An Overview of Realtime Expert Systems

Diaz, A.

National Research Council Canada
Conseil national de recherches Canada

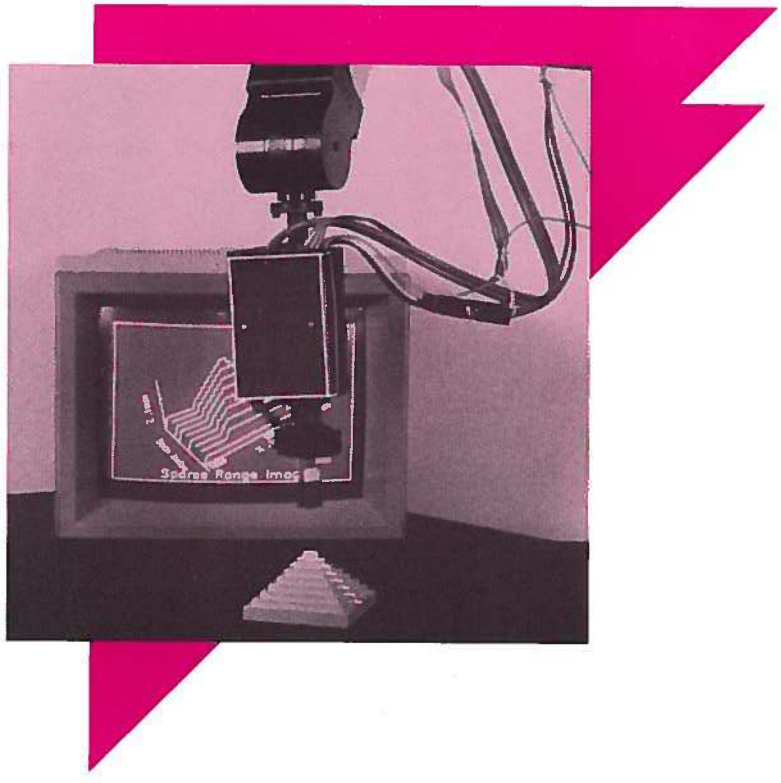
Division of Electrical Engineering
Division de génie électrique

ERA–380

An Overview of Realtime Expert Systems

A.C. Diaz

April 1990



Canada

NRC No. 31759

Additional copies are available free of charge from:

Editorial Office, Room 301
Division of Electrical Engineering
National Research Council of Canada
Ottawa, Ontario, Canada
K1A 0R6

Des exemplaires supplémentaires peuvent être obtenus gratuitement à l'adresse suivante :

Bureau des publications, Pièce 301
Division de génie électrique
Conseil national de recherches du Canada
Ottawa (Ontario) Canada
K1A 0R6

An Overview of Realtime Expert Systems

A.C. Diaz

ERA-380

NRC No. 31759

April 1990

Table of Contents

List of Figures

Abstract

This report presents an overview of existing realtime expert systems. These are expert systems that, among other things, interface directly with the process environment and its sensors and actuators, must perform under some time constraint, and must do multiple tasks. The report describes issues that must be handled by such expert systems, identifies components required to resolve these issues, and describes the approaches taken by current realtime expert systems to implement each component. It also takes several examples of existing realtime expert systems and describes their features, status, and future directions. This report resulted from a literature survey done in 1988 and includes systems and references up to the end of that year.

Résumé

Ce rapport donne un aperçu des systèmes experts en temps réel existants. Il s'agit de systèmes experts qui, entre autre choses, sont en interface directe avec le milieu de fabrication et ses capteurs et ses actionneurs, qui doivent fonctionner dans des limites de temps données et qui doivent accomplir plusieurs tâches. Le rapport décrit des problèmes qui doivent être résolus par de tels systèmes experts, identifie les composantes requises pour résoudre ces problèmes et décrit les approches utilisées par les systèmes experts en temps réel actuels pour mettre en oeuvre chaque composante. Il présente aussi plusieurs exemples de systèmes experts en temps réel existants et décrit leurs caractéristiques, leur état et leurs orientations futures. Ce rapport fait suite à une étude de la documentation menée en 1988, et les systèmes et les références mentionés datent d'avant la fin de cette année.

An Overview of Realtime Expert Systems

A.C. Diaz

1 Introduction

Improvements in computing power and software tools technology have brought about a resurgence of interest in applying knowledge-based (KB) or artificial intelligence (AI) techniques to realtime applications. The number of realtime expert systems (RTESs) that have cropped up since the early 1980's illustrates this interest.

Existing RTESs fall under several categories according to the major task they perform. There are interpretation systems, like Hasp/Siap [1], Stammer [2], MXA [3], and an ESM prototype [4], which take input signals from sensors and interpret the situation or scene that caused the signals. There are prediction systems, like ALFA [5], that infer likely consequences of given situations. There are monitoring and diagnostic systems, like Annie and Violet [6], Cooker [7], SA [8], SCAN [9], and DEA [10], which continuously gather data, interpret it, and set an alarm when a problem or fault is diagnosed. In addition to monitoring and diagnosis, repair systems, like Reactor [11] and FLES [12], suggest plans for remedial actions for the identified malfunctions. Finally, there are control systems, like Artifact [13], YES/MVS [14], and PICON [15], which govern the overall behavior of the system, performing the necessary and appropriate control actions to keep the system at its normal state.

The existing RTESs may be categorized not only by the tasks they perform, but also according to the areas in which they operate. There are systems used in military applications, for example, Stochasm [16], Hasp/Siap [1], Stammer [2], and an ESM prototype [4]. There are systems used in communications applications, such as an AI system for military communications [17] and Dantes [18]. There are systems used in different industries: Reactor [11] and CEALM/REALM [19] in the nuclear power industry, ALFA [5] in the electric power industry, LINKman [20] and a rotary cement kiln supervisor [21] in the cement manufacturing industry, and FALCON [22] and DEA [10] in the chemical process industry. There are also RTESs for robotics [Hallam 87], medical (e.g., [Rader et al. 87, Hayes-Roth et al. 88, Factor et al. 88]), aerospace (e.g., [12, 23, 24]), and financial [Behan & Lecot 87] applications.

This report forms part of the feasibility study for the PRECARN project to develop a RTES shell called ARTEMIS. It attempts to give the reader an idea of the issues addressed and the techniques used by the existing RTESs. Section 2 discusses the issues addressed by RTESs and identifies components required to resolve these. The subsequent subsections describe the techniques used in current RTESs to implement each component. Section 3 takes several examples of existing RTESs and briefly describes their features, status, and future directions. The PRECARN project proposes to build a toolkit with the first domain being process control; hence, the emphasis in this section is on generic tools and shells, and process control systems. Section 4 concludes the report with some comments and observations about the existing RTESs drawn from the literature surveyed for this report. This section also includes a table (Fig. 13) listing all the acronyms used in this report. The

bibliography is divided into two parts. The first part gives references to available RTESs, up to the end of 1988, grouped by system. This is, of course, not a complete list of all available RTESs. The second part gives other references that are of interest to this topic.

2 Issues Addressed and Techniques used by RTESs

In many papers [Bennett 87, Laffey et al. 88, Sauers & Walsh 83, Taunton 87, Sloman 85, Bowen 87, Dvorak 87], authors have identified a list of issues and problems that must be resolved by ESs operating in real time. These issues may be divided into three categories: performance, data and knowledge, and integration.

Performance issues address the question of *how the system should perform.* This includes

1. *Time constraints.* RTESs must be *timely.* By timely is meant that the system has the ability to respond when this is needed. Data must be received from the external environment at a certain rate, and the system must make the best decisions and give the best possible response when this is needed. Timeliness for one system may mean response in microseconds, whereas in another system it may mean response in minutes. Achieving realtime performance may be approached in several ways.

   - Use of a focusing mechanism that constrains the search, in the inference process, for knowledge that is relevant to the current state of the problem and thus is eligible for execution. This is important because many inference engines implement algorithms of search strategies that, if not constrained, result in combinatorial explosion. Two techniques have been used to implement focusing. The first explicitly represents control knowledge and makes this available to the inference process. This control knowledge has the task of guiding the inference process and focusing it on the important and relevant portions of the KB. It also helps resolve trade-offs, such as immediacy vs. accuracy, degree of certainty vs. level of abstraction, giving priority to one subproblem over other subproblems, and inference transparency vs. inference optimality.

     The second organizes the KB by grouping and partitioning it into functional chunks. The inference process then has a way of determining which chunks are relevant and which are not. This implies *a priori* knowledge of when and how a piece of knowledge is used (i.e., the use of the knowledge is fixed from the time of development and not dependent on the context of the running system).
   - Separation of the development and run-time environments and use of compiled rather than interpreted code.
   - Hardware solutions. Several options are being explored to speed up the execution of ESs. Firstly, there is work underway in implementing the inference engine using VLSI technology. At the AT&T Bell Laboratories an inference architecture that handles uncertainty and performs approximate reasoning has been implemented on a chip using VLSI technology [25]. Secondly, there is the use of parallel architectures. It has been reported in [Gupta 85] that 90% of the rule-based interpreter's time is spent in the match step (which determines which knowledge may be activated at a given time) and 10% on the conflict resolution

(which chooses from the list of activatable knowledge the one to execute) and the act (which actually executes the knowledge) steps. This group is experimenting with implementing the Rete Match algorithm [Forgy 82] using parallelism to speed it up. Other work has been directed towards multiple ESs, possibly executing on parallel processors, solving multiple problems or different parts of a single problem.

In addition to responsiveness, guaranteed response time is essential in *hard-realtime systems*. The system must therefore be able to assess the time it has available to achieve a goal, provide different levels of solution that may range from reflex-type reactions to in-depth inferencing, and assess goal priorities and timings in order to make the appropriate resource allocations.

2. *Critical decisions.* Oftentimes the decisions and responses the system makes should not only meet the time constraints, but are significant for reasons of safety — for the plant and its personnel. The system must, therefore, meet high standards for reliability, availability, and recovery after crashes. It not only needs some focusing mechanism in order to zero in on the areas significant to the task and an efficient inference algorithm, but also must have good knowledge of the domain on which it is working. This includes knowledge on

   - heuristics used by the domain experts
   - information on the structure of the process
   - standard operating procedures
   - physical design of the plant
   - possible remedial actions to known faults
   - expected behavior

   With regard to recovery after crashes, there must be a record of the data, results, and decisions made by the system before the crash and the ability to use these journals to recover after the crash. In other words, there must be some way of archiving not only the data used but also the trace of the inference process. The questions to be answered here include: how much information should the system keep, what procedures should be followed when archiving this information, and what medium should be used to store this.

3. *Continuous operation.* In most conventional ESs, once the last rule in the agenda has fired, the system stops. This cannot be the case for a RTES. It must be able to scan continuously for the occurrence of certain events and respond accordingly. Also, in spite of the discovery of a partial or complete failure of one or more parts, the system must continue to operate and should be terminated only by the operator or some catastrophic external event.

   For the system to operate continuously, there must be a mechanism to clean up old memory elements, either by deleting them or by archiving them for later retrieval if necessary. In addition, the system must have an efficient method for continuously allocating, deallocating, and garbage collecting unused memory. Caution must be

taken to ensure that memory management overhead does not degrade the performance of the system.

4. *Asynchronous events.* When operating in real time, events may occur that do not adhere to any schedule. A RTES must be able to receive these events and respond to them properly. It should therefore be able not only to receive information as it asks for it, but also to receive information that it has not specifically asked for at unscheduled and unexpected times. To do this, the system must be able to interrupt what it is currently doing in order to start processing a higher priority task, which may be to receive asynchronous input. It should be noted that priorities assigned may be context-dependent based on the total, current operational state of the system.

5. *Limited resources.* The system must be able to operate and manage limited available resources in terms of computing power, processing time, and space. It must, therefore, be efficient and have a way of focusing attention on relevant portions of the domain without ignoring on-going processes. Actions to take may have to be selected on the basis of multiple, and not necessarily consistent, objectives. The control knowledge and the inference engine again play an important role in addressing this issue.

   With regard to space, the issues of memory management, garbage collection, and archiving come up once again.

Data and knowledge issues address the question of *what are the characteristics of the data and knowledge processed by the system.* This includes

1. *Large KBs.* RTESs usually require and receive a lot of data and knowledge that must be stored in the KB. The system must, therefore, be able to handle large KBs that are easily constructed, augmented, modified, and maintained.

   In construction of large KBs, it is important to have knowledge acquisition (KA) tools that help the user enter the knowledge, verify and validate this entered knowledge, and store it in the KB. These tools are also useful when one is augmenting, modifying, and maintaining these large KBs. In addition, another important tool is an intelligent user interface that accesses the available knowledge in the KB and organizes this in such a way as to show the user only pertinent information in an easily understandable format. This is a valuable aid that will enable the user to know what is already in the KB and to determine what must be added, changed, or deleted. To accomplish this, it is essential that the KB itself is well-designed and organized to enable easy access to the desired information. Finally, a third tool that helps in handling large KBs is a learning mechanism that automatically augments the KB.

