ACCEPTABLE    *
*  ACTION ------> NO ACTION            *
*                 HAS TO BE CARRIED OUT *
****************************************


  Would you like another session ?
Enter one of yes no

>n

*** User Interface ***
Enter one of run resume save quit exit help {help}
```

```
>q

Are you sure?
Enter one of yes no {yes}

>y

$
```