2. *Uncertainty.* Uncertainty in a RTES comes in two forms: the input data received by the RTES may not be completely reliable, may contain noise, or may be inconsistent with other data or with expectations about this data; and the conclusions and hypotheses arrived at by the RTES may be uncertain because of uncertain evidence (as in noisy data) or because the knowledge about the process itself is uncertain, imprecise, and incomplete. A RTES must, therefore, be able to function with missing, incorrect, inconsistent, approximate, or irrelevant data and with uncertain, imprecise,

and incomplete knowledge. To achieve this a RTES needs a mechanism that determines the veracity, certainty, and reliability of the data it receives. It must be able to represent data reliability and knowledge uncertainty information in the KB, and it must also be able to propagate this through the inference process. In addition, uncertain conclusions and hypotheses bring about competing and alternative conclusions and hypotheses. A RTES must allow for the generation of these competing and alternative hypotheses and conclusions and have a mechanism for rating these in order to pursue those that seem the most promising.

3. *Dynamism and nonmonotonicity.* Data received by a RTES are not durable and may cease to be valid with time. Conclusions and hypotheses made by a RTES also do not remain static. As new data and information come in, prior conclusions and hypotheses may cease to be true or the measure of belief in these may change. A RTES must therefore be able to deal with dynamic and nonmonotonic input data and inferred facts and maintain a consistency between its view of the external world and what is actually there. To achieve this, the RTES must be capable of updating its KB as frequently as the application requires. It may adopt a sampling approach or it may provide an interrupt facility. In addition, a RTES must be able to know when its data are out of date and it must be able to backtrack out of a wrong path of reasoning in order to pursue an alternative path. It must also be able to clean up (or archive) information not currently needed. All through this process of backtracking and cleaning up information, the RTES must maintain the consistency and integrity of the KB.

4. *Temporal reasoning.* The dynamic history of measurements may be important to the reasoning process of the system. The plant state is not fully defined by measurements at any one time and reasoning about past and present, as well as the sequence in which they occur, is important. A RTES must therefore be able to maintain, access, and statistically evaluate historical data. To achieve this, the issues of memory management, garbage collection, and archiving come up again.

   In addition to being able to reason about past and present states, the system should also be able to reason about possible future states. This is achieved by having knowledge on the dynamics of the domain under consideration (e.g., knowledge on how the domain behaves in normal or abnormal situations, how this behavior propagates in time, the conditions that will bring this about, etc.). A RTES must have a mechanism, like a simulator, that uses this knowledge to infer future behavior and thus aid in the decision on which hypotheses should be pursued and provide information used to react to disasters before they can occur.

Integration issues address the question of *how the system should integrate with other systems.* This includes

1. *Integration of conventional and AI techniques.* Most, if not all, of the complex systems being developed have portions that require numeric computations and other portions that require symbolic computing. Much work has been done in implementing these systems without using AI. The issue here then is to decide which portions to implement

using conventional methods and which portions to implement using KB techniques. These two portions must then be efficiently integrated or interfaced.

2. *Interface to external environment.* A RTES gathers data from different sources such as sensors, actuators, operators, and other systems. These other systems may take the form of other ESs or other conventional systems like a management information system (MIS) and a process controller. The RTES must, therefore, interface with these. Ideally, the RTES should not cause modifications to these other systems. In addition, the RTES might also have to operate in harsh environments, where the other systems operate, such as the sensors and actuators.

An ES usually has three basic components: a KB, an inference engine, and interfaces to the external world. The KB usually has two parts: one that contains the knowledge about the domain called the domain knowledge and the other, called working memory, that contains instantiations of the domain knowledge as applied to the problem on hand. The inference engine or reasoning mechanism uses the KB to reason and infer conclusions, hypotheses, or solution elements that solve the problem. The interface to the external world allows the ES to communicate with its users and other information sources and destinations.

A RTES that addresses these performance, data and knowledge, and integration issues must have

- an architecture that allows timely, efficient, and continuous operation, good knowledge about the domain, some focusing mechanism, and working memory that allows the representation of alternative and competing hypotheses. The KB should, therefore, include not only domain knowledge and working memory, but also control knowledge to explicitly represent control strategies that guide the inference process.

- an inference engine that includes not only an intelligent scheduler and interpreter, but also other reasoning mechanisms like a mechanism for handling uncertainty and maintaining KB integrity, a temporal reasoning mechanism, an interrupt handler with the ability to receive asynchronous input, memory management, garbage collection and archiving facilities, and some learning capability.

- interfaces to the outside world including an intelligent user interface with KA tools, a process control interface which interfaces with the plant and its instrumentation, and an interface to other KB or non-KB systems.

The following subsections give a brief description of the techniques used by existing RTESs to mechanize each of these components.

2.1 The Knowledge Base

The knowledge base (KB) is divided into three major parts: domain knowledge, control or metalevel knowledge, and working memory or dynamic knowledge.

2.1.1 Domain Knowledge

Domain knowledge is defined as that knowledge (specific to a domain) used by the experts to solve problems within the target domain. In the literature on RTESs there is a consensus that two general types of domain knowledge are needed.

The first type consists of knowledge about the structures and underlying mechanisms of the target domain. It is an explicit representation of the descriptions of the different objects, components, entities, or concepts of interest to the system, and the interrelationships among them. It has been called by different names such as function-oriented knowledge [11], conceptual knowledge [26], structural and connectivity descriptions [27], and passive (declarative) knowledge [28]. In this report, this declarative type of knowledge will be referred to as structural description.

The second type is an explicit representation of the knowledge about the environment of the system and behavior under some known condition (or, in order to accomplish some task). It is usually embodied in rules of thumb or tricks of the trade used by experts and acquired from standard operating procedures, past experience, experiments in test systems, and analysis of computer simulation models. It too has been known by different names such as event-oriented knowledge [11], operational knowledge [26], state and behavior descriptions [27], and active (declarative) knowledge [28]. It is often referred to as heuristic or *shallow* knowledge. This type of knowledge must encode actions and events that occur in the world under consideration and embody knowledge on the performance of skills [Barr & Feigenbaum 81]. In addition, some formalism is needed to represent the temporal sequence of events and their causal relations. This procedural type of knowledge will be referred to, in this report, as behavioral knowledge.

For most of the current RTESs, structural description has been implemented using some form of object-oriented representation. Some of the representation schemes used include

1. Semantic or associative networks [Quillian 68]. A network of nodes and links where the nodes represent concepts and the links represent relationships between them. One form of relationship frequently used in these types of network is one between a parent node and a child node. The child is a specialization of its parent(s), inheriting properties from this. A parent node is a generalization of its child(ren) node. Different schemes implementing some form of a semantic network are

    - Frames [Minsky 68]. Data structures with slots to contain associated attributes and fillers to contain values for these attributes. They form a way of grouping information and expectations pertaining to a stereotypical object. They are used in systems such as Hasp/Siap [1], Starplan [29], PICON [15], and G2 [30] to describe the objects of interest to the system together with information such as how to use the object and what other objects are related or associated with it. These frames are sometimes organized to form a network to show relationships between frames and the concepts they represent, as done in FLES [12]. In other systems, e.g., DEA [10] and MODEX2 [Venkat & Rich 88], methods and demons form the procedural knowledge about the object and are also specified in the frame for the object. Methods are procedures attached to the object itself and

perform functions such as calculating outputs given the input values. Demons, on the other hand, are functions associated with a frame slot or attribute and are triggered when the slot or attribute values change, e.g., a change in a critical sensor value.

- Scripts [Schank & Abelson 77]. Frame-like structures designed for representing sequences of events and expectations about these. It is used in SCAN [9].
- A classification hierarchy [Quillian 68]. A lattice-like structure describing the components of a system and their interrelationships. Each component is described by a class in the lattice, and the lattice itself describes the interrelationship amongst components. Again this may show inheritance relationships between classes of objects. An object or event of interest to the system is created as an instantiation of a class in the lattice. This scheme is used in ESCORT [31] and Dantes [18].

2. Condition–action pairs. A representation scheme whose condition part specifies the situations under which the knowledge may be executed. This is also called the antecedent or evidence. The action component is the part that is done if and when the knowledge is selected for execution. This is also called the consequent or hypothesis. Representation schemes that implement condition–action pairs include

   - the response tree technique in Reactor [11]. It uses a diagram, in the form of a tree, that shows the successive paths which can be taken to provide a given safety function. This tree must be traversed (given the parts that are disabled) to find a path that can go from the top (the safety function being performed) to the bottom of the tree. The lower nodes of the tree then provide the conditions that enable its higher level nodes.
   - production rules. The most often-used scheme to represent condition–action pairs. They are usually of the form *IF condition THEN action.* RESCU [28] uses them to represent the specification of the desired products and process models that allow various ways to estimate the product quality.

In the case of behavioral knowledge, most systems use some form of rule representation. Some representation schemes for this type of knowledge include

1. Production rules. The information in the production rules represents heuristic knowledge and actions that describe the flow of information between the passive knowledge sources. This scheme is used in RESCU [28], Hasp/Siap [1], HEXSCON [26], ALFA [5], and YES/MVS [14].

2. Inference rules. These are an extended form of production rules that contain measures of rule strengths used in representing knowledge uncertainty. They are usually of the form *IF evidence(s) THEN (to degree S,N) hypothesis*, where S and N indicate levels of sufficiency and necessity. N is usually a multiplicative factor showing how the odds on the hypothesis are changed by observing a lack of the evidence, whereas S is usually a multiplicative factor showing how the odds are changed by observing the evidence. These inference rules are organized into an inference network where schemata represent the evidence, a schema represents the consequent or hypothesis,

and another schema describes the relationships between these. Stammer [2] and PDS [32] are examples of systems that use inference rules and inference networks.

3. Some systems organize rules in order to scope the problem and act as a focusing mechanism. For example, AND/OR and decision trees are used to reflect the condition and actions of the rules where only selected branches of the tree are worked on. Another method of grouping rules is by forming rule sets, linked via a network for problem scoping and to ensure that only potentially relevant rules execute.

   Blackboard systems, such as MXA [3] and an ESM prototype [4], implement knowledge sources (KSs) as groups of condition–action rules with procedural knowledge supported by the ability to use functions and procedures in the rules. These KSs operationalize the *cooperating experts* of the blackboard model, which share the burden of solving the problem on hand. These experts or KSs communicate with each other through the blackboard.

4. Rule frames. This scheme combines the production rule and frame schemes, where the frames describe the rules associated with it. Frame slots include the name of the rule, a textual description of the rule, its overall worth and type, and other information. Rule frames are used in PICON [14] and G2 [30].

5. Rules and demons. Dantes [18] combines rules and demons (which are procedural attachments to structured objects). It follows an approach that is very similar to LOOPS [Stefik & Bobrow 83], wherein a rule is defined on an object class and rule application takes place for an instance of this class. Rules have the properties of class, name, trigger, state, environment, and body. Only the rules associated with the appropriate objects are awakened and executed, thus providing a focusing mechanism.

6. The relational model. In the Artifact shell [13], the condition of the rules is a relation composed of some conjunction of state attributes, and the action is some modification to the control input of the system. A collection of these relations that allow all condition–action rules to be deduced is compiled and forms the relational model. This model reduces the number of heuristics required to describe the process.

7. Linguistic rules. These are rules, much like production rules, that express linguistic approximations. In these rules, each term has an associated fuzzy membership function that for a given process condition is used to establish a value in the interval (0,1); the logic value is a measure of the fulfillment of the condition for a given process state. Therefore, instead of having a rule condition of the form *if traffic density* > 0.75, a linguistic rule will have the condition *if traffic is very heavy*. For more information on this, the reader is referred to [Zadeh 65, Holmbad & Ostergaard 82]. LINKman [20] and the AT&T VLSI chip [25] are examples of RTESs that use linguistic rules.

8. Goal/subgoal representation. This representation scheme organizes the system into a lattice (usually a hierarchy) of goals and subgoals, where each goal represents a plan to be carried out and its subgoals represent its decomposition into subplans. Some form of the goal/subgoal representation scheme is used in the Artifact shell [13] and Cooker [7].

2.1.2 Control or Metalevel Knowledge

Metalevel knowledge is defined as knowledge about knowledge, such as knowing the extent and origin of knowledge, the reliability of certain information, the relative importance of specific facts, and having a feel for the progress in solution of a problem [Barr & Feigenbaum 81]. Control knowledge, on the other hand, is that knowledge used to determine what is to be done next, given several alternatives [Hayes-Roth 84, 85]. To make intelligent control decisions, this control knowledge must contain metalevel knowledge. For this reason, metalevel and control knowledge are taken as equivalent.

It is rare that control knowledge is separated from the inference mechanism or the domain knowledge and explicitly represented. This may be because domain experts often have difficulty retrieving control knowledge and differentiating it from the domain knowledge [Hayes-Roth 84, Hayes-Roth 85]. In the available systems that do represent control knowledge explicitly, more often than not they use the same representation scheme as used for the domain knowledge, more specifically the behavioral knowledge.

In the blackboard systems Hasp/Siap [1] and MXA [3], control knowledge is isolated and used to decide what objectives are important at the moment, which strategy to employ, and which ordinary or domain knowledge sources (KSs) to fire in order to employ the chosen strategy. The chosen objectives may be interrupted or several may be pursued at one time. These KSs are usually arranged in a hierarchy that represents a plan for organizing the problem-solving activities needed to form hypotheses. The inference engine percolates down this hierarchy from the highest level control KS (in most cases this is the most general or the KS of the greatest granularity) to the lowest level control KS, and then on to the domain KSs; it is guided and focused at each step. Note that there is no competition between metalevel and ordinary KSs (i.e., control or metalevel KSs are always given priority over domain KSs).

In a system that chunks and partitions its KB by function (as in ESCORT [31]), some of the identified operations clearly make control decisions. These functions then contain control knowledge.

Note that control knowledge seems to be isolated and explicitly represented in systems that have a functional division of the system. This is true for blackboard systems that often have separate KSs for separate functions and for systems that divide the application into different operations with control considered as a separate function. For object-centered systems though, this explicit representation of control knowledge appears to be absent.

2.1.3 Working Memory

Working memory (WM) is that part of the KB that contains hypotheses, decisions, and (partial or final) solutions obtained during the problem-solving process. It is sometimes called the *dynamic KB* because it constantly changes during the problem-solving process.

Some of the schemes used to represent WM include

1. A database of assertions. These assertions are in a form that may be tested and compared to the conditions or situations that must be met in order to execute a

piece of knowledge. For example, if the KB uses some form of production rules, the assertions use the same format as the condition part of the rules.

These assertions may contain additional information such as some measure of belief and disbelief for handling uncertainty. Some RTESs divide assertions between facts and static information loaded at system start up and inferences which are added to the database as a result of executing some piece of knowledge. These inferences are supported by the assertions representing the conditions that added them. This type of WM representation is used in Stammer [2] and Artifact [13].

2. An instantiated model of the system showing its current status. In some systems like ADS [33] and YES/MVS [14], the KB contains some model of the system. During problem solving this model becomes instantiated with values that describe the current status of the system. This instantiated model of the system is updated when pertinent status information is presented, based upon responses to queries, acknowledgement messages to control commands, and historical data.

3. Blackboard model. The blackboard paradigm organizes hypotheses and decisions made in the problem-solving process. This was first implemented for the Hearsay projects in the 1970's [Hayes-Roth & Lesser 77, Lesser et al. 75, Lesser & Erman 77]. A multidimensional data structure, called the blackboard, is constructed to contain the hypotheses and decisions, with communication between the KSs being only through this blackboard. These hypotheses and decisions comprise the blackboard entries found in specific positions in the blackboard (i.e., each is indexed by the appropriate dimensions of its position in the blackboard). Links between blackboard entries show the relationship between them, with a *degree of support* specified. For example, expectation links relate general entries with the more specialized entries associated to them and reduction links relate specific entries to their more general entries. This is used in Hasp/Siap [1], MXA [3], Expert Navigator [23], and an ESM prototype [4].

   In some blackboard-based systems, like MXA [3], two blackboard structures are used. One contains decisions made in solving the domain problem, and the other contains decisions made in solving the control problem (which, as stated above, is deciding on which actions to perform given several alternatives). This control blackboard shows a plan of invocation for a set of KSs needed to accomplish a task and the progress made so far.

4. Hypothesis network. This is another way of organizing the hypotheses made and is used in ESCORT [31] and Dantes [18]. It is similar to an inference network and is designed to perform the two functions of allowing the generation of explanations and the propagation of truth values. There are three data types usually found in the hypothesis network. These are

   - hypothesis. This states whether a particular assertion is true at a particular instant. It has also been called the result (in Dantes [18]).
   - super-hypothesis. This gives a description of the assertion. It is called a hypothesis in Dantes [18].

- time-slot. This gives a time interval and a truth value for the assertion during that time interval. There are multiple instances of time-slots attached to a super-hypothesis to represent the truth value of the assertion at different time intervals. This is called a symptom in Dantes [18].

The hypothesis network grows and contracts as the necessary knowledge on how to construct it is in the system (i.e., a hypothesis is constructed as a result of an event. If a super-hypothesis does not exist, this too is constructed.) The links between hypotheses represent explanation of cause rather than logical inference. The reader is referred to Fig. 3 in Section 3 for a diagram of a hypothesis network.

5. Object plane. This is a technique for organizing WM on an object, that is of interest to the system, into an object plane. The object plane is composed of an inference network built by rules associated with the object. Rules expressing relations between different objects or between different planes for the same object create inter-plane links between object planes. To express changes in an object in time, an object may have multiple planes with different time-stamps. Figure 10 shows a diagram of multiple object planes in HEXSCON [26].

2.1.4 General Frameworks or Architectures

In addition to the representation schemes given in the preceding subsections, several general frameworks or architectures have been developed and used to represent the system knowledge. Examples of these are

1. Paragon [Ferguson 86]. A hybrid system realizing a model-based paradigm. For its knowledge representation it integrates frames, semantic networks, classification hierarchies, blackboards, demons, transition networks, and rules. It uses a dynamic conceptual network that describes concepts, their interrelationships, and behavior. Conceptual (nodes) and relational (links) entities are the two main types of entities in this network. Both are defined in a classification hierarchy. Each class of entities is characterized by a specific definition that describes the behavior of the entity. The domain is structured along five relationships: definition, composition, functional relationships, structural relationships, and sequential behavior. Each relationship has a well-defined behavior that describes how information is propagated between associated concepts, thus allowing knowledge to be described in a modular way at varying levels of granularity. A concept, on the other hand, can have a set of attributes that describes it as an individual entity. Types of concepts include

   - primitive concepts that organize the local attributes, interrelationships, and behavior of instances
   - abstract concepts that are defined by the aggregation of other concepts, thus showing the external relationships to other concepts while hiding the internal details of their components
   - definition concepts that are a covering set for the specializations that belong to the concept and are used to store generic information and to create new specializations based on the definition of their current specializations

- event concepts that specify how to determine the attribute values in a state and the actions that may occur

To describe behavior, a representational methodology to model temporal knowledge within a declarative framework was developed. Sequential descriptions are used to represent processes and to model the dynamic behavior of concepts within a domain. A process is described by a set of states, the conditions in which a state transition may occur, and the events that take place in each state. An event in a process is used to compute the value of attributes local to the concept that the process describes. Paragon uses a theory of locality that defines the intercommunication of independent parallel processes and constrains access to information by requiring explicit causal relationships to be defined between the communicating processes. This architecture has been prototyped in StarPlan II [29].

2. HCVM (Heuristic Control Virtual Machine) [34]. An event-driven object-oriented computational framework that explores application in realtime heuristic control. This makes no commitment to a particular knowledge representation (KR) strategy beyond the object level. It is based on Advanced Design Systems' Schemer architecture. The system is divided into modules that can be activated or scheduled for execution synchronously or asynchronously (i.e., by being invoked explicitly by some module or by the recording of a data event). For the first case, the modules represent either particular data structures together with procedures used to maintain and manipulate them or procedures that can be invoked according to message-passing conventions. It can be used to build complex KR hierarchies in which hierarchical elements can individually maintain their own state and guard against unwarranted transactions. For the second case, a module-activating event is represented as a data pattern that can be recorded in the HCVM's knowledge space which resembles the blackboard architecture. This has been prototyped in MCM [34].

3. An architecture for realtime KB control [Maletz 87]. This architecture contains three layers of knowledge. The innermost layer consists of a collection of models that can be used to support model-based reasoning. It has six classes of models, namely event, planned action, executable action, system state, time/state, and time constraint models. It uses the pre-condition/post-condition formalism to represent event, planned action, and executable action models. A system state model indicates the current state of the domain and is used by the focusing mechanism. The time/state models are represented as directed acyclic graphs (DAG) and are used to support temporal reasoning. Lastly, time constraint models characterize the time available to the expert system to make control decisions and to perform actions. Time constraints associated with control decisions are based on the current system state model and those associated with the performance of actions are incorporated in the executable action models.

   The middle layer supports control-related reasoning. First, it has domain-specific control knowledge, in the form of rules, that indicate actions that are applicable in certain circumstances. Second, it has a KR for reasoning about control decisions and actions that includes action generation, action simplification, and reasoning about events and planned actions. Third, it has a hypothetical reasoning capability, in the

form of domain-independent metarules, used to consider alternate control strategies and control actions. These metarules generate and maintain a search space that reflects alternate control strategies, actions, and interpretations. This search space is represented as a DAG of contexts. Each context has a set of assumptions and deductions associated with it and there is inheritance between contexts.

The outer layer of this architecture contains the performance improvement and time constraint management techniques. Incremental performance improvement techniques, including data-driven computation and rule compilation, increase the reasoning speed. Rule optimization and focus of attention heuristics decrease the amount of reasoning required. Time constraint management techniques include reasoning at multiple levels of abstraction and search space reduction heuristics (including pruning and simplification operators). This architecture is used in the NAVEX system [24] to coordinate space shuttle radar tracking.

4. A realtime control framework for approximate reasoning [Lesser et al. 88]. [Lesser et al. 88] stress that in order for an AI system to meet realtime constraints, a sophisticated control component that can plan out its problem-solving activities is needed. They introduce a framework for this and have prototyped it in the DVMT system [35]. This approach has the system actively planning the uses of its resources with the key aspects of

   - trying to develop the best solution to the overall problem that satisfies the time constraints
   - having the ability to reason about its criteria for acceptable solutions, its problem-solving state, and its plans for achieving an acceptable solution
   - having the capacity to estimate the time needed to complete its plans
   - should the best (most complete, precise, and certain) solution be unobtainable within the available time, the problem solver resorts to approximate processing strategies that trade the quality of the solution for time.

   The three classes of approximate processing strategies identified are

   - approximate search strategies that limit the number of alternative hypotheses examined
   - data approximations that take an abstract view of the data in order to limit the number of processing alternatives
   - knowledge approximations that work with data approximations and that group and summarize several KSs into a single less-discriminating KS.

5. A parallel process lattice [Factor et al. 88, Factor & Gelernter 88]. This is a network of concurrent processes, organized in such a way as to have an easily visualizable logical structure that reflects the structure of the domain. Each node in the lattice is a separate process that maps a set of input states into a set of output states, and the communication between the nodes follows the edges in the lattice. The nodes are arranged as a series of ranks and layers. Nodes in the bottom rank are wired directly to some external data source, whereas nodes in the higher ranks correspond

to increasingly abstract and general decision procedures. Data and queries may flow upward or downward through the lattice. For example, raw data values received by the nodes in the bottom rank flow upward causing the nodes in the higher ranks to change their state (e.g., change the likelihoods of conclusions made). If a user queries the state of a node, this query may have to flow downward in order to obtain data from the nodes in the lower ranks.

This model is similar to the blackboard model with the multiple cooperating experts or KSs. The difference between them is that the process lattice model organizes the processes and the information flow between them. The blackboard model, on the other hand, organizes the hypotheses and decisions made by the processes or KSs. This indirectly organizes the KSs and the information flow between them, but this organization is not easily visible.

Its authors claim that the parallel process lattice supports parallelism and bidirectional operation naturally. All nodes can run in parallel and the same process can run in either query-driven or data-driven mode. A prototype designed to monitor patients in post-operative ICU has been developed using this model.

2.2 Reasoning Mechanisms

Reasoning mechanisms are tools that enable the system to deduce and verify a multitude of new facts beyond those it has been told explicitly.

This section discusses the different techniques used by current RTESs to realize the different mechanisms identified as required by RTESs.

2.2.1 Intelligent Scheduler and Interpreter

The basic control loop of

- retrieving, in a list that is sometimes called the conflict set, all plausible or possible knowledge that may be executed at each cycle in the problem-solving process (sometimes called the match step)
- refining, ordering, or selecting from this list the knowledge to be activated for the current cycle (sometimes called conflict resolution)
- executing the knowledge on the current state of the problem (sometimes called the act step)

acts on the KB in order to derive solutions to the problem on hand. The scheduler is the mechanism that governs the order of processing the knowledge, thus it performs the first two steps in the control loop. The interpreter completes the cycle by actually applying the knowledge to the current state of the problem, thus performing the third step in the loop.

Some strategies and techniques used by current RTESs to control and schedule knowledge activation include

1. Default control strategies included in shells. Some RTESs are built using shells or tools that supply default control strategies. For example, YES/MVS [14] uses OPS5 that provides two modes of conflict resolution namely, LEX and MEA, which combine the three strategies of

   - refractoriness which disallows the firing of a rule on the same data twice
   - recency which gives priority to rules whose conditions match more recent working memory elements (WMEs)
   - specificity which gives priority to more specific rules or rules with more conditions

   Many times it is found that these defaults are not adequate and must be extended. YES/MVS adds a priority mode which fires rules with a higher priority attached to them.

2. Forward reasoning (or chaining). When using a form of the condition-action pairs scheme, these pairs may chain together (i.e., the action of one forms (part of) the condition of another). Forward chaining, sometimes called data-driven, goes forward through this chain, from conditions that match the data in WM towards conclusions that may be established from these conditions. Stammer [2], LINKman [20], and Dantes [18] are examples of systems that use forward chaining. This is extended in some systems to add more power and flexibility. For example,

   - Stammer [2] uses demons to monitor messages for particular situations of interest. It uses the stream approach as a focusing mechanism to select appropriate rules that may execute and the incremental deduction technique to avoid repetitive execution of a rule on the same database.
   - Dantes [18] uses the classification hierarchies in its domain KB to select and activate the object(s) that may be interested in the newly received event. These activated objects, in turn, select rules for firing. These firings may, in turn, produce deductions that may fire other rules, thus continuing the inferencing and possible updating of the hypothesis network in WM. This logic can be viewed as generalized state transition reasoning.

3. Forward and backward reasoning combined. Backward reasoning (or goal or hypothesis driven) goes backward through the chain of condition-action pairs, from conclusions that have to be established towards the conditions that are necessary to establish these, or the data that has to be acquired. Reactor [11] and PICON [15] are examples of systems that use both forward and backward reasoning. Forward reasoning is used to reach a conclusion and, if this is not possible, backward reasoning is done to get the information needed to proceed.

4. Opportunistic. For the blackboard systems, like Hasp/Siap [1], the MXA shell [3], and an ESM prototype [4], hypothesis formation is opportunistic in the sense that both data-driven (usually from the bottom or most specific level of the blackboard, to the top or most general level of the blackboard) and model-driven (in top-down fashion) techniques within the general hypothesize-and-test or generate-and-test paradigms are used. At each point, the most promising KS is executed. This may be viewed as employing the best-first search strategy.

5. Progressive reasoning. This technique allows HEXSCON [26] to obtain the best possible decision within the time available. Reasoning is done at several levels depending on the response times required. For these different levels of reasoning, different types of knowledge are used.

6. Model-based reasoning. The model-based fault diagnoser [27] uses a modeling process to diagnose abnormal behavior. The system first identifies some behavior that deviates from the expected or desirable operation. It then creates hypotheses that try to identify the correct problem item using the violated expectations principle [Davis 83]. A list of components deviating from normal is formed, which is the basis for the suspect component list. The diagnoser then creates hypothetical environments with particular faults within them and, by comparing them to the real system under consideration, tries to identify the faulty component. Knowing the way components could fail, as well as a list of the suspect components, allows the creation of a set of imaginary environments that each contain a single fault. All these environments coexist within the computer memory and it is possible to switch between environments easily. All imaginary environments are propagated in time, up to the point where the deviation in the real environment occurred. Environments may be propagated in parallel, and when an environment (hypothesis) becomes less probable, it moves down the list of hypotheses but may be brought back into contention later.

   [Kramer & Finch 87, Finch & Kramer 87, Kramer & Finch 88] describe a model-based reasoning approach to diagnosing malfunctions in a chemical process based on behavioral knowledge of the process. In this approach, a process is decomposed according to its functions. It is represented as a directed graph (digraph) of its intentional systems called the process graph. Each system in this digraph is responsible for one important function to be performed by the process. This decomposition is then used to narrow and focus the diagnostic procedure. When some controlled or manipulated variable deviates from its expected value, a malfunction event is observed. The diagnoser first examines the process graph to determine source systems or systems that are directly affected by the malfunction. Then rules are applied to determine which components within these systems are responsible for the faults.

   [Venkat & Rich 87, 88] also describe an approach to diagnosing malfunctions in a chemical process, as implemented in the MODEX and MODEX2 prototypes. This approach not only uses model-based reasoning, but also combines it with compiled, experiential knowledge. It implements this as an object-oriented two-tiered architecture where actions come from sending messages between objects with the object responding to messages by executing procedures called methods. In MODEX2 each class of equipment is represented as an object with various attribute slots and methods that characterize the class of equipment. Methods attached to objects correspond to constraints, confluences, causal, and fault models. Constraints are derived from restrictions based on the laws of conservation of mass and energy. Confluence equations represent the influence of one state variable on another state variable. Causal models are used to generate local effects given some cause. Fault models associate plant faults with sets of events which may potentially be the local cause of the fault. The fault models have a two-tier KB. The first is composed of process-specific compiled knowledge. The second is composed of process-general knowledge. This second

tier is what is defined in MODEX2 as deep-level knowledge and combines structural and functional knowledge about the process. MODEX2's reasoning mechanism first invokes its compiled KB or heuristic rules to try and arrive at a quick diagnosis. If this fails the reasoning drops down to the deep-level KB where first-principles and model-based reasoning are used. The system may flip from one tier to another.

Another approach makes use of a qualitative model of complex systems [Leitch 87, Bobrow 85]. This model is not as detailed and precise as a quantitative model but is suitable for situations wherein the system being modeled is in some way ill-defined or when there is not enough numerical information to enable the use of a quantitative model. The structure of the system is described in terms of qualitative constraints which may be viewed as qualitative differential equations. This differs from ordinary differential equations because some constraints may specify that a functional relationship exists between physical parameters but can only specify qualitative relationships such as monotonically increasing or decreasing (i.e., that the parameters are moving in the same direction or in opposite directions). Qualitative simulation methods [Bobrow 85] use this qualitative model to derive predictions of the system behavior. These behavioral predictions can then be used to do tasks such as fault diagnosis, planning, and training of personnel. This type of reasoning has been used in the QPA system [36].

7. Refutation. The AI system for military communications [17] makes use of the theorem-proving process of refutation. To check the status and availability of some communication link, a question is given about some link. This is changed into a theorem and proved by refutation (by assuming that it is false and trying to refute this, thus concluding that it is true) using the available facts as clauses. Recommendations for repair are made using the same resolution process. It uses the negated theorem (question) and the clauses (available facts) as the if-rules. The system then creates the then-rules with the missing clauses that cause the resolution process to terminate before reaching the nil clause, causing the theorem to be proven. Depending on the number of possible paths that can be followed in the resolution process, there may be a list of recommendations. These can be evaluated by assigning them cost factors (or the cost factors may be assigned to the path in the resolution process). In order to get realtime response for a large system with many networks, many nodes, many links, and different transmission media, this process was implemented in hardware (see the second reference of [17] for details).

8. Depth-first strategy. In this strategy a hierarchy is processed in top-down, left-to-right fashion. The establish-refine control strategy in [Chandra & Punch 87] is an implementation of this. In this system, the malfunctions handled by the diagnostic system are organized into a hierarchy(ies). Each node in the hierarchy is asked to establish the likelihood of the hypothesis it represents, given the data. The node, using the knowledge groups (see the section on handling noisy data for more information on knowledge groups), determines an overall measure of likelihood. If the node establishes (i.e., the malfunction is judged likely), then each subnode is asked to try and establish itself; otherwise, the node is ruled out. Using this paradigm, the control loop consists of running the hierarchy(ies) to find questionable values. Should some be found, then

the resolution techniques to resolve the values (if possible) are run. The database is updated with any changes (as formulated in the resolution of the data values done in the previous step).

The strategy used in Cooker [7] is another implementation of the depth-first strategy. As mentioned in the KB section, Cooker uses a hierarchy of goals and subgoals. To start off, the system sends an activate message to the top level goal which causes it to try and establish its subgoals. The reader is referred to [Kaemmerer & Allard 87] for a description of the Problem-state Monitoring system (PSMS) which is an adjunct to the inference engine of Cooker wherein an augmented transition network of problem states allows the system to model a realtime problem resolution process, even if it follows a nonmonotonic course with subproblems re-occurring in the same episode. Also, the PSMS approach supports the requirement that the system be capable of updating its recommendations in real time and retracting advice that has become unnecessary.

2.2.2 Handling Uncertainty and Imprecision

There are two types of uncertainty that must be handled by RTESs. The first has to do with noisy and imprecise data. Input to realtime systems includes data from sensors which may fail or give incorrect readings. The second has to do with incomplete and uncertain knowledge.

This subsection first discusses techniques used to handle the first type of uncertainty involving noisy data. Then it discusses techniques used to handle incomplete and uncertain knowledge.

Handling noisy data — Data received by a RTES may be incomplete, incorrect, approximate, or irrelevant. The RTES must be able to take these data and determine their reliability. Techniques used to determine input reliability include

1. Redundancy of data sources. Many systems have approached the problem of noisy data by duplicating data sources. For example, duplicate sensors are employed to take the same measurement, thus a discrepancy in their values signifies that something is wrong; or duplicate sensors are employed to take different measurements that are somehow related and this relationship is used to verify the sensor values. PDS [32] and the SICON prototype using PICON [15] are examples of systems that use composite sensors and redundancy between plant measurements, together with some metadiagnosis, to handle sensor degradation.

2. The control approach. It is not necessary to explicitly represent certainty in order to solve problems under uncertainty. Some systems have used an appropriately designed control strategy that exploits properties of the uncertain domain to reduce the effects of uncertainty. This control approach to uncertainty is often used when interpreting noisy, but converging, data. The system establishes and extends islands of certainty and these are used to guide the understanding of the rest of the problem [Cohen 84]. The opportunistic control strategy employed by the blackboard systems, such as

Hasp/Siap [1] and MXA [3], uses a control approach to focus the efforts of the system on tasks that reduce its uncertainty.

3. Retrospective analysis. In most diagnostic systems, spurious readings are handled by preprocessors that smooth or omit these before they are even seen by the system. In PDS [32], it was found that retrospective analysis of unaltered sensor data helped in the refinement of the rule base. The ability to store and analyze successive readings of a sensor, or successive values of any other node, helped the system to handle spurious sensor readings.

4. Using a higher level of redundancy based on diagnostic expectations. Recall that most sensor validation is based on the concept of hardware redundancy of sensors. [Chandra & Punch 87] propose a technique of validating data making use of a higher level of redundancy based on diagnostic expectations. This technique emphasizes that a diagnostic conclusion can, and should, be made based on the preponderance of other evidence, and that the datum value that does not meet expectation should be questioned and further investigated. For their implementation, malfunctions that may occur in the system being diagnosed are organized hierarchically reflecting system:subsystem or function:subfunction relationships between the malfunctions. Each node in the malfunction hierarchy represents the hypothesis that some particular malfunction has occurred and contains knowledge about the conditions under which the malfunction hypothesis it represents is plausible. These conditions are represented as features that are compared with the current data and this set of features is called a knowledge group. The expectations of a malfunction embodied in the knowledge group are designed to give a rating of pattern fit to data. If the fit is not as expected, those data values that do not meet the expectations are identified as questionable and are further investigated. A number of methods to resolve questionable data were identified as making use of

   - redundancy of usage, meaning that the system examines the state of the nodes in the hierarchy to see if any other nodes were satisfied or unsatisfied with the datum value in question; and, depending on the result, increases or decreases the evidence that the value is incorrect
   - redundancy of expectation wherein the subnodes of the node with the data conflict are examined to see if they have any expectations that can resolve the conflict
   - hardware redundancy of need wherein computationally expensive methods are invoked if there is evidence that their use would pay off

   Advantages gained by using this method include

   - questioning data based on diagnostic expectations provides a way to focus on only some data, as opposed to hardware methods which must worry about all data *a priori*
   - even if a data value is questioned, the diagnostic process can continue if other evidence exists for the failure
   - good integration with systems that rely solely on hardware redundancy by using those values as data for both diagnosis and a higher level of data validation

- programming of such a system is facilitated by tools like CSRL [Bylander & Mittal 86]

Handling incomplete and imprecise knowledge — Knowledge in the KB for a RTES may be incomplete or may contain some measure of uncertainty. As discussed in the section on the KB, this uncertainty is represented in the domain knowledge as some measure of confidence on the conclusion given the evidence, such as the measure of rule strength found in inference rules. WMEs also have some measure of confidence derived from the measures of confidence of the evidence (which are other WMEs) supporting this WME, together with the measure of uncertainty attached to the knowledge that added this WME and some combining function. This combining function is the mechanism responsible for propagating uncertainty through the inference process. The different techniques used for implementing this combining function use

1. Numeric approach. This uses various *a priori* and conditional numeric estimates to propagate uncertainty through the inference process. Three such approaches used in RTESs are

   - Certainty factors. This is the most common technique for handling incomplete and imprecise knowledge. It was first implemented in Mycin [Shortliffe 76]. Systems that use a similar algorithm include Stammer [2], PDS [32], and Comdale/C [37]. As in Mycin, each rule in the KB has a strength which is an estimate of confidence in the conclusion drawn by the rule given that all its conditions are certain (with a confidence equal to 1). In addition, each WME has associated to it a measure of belief (MB) and a measure of disbelief (MD). The MB and MD of the conclusion are calculated using some combining function like $MB = MB(conditions) \times strength(rule)$ and $MD = MD(conditions) \times strength(rule)$.
   - Bayes' Rule. The application of Bayes' Rule is the method used in the MXA shell [3] to update probabilities (or likelihoods) of hypotheses (or blackboard entries). This takes into account the probability history of the hypothesis as summed up by its current probability, and the probabilities of all supporting hypotheses, both for and against. It uses both the prior and current probabilities of the supporting hypotheses which are recorded as part of the hypotheses. The affirm and deny strengths plus an overall scaling factor are associated with this link set up between hypotheses.
   - Dempster-Schafer theory [Shafer 76]. This is an alternative numeric approach found in RTESs such as HEXSCON [26]. It allows the assignment of degrees of belief to subsets of the frame of discernment (the set of all singleton hypotheses under consideration), whereas Bayesian theory assigns degrees of belief to singleton hypotheses. It also allows the distinction to be made between uncertainty and lack of information. As evidence becomes available, Dempster's rule of combination updates the values of beliefs and probability assignments for the subsets of hypotheses.

2. Fuzzy set theory or logic [Zadeh 65]. Fuzzy set theory or logic uses linguistic statements wherein uncertainty is estimated using variables such as high, low, and OK.

It translates these into forms which mathematically represent the vagueness of the concepts and propagates them using the fuzzy intersection and union operators. It is used in LINKman [20], the AT&T chip [25], and an AI system for military communications [17]. The last two implement their inference engines in hardware. The reader is referred to [17] and [25] for more information on these implementations. In the AT&T VLSI chip for approximate reasoning [25], the inference processing unit consists of two basic units — a minimum and a maximum unit — used to implement the fuzzy intersection and union operators.

3. Model of endorsements [Cohen 84, Cohen & Grinberg 83]. This is a heuristic approach to reasoning about uncertainty. It is used in an ESM prototype [4]. Records of information affecting certainty called endorsements are associated to a hypothesis. This information includes the kind of evidence available, e.g., direct, corroborative, and eyewitness, and the kinds of methods used to produce the current hypothesis from uncertain predecessors, e.g., averaging and eliminating one of the conflicting estimates. Endorsements are ranked; the hypothesis with a superior endorsement is considered more certain than one that is less well endorsed. These may also be propagated over inferences, but in a manner that is sensitive to the context of the inference.

2.2.3 Consistency Maintenance Mechanisms

The environment in which a RTES operates is dynamic and nonmonotonic. Data received and conclusions reached by the system may have to be adjusted when new information alters their bases of support or when the data are found to be out of date. The system must take care of maintaining the consistency and integrity of the KB when doing these adjustments. A technique for implementing this revises the measures of confidence or belief attached to the data and conclusions as found in WM. Alternatively, a truth maintenance procedure as found in truth maintenance systems (TMS) [Doyle 79] has been used. In TMS, revision of belief is handled by finding the WME that must be adjusted and adjusting this and all WMEs supported by it. For example, if the evidence supporting a conclusion is adjusted from being believed to being disbelieved, this evidence is withdrawn together with all the conclusions derived from it and the other conclusions derived from these first conclusions and so on along the chain.

Not much information has been found in the literature on how RTESs handle belief revision. Exceptions found were Stammer [2] and MXA [3]. Stammer [2] takes care of two situations: later firing of rules that previously failed and fired rules that should be withdrawn. The first situation is addressed by pulsars which are streams of pulses. The second situation is handled by the delayed computation of confidence. It is calculated only when needed using the *current* degree of belief in the conditions.

The MXA shell [3], on the other hand, treats an update to the likelihood of a hypothesis as a significant event. This prompts the system to recalculate the likelihoods of all hypotheses, directly or indirectly, supported by this one. In this way, all hypotheses are kept up to date and consistent.

To handle data that cease to be valid in time, some systems like PICON [15], G2 [30], and RESCU [28] associate a time-stamp and a validity interval with the data. The time-stamp specifies when the data were received or inferred, and the validity interval specifies

the duration for which this is valid. These data will only be used if their validity intervals have not expired.

2.2.4 Temporal/Spatial/Causal Reasoning Mechanisms

In a RTES, objects, events, and the relationships between them have these characteristics: they may change with time, they occupy space, and there may exist some cause-and-effect relationships between them. Thus, a RTES requires some mechanism that uses these characteristics and information to perform temporal, spatial, and causal reasoning.

It is important for the reasoning process to have knowledge on past and present states and possible futures. Knowledge on past and present states is obtained by using historical information which, in many RTESs like YES/MVS [14], PICON [15], ESCORT [31], HEXSCON [26], Dantes [18], G2 [30], and SCAN [9], are time-stamped. Variables (which may be data from the sensors or calculated/inferred fields), time-slots in the hypothesis network, and object planes have times associated with them to show when the measurement was received or when the deduction was arrived at. Most often, these also have associated time intervals which state the duration for which these timed values are good. The temporal reasoning mechanism uses this information to perform types of reasoning such as trend analyses and reasoning about decision lags. These lags are caused by the wait time that must be effected before making adjustments to control actions so as not to mask the effects of an adjustment or overcompensate for it. For example, in a flotation circuit that needs a series of adjustments, there is a wait time of several minutes between each adjustment.

In RESCU [28] the knowledge representation language allows the specification of time in both the antecedents and the consequents of the rules. These temporal qualifiers are then used by the inferencing mechanism to reason both with and about time.

Creation of possible futures by some sort of simulator is made possible by using the domain knowledge. These possible futures allow the RTES to make better judgements on which actions or hypotheses to pursue and to avert disasters before they can occur. For example, in the model-based fault diagnoser [27], the process models contain information on the abnormal behavior of the system. This information is used by the inference engine to propagate imaginary environments (refer to Section 2.2.1) which are used to diagnose faults in the mechanical process.

SCAN [9] uses the internal representations that deal with sophisticated time and order relationships in order to generate multiple hypotheses, fill in information gaps, set up expectations, notice trends and shifts in the action and activity, and provide clear explanations of its inferencing. It uses the formalism of time intervals and relations between these intervals developed by Allen [Allen 84, Allen & Hayes 85]. G2 [30], on the other hand, uses the description of the location of an object to perform some form of spatial reasoning.

2.2.5 Interrupt Handler

In realtime systems, events may happen that do not follow any schedule. There is a need, therefore, for a mechanism that interrupts the system in order to process a higher priority task.

In HEXSCON [26], there are two possible types of interrupts. The processor interrupt preempts the conventional realtime operating system, whereas the logic interrupt preempts the KB part while it is going through its reasoning process. Partially reasoned processes may be kept for later resumption, but this is found to be too expensive and so is not done. It should be noted that in HEXSCON [26], the KB part runs as a single task in the realtime system.

PICON [15] receives its data from a separate module that runs on a separate processor. When this module observes a significant condition, it triggers some PICON (whenever) rules that are then considered for activation by the PICON inference engine. Similarly, the whenever rules in G2 [30] process unrequested sensor values. In addition, G2 takes the rules to be fired and divides them into component tasks (which are very small and take little time). A task cannot be interrupted while it is executing. However, it is given only a certain amount of time to complete. If it has not completed by the end of this time, it is deferred and the next (highest ranked) task is executed.

2.2.6 Learning Capabilities

Machine learning is concerned with improving the performance of the system through automated knowledge acquisition and refinement. Learning filters and incorporates environmental observations into a KB that is used to facilitate the performance of some task.

Stochasm [16] makes use of knowledge of the order in which alarms are triggered to diagnose the cause of malfunctions. Its authors report that it has learning capabilities that allow it to adapt to changes in the system and the environment. Its *patterns* module performs learning by associating the sequential order in which alarms are received with the specific malfunction that causes that sequence. This is done by incrementally building a tree with the alarms being represented as the nodes and with statistical and temporal data stored in these nodes. These data are then used by the *decisions* module to decide on the malfunction, given that there are many possible malfunctions causing the sequence of alarms. It is important to note that *patterns* learns all sequences and their associated causes automatically and uses that knowledge effectively in its pattern recognition work. Nowhere in Stochasm are there rules that explicitly spell out any particular sequence of alarms or explicitly associate a sequence with a particular malfunction. All of this is done automatically, in real time, using a combination of rules on how to learn from inputs and feedback. Another potential form of learning that may be added consists of employing a hierarchical control structure that, based on the data in the nodes, performs and regulates the pruning of older and seldom used branches of the tree.

2.2.7 Memory Management, Archiving, and Garbage Collection Facilities

Most realtime systems deal with large amounts of data and must be in continuous operation. To ensure high performance and efficiency, unnecessary and useless information must be removed. This may be deleted or archived onto some other media for later retrieval, if necessary. YES/MVS [14], ESCORT [31], Dantes [18], and RESCU [28] are some systems that have some capability of cleaning up WM.

ESCORT [31] can delete false hypotheses that have no parents in the network. It also tries to delete children of these hypotheses nodes if the same conditions apply after the parent node is deleted. Note that super-hypotheses and time-slots are not currently being cleaned up.

Several mechanisms are employed in Dantes [18] for the selective removal of historical data. In a process which they call deduction fixing, all deductions which are not used to deduce the given set are removed from the hypothesis network related to an abnormal situation. Using the time-out features described in the section on temporal mechanisms, the system is able to judge when to remove deductions after certain periods.

RESCU [28] attaches a time-validity qualifier to its WMEs and it automatically removes temporally invalid data.

With regard to archiving, many systems keep a history of sensor and other variable values for trend analysis. It is rare, however, to find a system that archives a trace of the inference process. Retrospective analysis of why the ES made decisions that it made is not available after this information is cleaned out of the KB.

After memory elements are deleted or marked as unused, the garbage collection facility gathers these to prepare for reallocation. For LISP-based systems, garbage collection is done automatically, but it is found that this facility degrades the system performance [22].

2.3 Interface

A RTES must be able to communicate with the outside world (see Fig. 1). It must be able to interface with its human users. In addition, it must be able to interface with other systems which may be other systems (KB or non-KB) that may be running on the same or on different machines. For example, RTESs like those used in process control should be able to communicate with process control machines or with sensors and actuators.

This subsection is divided into three parts. The first deals with the human–machine interface (HMI) both in the development and operation modes. The second deals with interface to sensors and actuators or to the process control computer. Finally, the last deals with interface to other systems both KB and non-KB.

2.3.1 Human–Machine Interface

It is important to make the interface with a human user friendly and easy to use; forms and lay-outs familiar to the intended audience must be considered.

There are two modes in which the system may have to interface with human users. The first is the construction, maintenance, and enhancement mode wherein the knowledge builder sets up and maintains the KB, which will be called here the development mode. The second mode is the operator mode where the running system interfaces with the operator. This needs to show to the operator the current state of the system and explain and justify to him or her the behavior, decisions, and actions of the system.

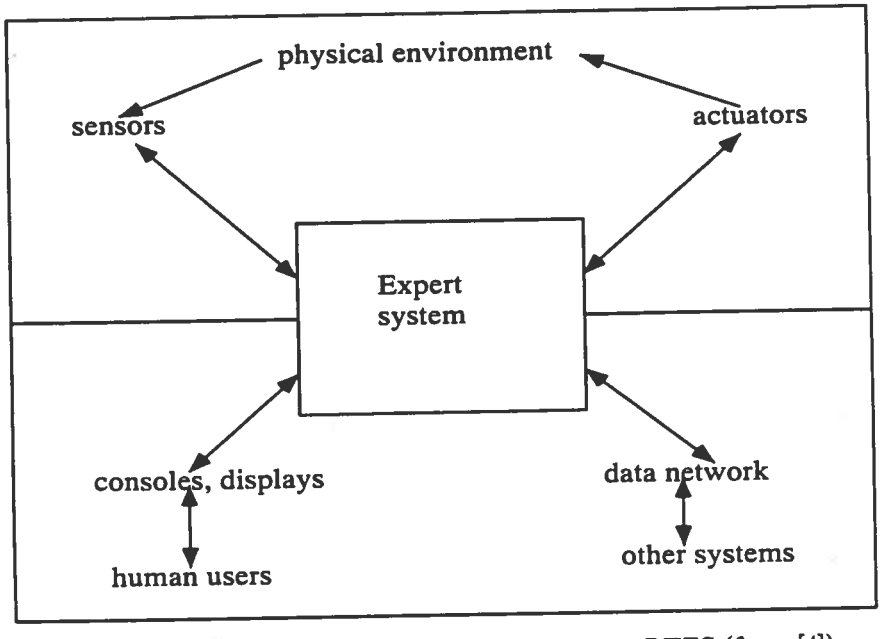


Figure 1: Conceptual model of the interfaces to a RTES (from [4])

Development mode — In this mode knowledge is acquired from the expert and used to construct, maintain, and enhance the KB. It has often been reported that knowledge acquisition (KA) is the bottleneck in ES development. One reason for this may be the lack of user-friendly KA tools. In many of the ESs available, it seems that the person entering the knowledge to the KB must be familiar with how the system works and some AI techniques in order to enter correctly the necessary information. This then requires a knowledge engineer (KE) who, most often, is not the expert. Communication between the KE and the expert then becomes the crucial point. There is work being done to derive methodologies and techniques to facilitate this communication, but this is beyond the scope of this report. Another option to consider is to develop user-friendly KA tools which can be used by the expert to enter his or her knowledge, thus cutting out the middleman. To accomplish this, it is important to consider the way the expert uses knowledge and what form(s) or representation scheme(s) he or she utilizes and develop a system that may receive knowledge in this format. This can then be translated to the representation scheme used by the ES, if necessary. A question that may be raised at this point is that experts in different domains and applications may think or use different schemes to represent, organize, and analyze their domain expertise; is a user-friendly KA tool then not feasible in a generic, domain-independent shell? An answer to this is to provide a tool to the KE to enable him or her to tailor or custom-fit the KA tool so that the expert can use it in the form that he or she is at ease with.

In the development mode of the HMI, some of the tools that must be included are

1. KA tool. This is the tool that will communicate with the expert in a language he or she understands and is comfortable with. This may include an editor to facilitate this entry of knowledge. Some systems like RESCU [28], Comdale/C [37], FALCON [22],

and SA [8] have gone the route of the KE, interviewing, observing, videotaping, and asking the experts to fill forms and solve test cases in order to acquire their knowledge. Other systems, like ADS [33], Stammer [2], PDS [32], and LINKman [20], provide an interactive system that allows the user to enter rules or forms and that provides some ability to extract and concentrate on information pertaining to some object or concept under consideration. Some form of natural language (NL) knowledge entry has also been provided for in systems like PICON [15] and G2 [30]. In addition, these two last systems use an interactive capture tool to acquire the structural description of the process that is being controlled. The user graphically constructs and specifies attributes for the underlying plant structure and models, making use of icons, with each item in the schematic having an associated frame. The user builds a schematic by selecting component objects. The system inquiries proceed with pop-up windows used to get information to fill in the frames in the KB. The schematic may be viewed in multiple windows at a variety of scales. The G2 tool also provides a way for the KE to tailor the shell for an application. The KE may build the hierarchies of icon objects needed to build the application system. The expert can then use these to enter his or her knowledge.

In [Rodriguez & Rivera 86] a method for translating an expert's knowledge, in the form of fault trees and flow charts, into production rules is implemented. The fault tree analysis (FTA) approach is recommended when the knowledge is presented in the form of engineering drawings, operational guidelines, maintenance procedures, and heuristic rules. The fault trees describe the failure or success of the performance of the system being constructed and bridge the gap between knowledge contained in the description of the system and the KB used by the ES. The fault trees are supposed to illustrate how a number of basic causes are combined to lead to a certain undesired or critical state of the system or both. The relation between primary events is a combination of AND and OR gates. A fault tree can be synthesized and incorporated into an expert system by reducing it to its minimal cut set or directly translating it to *if-then* rules (p. 55 of [Rodriguez & Rivera 86]). The flow chart approach is appropriate when knowledge is procedural and is obtained directly from human experts or from a manual or handbook. It explains how the human expert takes decisions and arrives at conclusions. The three basic building blocks, sequence, decision, and loop, can be easily translated to *if-then* rules.

The KA task in SA [8] is aided by the use of the ID3 algorithm [Quinlan 85]. This is a form of learning from examples. In SA [8], examples of an expert's decision-making are obtained by having the expert fill up forms inquiring about diagnoses and their associated symptoms. The expert chooses a diagnosis and indicates a combination of symptoms that give rise to the diagnosis. The ID3 algorithm is then run through these examples to produce decision trees implemented as *if-then-else* statements. The resulting SA1 is an automatically generated Fortran program of around 600 lines, mostly composed of the decision trees, with some text of canned questions and diagnoses.

2. knowledge translator. This translates the knowledge entered by the knowledge builder to a form usable by the expert system. HEXSCON [26] and RESCU [28] provide functions for compressing and compiling the knowledge entered to the more compact form actually used by the expert system.

3. knowledge verifier. This verifies and validates the entered knowledge, not only syntactically, but also semantically, ensuring logical consistency and completeness. In systems like ADS [33] and LINKman [20] some syntactic checks and simple semantic validation are done. In the PICON implementation done in Texaco [15], the VISIT and RETRIEVE functions were added to facilitate KB maintenance and to check for consistency or schematic integrity by specifying certain descriptive information.

4. knowledge tester. This tests the knowledge given some form of data to make sure that it is performing the way it was intended to perform. PDS [32] offers a capability of manually entering sensor values for testing. FALCON [22] uses a simulator to test, offline, the knowledge put into the system and it is also used as a source of information for this phase in the development of the ES.

Operator mode — In this mode of the HMI, the system deals mostly with the operators watching the system during operation. Thus, the system has the three-fold task of

- accepting user commands, queries, and other information
- explaining to, and allowing the operator to examine, the system status
- allowing the operator to modify the status and give commands which may result in the redirection of the system processing

In this mode, systems like ADS [33] and Reactor [11] give the operator the ability to examine WM using some graphic display. Examples of this are the flowgraphs used in ADS [33] which show the current application description and the schematic used in Reactor [11] which shows the success path through the response tree. It is also common to find explanation facilities in systems that provide a straight trace of the inferencing process (e.g., a trace of rule firings).

Interactive ways of obtaining from the system justifications for its decisions and actions and continuous advice on the system state are also made available by systems such as Hasp/Siap [1], Stammer [2], ESCORT [31], and YES/MVS [14]. The system has some mechanism to determine which information would be of interest based on the types of questions the user asks and the current situation of the problem. Some guidance is given to the user to direct his or her line of questioning. In PICON [15] and G2 [30], the knowledge is organized in such a way as to be able to obtain information pertaining to one frame of knowledge and frames associated with it.

Operator input has been looked upon as a special type of sensor data by some systems. This way, this information may contain commands that may affect subsequent system decisions.

2.3.2 Process Control Interface

For the first implementations of process computers, direct digital control (DDC) was used, but this had very serious consequences for the process operation in the event of a computer failure. Control philosophy then shifted to using conventional analog instrumentation for control; and the computer was used to adjust setpoints or to perform supervisory

control. This way, a computer failure did not prevent the control of the process and a shift to manual supervisory control could be done. Later, DDC was again applied, together with analog and digital back-ups that were used in case of failure in the primary equipment. Most recently, distributed DDC has been applied with mini- or microcomputers controlling a part of the control process, supervised by a computer performing the supervisory control function. In the presence of critical process variables, that part may be provided with a backup control [Roffel & Chin 87].

In most applications of AI to realtime control, there is some layer between the environment that actually takes the measurements from sensors and sends actuations to actuators and the environment of the ES. This layer may include a process control computer that may perform some analysis on the data, for example, statistical analysis, scaling, linearization, and digital filtering. These massaged data are then made available to the ES. Examples of systems that do this include Hasp/Siap [1], YES/MVS [14], PICON [15], G2 [30], and PC Online (part of the Personal Consultant Series [38]).

2.3.3 Other System Interface

A few of the available systems provide the capability of communicating with other systems. For example, in ONSPEC [39] there is the capability of communicating with a database management system (DBMS) and a spreadsheet program.

The SICON prototype using PICON [15] allows communication between multiple ESs. The cooperating PICONs are organized into a hierarchy with a master supervisory system supervising the inter-PICON communication. The PICONs may read variables from other PICONs and use these in their inference process.

G2 [30] provides the Intelligent Communications Protocol (ICP) designed to allow communication between multiple G2's and foreign software.

In AF [40], the hypotheses and procedural code for a local knowledge domain are integrated into an activation framework object (AFO) and these collectively form a community of experts which communicate by means of message passing. Each AFO is organized as an independent blackboard system with its own local blackboard and a set of KS procedures. Each AFO is self-contained; and its external interface is defined by the formats of the messages it sends and receives. A system consisting of many AFOs can be distributed over a number of processors with the scheduling of the AFOs being a function of the message traffic.

3 Examples of Available RTESs

This section gives a brief description of several existing RTESs. It is divided into two subsections; the first describes domain-specific RTESs, and the second describes RTES shells or tools.

3.1 Examples of Domain-specific RTESs

The following RTESs are specific to an application or domain. Figure 2 summarizes their features. It should be pointed out that blank slots in both Figs. 2 and 8 do not mean that these features are not available in the system, but that it was not documented in the literature surveyed. It should also be noted that some systems, like ESCORT and LINKman, are put in this section even though they are currently being marketed as tools. The reason for this is that these systems were first developed as domain-specific systems, but later, it was realized that the tools built with these could be used in a variety of realtime application areas.

Name	System status	High perf.	Guaranteed Resp. time	Focus mech.	Noisy data	Knowl. uncert.	Truth mtce	Temp./ Causal/ Spatial	Interrupt handler	Asynch. input	Compile env.	Garb.coll/ Archiving	Continuous operation
Yes/MVS	Experimental	✓					✓	T		✓	✓	G (WME)	✓
Escort	Implemented	✓	✓	✓	✓	✓		C				G (WME)	
Kiln supervisor	Experimental	✓		✓									
LINKman	Commercial	✓		✓	✓	✓		T	✓	✓	✓		✓
Falcon	Completed				✓	✓		T,C		✓			✓
MCM	Prototype			✓	✓	✓		T	✓	✓			✓
DEA	Prototype			✓				C					✓

Figure 2: Domain-specific RTESs

3.1.1 The Yorktown Expert System for MVS operators (YES/MVS) [14]

DEVELOPER: IBM T.J. Watson Research Center, Yorktown Heights, New York

DOMAIN AND DESIGN GOALS: A realtime interactive control system that operates continuously to monitor activities and to solve problems related to computer operations. It acts as an aid to computer operators in dealing with problems such as JES queue space management, problems in channel-to-channel links, scheduling large batch jobs off prime shift, MVS-detected hardware errors, monitoring software subsystems, and performance monitoring.

Expertise put in the ES was obtained from the operations staff at Yorktown, systems programmers, manuals, and the MVS operating systems designers.

ARCHITECTURE: The system is divided into three virtual memory (VM) partitions for speed and functional separation. These are the

- expert VM
- MVS Communications Control Facility (MCCF) which acts as a message filter and translator between the expert VM and the subject MVS
- VM that controls the YES/MVS operator's display console

FEATURES

Knowledge representation: Represents knowledge using OPS5 production rules of the form *(p rule-name conditions − > actions)*. Testing for this system was done online. It was found that a record of test cases could not be used due to the dynamic interaction required; an MVS simulator would have been too complex and too large. Instead, rule walk-throughs, rules partially simulating aspects of MVS, and hand interaction were used in the testing.

Working memory organization: An accurate model of the MVS is impossible due to the high nonmonotonicity of the system. Instead, a current model of the subject MVS, giving a reasonably good description of its state, is used. Time-stamps and validity flags provide information on the currentness of the model.

Focusing mechanism: Not available from the literature (NA).

Inference method: YES/MVS uses the OPS5 LEX and MEA conflict resolution strategies together with a priority mode. This mode uses a task-id and an associated priority which are added to the condition part of the OPS5 rules. All the rules with a priority less than the highest are removed from the conflict set during the conflict resolution step.

Methods for handling uncertainty

Data uncertainty (input reliability required): NA

Knowledge uncertainty: NA

Consistency maintenance mechanism: A truth maintenance approach using OPS5 is used for keeping a dynamically correct priority assignment of the jobs.

Temporal/causal/spatial reasoning: It provides functions that use the time-stamped variables, together with a timer function and timer queue, to support actions that produce WMEs at some future time.

Interrupt handler/asynchronous inputs: A communication phase was added to the OPS5 inference cycle when external messages are picked up and outbound messages are sent from YES/MVS.

Other interesting features:

- It provides a way of cleaning up old working memory elements (WMEs).
- Other extensions made to OPS5 include
  - compiling the right hand-side or action part of the rules
  - tuning of the match step in the inference cycle using LISP macros
  - adding the Priority mode as a conflict resolution strategy
- Rules may be distributed among multiple OPS5 systems using concurrent processes in the form of separate VMs supported by a host computer.
- Continuous operation is effected by the implementation of a LISP function that causes the system to enter a wait stage and not terminate when there is no rule eligible for firing.

LANGUAGE: The expert and display control VMs are in OPS5, whereas the MCCF is in a system language called REXX and in assembly language.

HARDWARE: YES/MVS runs on an IBM 3081, separate from the target MVS system, under the VM/SP operating system. Interface to the target MVS is through an emulated JES 3 console.

STATUS

System: Experimental. The first version, YES/MVS I was in use at the Yorktown Computing Center for most of a year. YES/MVS II is being developed for further experimentation.

Data used: Live on-line data

Performance rating: Peak message rate from MVS to the operator is more than 100 per minute.

Size: 500 rules in both the expert and display control VMs.

Lessons learned from YES/MVS I:

- It is feasible to automate the operator's decision-making process in detecting, diagnosing, and responding to problems in large computing complexes.
- The expense would be very high if every large computing center had to develop its own system. (Development of YES/MVS I required a number of person-years of effort and it is not complete.)
- The economic viability of automated problem handling remains unproven.

FUTURE DIRECTIONS: To broaden the coverage by adding subdomains, such as

- facilities to assist operators during initial program loading of the MVS
- planned and emergency shutdowns
- learning component for scheduling large batch jobs item capacity planning
- configuration and installation

YES/MVS II uses YES/L1 [Cruise et al. 87] instead of OPS5. It concentrates on the aspects of customization and maintenance, so as to make the customization of YES/MVS for installation at a new computing center manageable.

3.1.2 Expert System for Complex Operations in Real Time (ESCORT) [31]

DEVELOPER: PA Computers and Telecommunications, United Kingdom

DOMAIN AND DESIGN GOALS: Designed to relieve the cognitive overload on operators of information systems generating large volumes of dynamic data. It applies expert knowledge to the incoming data and advises the operator on the best course of action.

ARCHITECTURE: ESCORT is composed of five interacting ESs, each concerned with one aspect of realtime processing. The five ESs are

- the event detector that recognizes significant events which may require operator intervention, such as alarm states in the process plant
- the event prioritizer that assigns priorities to the recognized significant events
- the main diagnostic system that analyzes the significant events and generates advice for the operator
- the final prioritizer that orders the diagnoses according to their relative importance
- the scheduler that determines what the system should do next, taking into account the plant state, the diagnoses made, and any operator requests

FEATURES

Knowledge representation:

- Plant item and plant structure definitions are represented using a class inheritance lattice where each component is described by a class in the lattice and the lattice describes how they interrelate.
- Operational knowledge is represented using rules. These rules operationalize the filter operation, initial prioritization, diagnostic system, final prioritization, and scheduler.

Working memory organization: ESCORT uses a hypothesis network (refer to the section on working memory and Fig. 3).

Focusing mechanism: The rules are grouped into rule sets linked via a network for problem scoping and to ensure that only potentially relevant rules execute.

Inference method: Only key sets of rules are fired which in turn may fire other rule sets.

Methods for handling uncertainty

Data uncertainty (input reliability required): It has the ability to recognize control and instrumentation problems (e.g., transmitter failure) and plant failures (e.g., burst pipes).

Knowledge uncertainty: ESCORT uses a strength mechanism (arbitration and negotiation) to range strengths between 0 and 1.

Consistency maintenance mechanism: NA

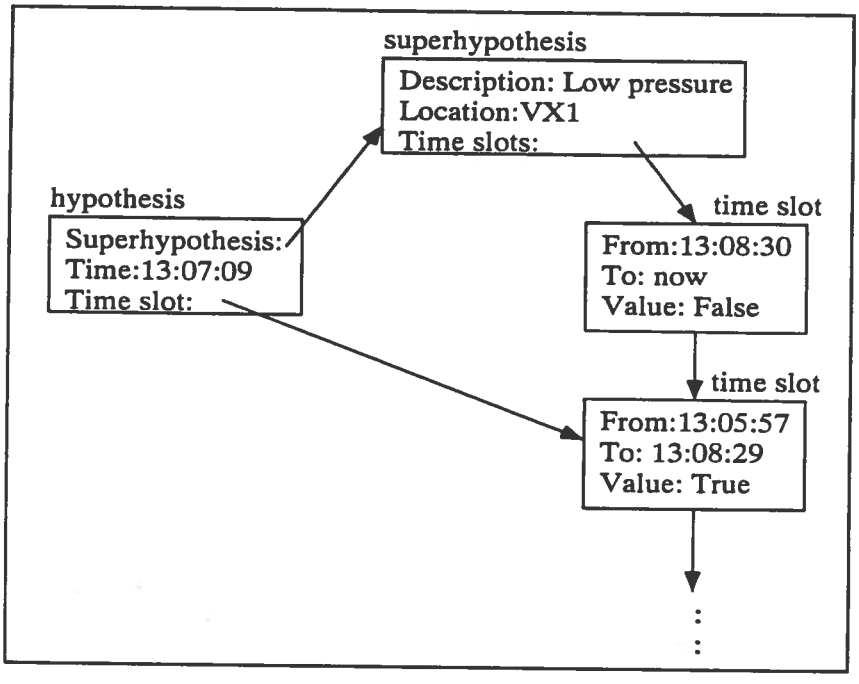


Figure 3: A hypothesis network

Temporal/causal/spatial reasoning: It contains rules that deal with causal reasoning about the process.

Interrupt handler/asynchronous inputs: NA

Other interesting features:

- The system is designed to be installed in the target plant having access to the same data as the process control computer.
- It allows data to be received from, and information to be sent to, management information systems (MIS), process and other computers. Some of these other computers may be ESCORTs located in other plants or doing other tasks.
- The advice to the operator is given in a simple and concise manner, giving the cause of the alarm and the importance of the underlying problem as compared to the other existing problems. It allows the operator to control what information is given.
- In its interface, it can presently show an overview of the simulated process and the status of each control loop. The operator is allowed to adjust set points, change the status and operating control, relieve and shut-off valves.
- ESCORT provides a facility that cleans up false hypotheses that have no parents in the network, together with all their children for which the same conditions apply, after the parent node is deleted.

LANGUAGE: Common LISP and KEE for the expert system and Pascal for the realtime database

HARDWARE: ESCORT runs on a Symbolics 3620 and the realtime database runs on a MicroVAX with interface via Ethernet.

STATUS

System: Commercial. Several feasibility studies are being done in large petrochemical and other processing plants. It is intended to be a tool that must be reconfigured for each implementation. One such implementation is at British Petroleum's butadiene plant at BP Chemicals Grangemouth, which is reported to be the biggest RTES in existence in the United Kingdom [Johnson et al. 88].

Data used: ESCORT is demonstrated using a simulation of a North Sea platform process plant and associated process control systems (where a maximum of 500 analog and 2500 digital signals confront the operator, partial shutdowns occur every half hour, and total shutdowns once a week).

Performance rating: Advice is provided within 1 s of the alarm. Only 15–20% of the rules are fired for a given event.

Size: NA

FUTURE DIRECTIONS: Work will be done to establish the foundations for the very large, supra-integrated computer systems for process plant management, including the three following topics:

- the current control room environment of process plant operators
- connecting to existing (or soon to be) installed information sources
- the process and plant managerial objectives over the next decade

3.1.3 A Rotary Cement Kiln Supervisor [21]

DEVELOPER: University of Newcastle upon Tyne and Rugby Portland Cement, PLC, United Kingdom

DOMAIN AND DESIGN GOALS: To apply ESs technology to process control in cement manufacturing

ARCHITECTURE: The proposed system has six parts: the KB, a database, the operating mechanism which is the inference engine, a calculation tools module that contains arithmetic operators and links to programs such as plant dynamic simulations or fault diagnosis routines, a translator to interface with the human user, and another translator to interface with the plant.

FEATURES

Knowledge representation: This uses production rules in the form *IF (a condition) THEN (a consequence follows)*. It calls this set of rules its long-term memory (LTM).

Working memory organization: The data pertinent to the current state are called the system's short term memory (STM). When the information in this STM has been utilized, it may be replaced in total or in part by new data.

Focusing mechanism: The total operation is organized into a network of discrete tasks and goals. This organization systematically breaks down a task into its subtasks, thus a clear approach to handling complex problems is made available. This is another implementation of the depth-first strategy (see the section on the intelligent scheduler and interpreter).

Inference method: This is goal-driven and uses backward chaining

Methods for handling uncertainty

Data uncertainty (input reliability required): NA

Knowledge uncertainty: NA

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: NA

Interrupt handler/asynchronous inputs: NA

Other interesting features: NA

LANGUAGE: York Portable Prolog

HARDWARE: This ES may run on several computers like the IBM-PC. There is an interface written to connect external inputs from the RS-232 and A/D facilities to Prolog.

STATUS

System: Experimental

Data used: NA

Performance rating: NA

Size: NA

FUTURE DIRECTIONS: The following areas will be investigated:

- fuzzy set theory and other decision-making techniques will be investigated to yield more rules that are more applicable to the continuous nature of the process variable
- prediction of unsteady state responses

3.1.4 LINKman [20]

DEVELOPER: Sira and Blue Circle Industries, United Kingdom

DOMAIN AND DESIGN GOALS: To design a control system for the cement kilning process using a linguistic rule-based ES approach

LINKman is a self-contained supervisory control system that may be used in a stand-alone manner or alongside existing instrumentation and control systems (refer to Fig. 4 for its architecture). It is designed to provide

- robust on-line closed loop control of complex multivariable processes at a supervisory level
- extensive user interfacing to aid in the day-to-day process operation
- a powerful and flexible information processing system that may record historical data and provide several management reporting functions

ARCHITECTURE: Refer to Fig. 4.

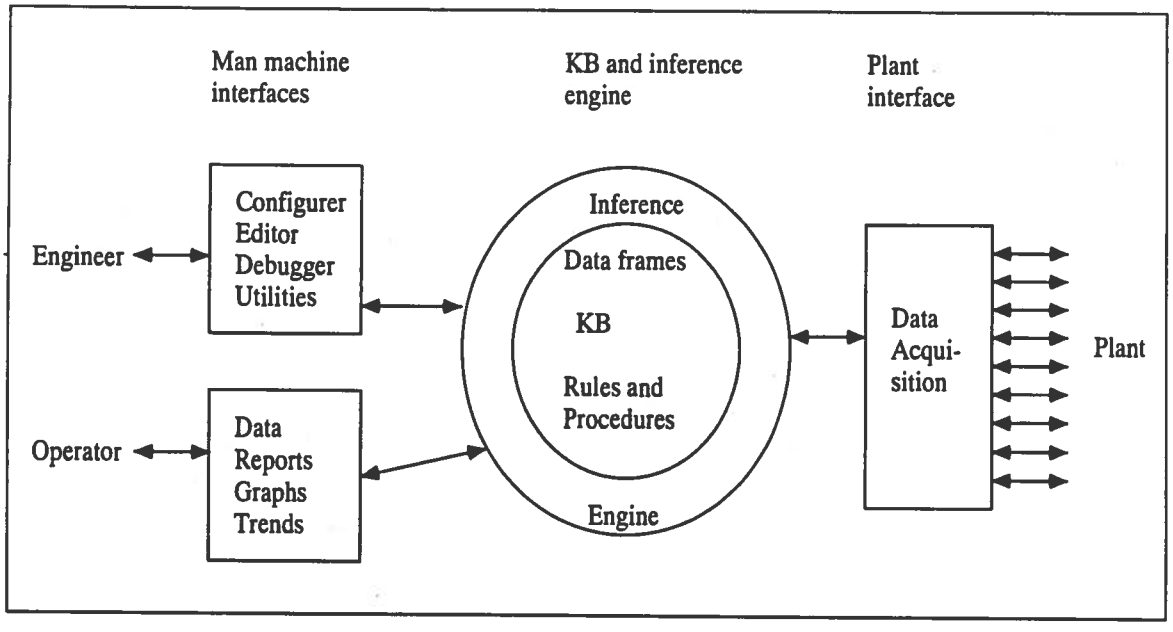


Figure 4: Architecture of LINKman

FEATURES

Knowledge representation: Two representation schemes are used: data frames and linguistic rules. Data frames describe plant items and their treatment. Information in the frames includes the naming of inputs and outputs, specification of engineering unit conversions, linearization algorithms, alarm limits, and other functions. Linguistic rules of the form *IF condition THEN action* with the *condition* and *action* parts qualified by a linguistic expression, like high, normal, low, and medium, describe the process by expressing information, such as operating decisions, procedures, timing,

and control relationships between plant conditions and output actions. Related rules are grouped into ruleblocks.

Working memory organization: Some 64K RAM is allocated to the short-term database.

Focusing mechanism: LINKman uses logical decision rules to select between different control schemes or ruleblocks.

Inference method: Simple forward chaining is used, guided by the outcomes of the logical decision rules.

Methods for handling uncertainty

Data uncertainty (input reliability required): Information in data frames may include a selection of predefined algorithms for treating I/O data. For example, noisy signals can be filtered or averaged. Also cross-correlation can be invoked with tolerance limits to detect faulty sensors.

Knowledge uncertainty: This implements logic loosely based on fuzzy logic and fuzzy set theory [Zadeh 65]. LINKman's treatment of uncertainty is more in line with the needs of process control than classical fuzzy logic.

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: All I/O is time stamped and facilities are provided for trend analysis.

Interrupt handler/asynchronous inputs: All data acquisition and keyboard handling is driven asynchronously under interrupt control.

Other interesting features:

- Slot filling exercises conducted via an engineering terminal were used to enter the information in the data frames. The Linguistic Control Language (LCL) which provides trace and debugging facilities and a number of display functions to aid in KA was used to enter the linguistic rules. The LCL is implemented as a two-pass compiled language.
- Some rule checking is done to ensure that rules do not refer to undefined items. Other than this, no automatic consistency checking is implemented (e.g., for conflicting rules).
- The use of LISP and Prolog for the development of the ES was considered, but they were discarded during the initial design stage because of the realtime operating constraint and the amount of floating point mathematics required.
- The use of DEC's RSX 11M operating system provides a full multitasking multiuser realtime environment.

LANGUAGE: All the software is written in C and MACRO.

HARDWARE: One configuration of LINKman runs on a PDP 11/73 under DEC's RSX 11M operating system with 4 Mbyte RAM, a battery backed clock/calendar, eight serial line interfaces, a 40 Mbyte Winchester, a 60 Mbyte tape backup. It is housed in a rugged enclosure, which makes it usable in very harsh environments. Data acquisition

from the plants is via a 64 channel Opto 22 I/O system. It is a remote, serially addressed sub-unit providing 32 digital and 32 analog channels per rack. To increase the I/O count in increments of 64 channels, these racks can be daisy chained.

LINKman is also now available on the DEC VAX range, running under VMS. It has a range of I/O and communications options including a stand-alone Opto 22 system, an open architecture and General Purpose Protocol, a number of commercial I/O drivers (including TCS), and special drivers can be written to meet specific requirements.

STATUS

System: Commercial. LINKman is being marketed by the Sira subsidiary, Image Automation, Ltd. In addition, Fuller Company, U.S.A., has been licenced to market this.

No. of installations: 13

Installation information:

- Blue Circle Cement, United Kingdom. An experimental software package was first developed (studies began in 1982) to run on Blue Circle's Kent P4000 Distributed Control System at its Hope Works in Derbyshire. This led to the final LINKman design which was installed on a trial basis to perform closed loop supervisory control and optimization at Blue Circle's Aberthaw Works in South Wales, early in 1985. As of November 1988, over 60% of Blue Circle's United Kingdom cement output is produced under LINKman control (six sites and 5 Mtonnes plus of capacity). The six sites are in the Aberthaw Works, Hope Works, Northfleet Works, Westbury Works, Cookstown Works, and Masons Works of Blue Circle Cement. The runtime of LINKman ranges from 70 to 90% at these six sites.
- British Petroleum, Llandarcy, South Wales. Controls a rotating disc column (liquid/liquid extraction) and its associated sub-processes on a lube oil production plant. It has been estimated that the payback for LINKman in this installation is less than 12 months.
- Pilkington Brothers, St Helens. The process being controlled here is a float glass line and the evaluation project aims to minimize energy and increase product yield.
- two other systems in California, U.S.A., and Obourg, Belgium

Comments: The following observations were made during the LINKman development process.

- A potential benefit to gain from the rule-based approach is the transportability of knowledge between similar but not identical processes.
- With the tools available in LINKman, process tuning is relatively simple and may be conducted by the process specialist.
- The control philosophy is visible, thus simplifying process management, with the use of linguistic rules. As an unexpected bonus, this has enabled Blue Circle to gain greater insight into the process and to improve general working practices.

- The system is well liked by the operators.
- Some economic benefits derived include: an estimated savings of several million pounds a year, fuel savings in the range of 5 to 10%, output increases of 10% plus, reduction of the average burning zone temperature by approximately 100 to 200degC, and improvement in product quality and plant stability.
- The authors believe that the technique used in LINKman is general enough to enable other processes, aside from cement manufacturing, to benefit from it. This has been proven with the application of LINKman in petrochemicals and glass manufacture.

FUTURE DIRECTIONS: A MicroVAX version will be available first quarter of 1989.

3.1.5 The Fault AnaLysis CONsultant (FALCON) [22]

DEVELOPER: The University of Delaware, E.I. du Pont de Nemous and Co. Inc., and Foxboro Co., Massachusetts

DOMAIN AND DESIGN GOALS: Designed as a research vehicle to provide a diagnostic tool for the process operator that monitors and analyzes alarm signals in a chemical process, specifically the formation of adipic acid in a commercial reactor. It aids the operator by converting an overwhelming amount of raw data into useful information.

Knowledge for the system is obtained from operations personnel and it is meant to be used by process operators and researcher/engineers with a separate interface.

The project has several goals. These are

- increase safety of industrial operations
- increase productivity
- determine feasibility of using an ES to detect faults in real time on an industrial process
- determine time and costs involved in building a KB for the expert system
- determine hardware and software resources for the ES
- determine how the system will respond to questions from operations personnel
- design the human interface for the process operator
- determine operator acceptance of the ES

ARCHITECTURE: The system is divided into five modules: supervisor, simulator, monitor, fault analyzer, and human–machine interface (see Fig. 5). The time-critical monitor module continuously examines process data and determines whether a disturbance exists; converts sensor data to a symbolic value (high, low, OK) which is sent to the fault analyzer; trends and rates of change are computed from data, converted and passed to the fault analyzer. The fault analyzer analyzes the alarm signals.

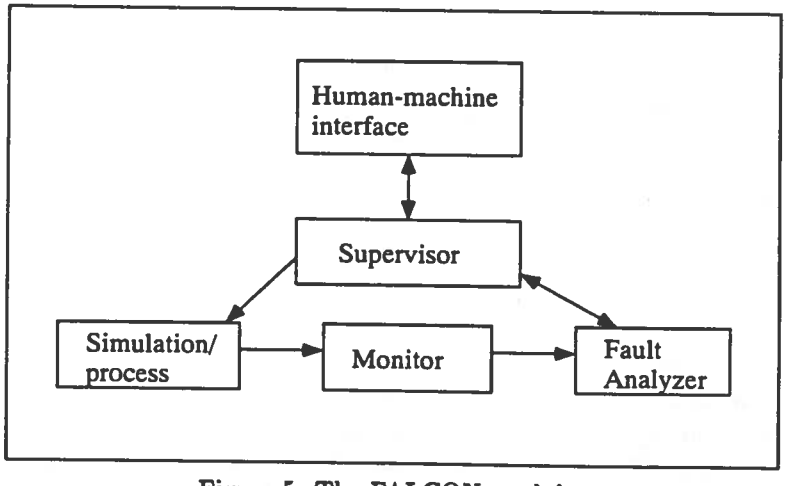


Figure 5: The FALCON modules

FEATURES

Knowledge representation: It uses *if-then* rules that apply first principles and heuristic knowledge. It has an input knowledge verifier that checks for missing information, syntax errors, and logical inconsistencies. At runtime it has a mechanism to check if a rule is flip-flopping, which may signify logical inconsistency.

Working memory organization: NA

Focusing mechanism: NA

Inference method: It is predominantly forward reasoning with some backward reasoning. It has two versions, one that reasons from observed data and another that examines a model of the process to find possible causes of the disturbance.

Methods for handling uncertainty

Data uncertainty (input reliability required): The inference engine checks to see whether or not the data are in range and determines if a sensor has failed.

Knowledge uncertainty: It has a way of rating hypotheses using the number of observed faults explainable by the hypothesis and the number of observed fluents (connections between components that are measurable quantities that would be represented by variables in a mathematical model of the system) inconsistent with the hypothesis.

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: The model of the process includes time delay information which allows intelligent diagnosis while the disturbances are spreading and allows the operators to act in order to avert potential disasters.

Interrupt handler/asynchronous inputs: The monitor module continuously scans and examines process data.

Other interesting features:

- It took 12 person-years (PYs) to create, but the authors are confident that with the available tools, it should take 3 PYs to redo.
- It has graphic tools to provide hard copy graphs of the process data.
- FALCON is able to identify incidents and give diagnosis, which at times were contrary to that given by the expert, with FALCON proving right in the end.
- FALCON provided greater insight into the process, which led to the addition of some instrumentation.

LANGUAGE: The fault analyzer is written in LISP and the human–machine interface is in Fortran.

HARDWARE: The system was developed on the VAX 11/780 under VMS and the test-bed runs on the same machine. It will, however, run on a Micro-VAX-II. The operator interfaces with the system using a color-raster CRT and a touch screen using an HP-2397A terminal. The interface between FALCON and the control system is via Ethernet.

STATUS

System: The current experiment with ESs has been completed. It was tested in the control room for a period of 4 months. R. Shirley of Foxboro is not certain on whether it is still on-line; but he said that it is unlikely that FALCON, in its current form, will be put in production. This is because FALCON works on only one specific process and was not designed for the industrial environment, efficiency-wise.

Data used: Simulation, taped process data, live, transmitted data, and actual tests in the control room.

Performance rating: It processes around 35 variables every 15 s. It takes approximately 7 s to perform an analysis.

Size: The fault analyzer consists of around 650 rules.

Concerns: The following concerns were expressed by the authors:

- high cost of deployment (took 12 PYs).
- significant costs incurred when refining and maintaining the ES after installation. This implies that the ES development tool must be sufficiently user-friendly that the process engineer can maintain the system after a modest amount of training.
- the use of LISP for an on-line realtime application. LISP does a good job of automatic memory management wherein it automatically cleans up unused memory. The cost, however, is that LISP periodically stops program execution to reclaim this unused memory causing the machine to sometimes go dead at random times until the garbage collection is completed. In the FALCON experience, overall process time including garbage collection is 75% of real time.

FUTURE DIRECTIONS: The parties involved in this experiment are very happy with the results obtained and even though FALCON is not now being used, it has proven the feasibility of using ES technology at Du Pont.

3.1.6 Materials Composition Management (MCM) [34]

DEVELOPER: FMC Corp. and Teknowledge, Inc., California

DOMAIN AND DESIGN GOALS: To design a process management system that applies heuristic control to manage the composition of ingoing materials, uses conventional analytical control equipment to maintain a number of physical parameters related to temperature and pressure under which materials react, and supervises both for optimal performance; applied to chemical manufacturing process control.

ARCHITECTURE: The MCM communications handler manages all communication between MCM and the outside world. The packet handlers time-stamp and unbundle the data packets received. The data handlers screen and store the data that can trigger the task handlers. The scheduler controls the execution of triggered task handlers. (Refer to Fig. 6.)

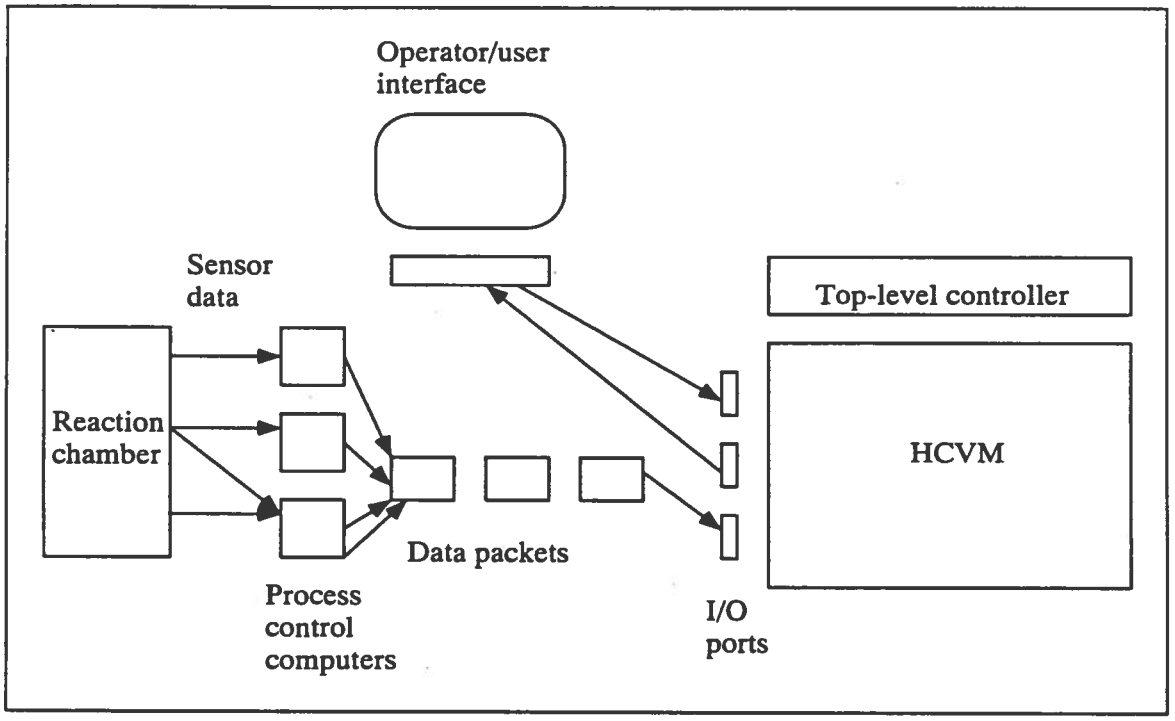


Figure 6: An overview of the MCM system architecture

FEATURES

Knowledge representation: Diagnostic and planning knowledge and data are encapsulated in a collection of modules analogous to the KSs in the blackboard model. These modules operationalize the tasks of process monitoring, situation assessment, action planning, plan execution and monitoring, and control management. The modules may be in LISP code, if-then rule sets, or another HCVM (see the section on domain knowledge for more information on an HCVM).

Working memory organization: NA

Focusing mechanism: HCVM is object-oriented where actions, procedures, and knowledge appropriate to an object are implemented as a separate object class.

Inference method: Event-driven

Methods for handling uncertainty

Data uncertainty (input reliability required): Data reliability is checked for consistency and plausibility using previous assessments and experiential knowledge.

Knowledge uncertainty: Conclusions have the belief levels no evidence, possible, or definite.

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: It contains some mechanism that examines history lists of time-stamped instances of time-varying data types.

Interrupt handler/asynchronous inputs: The HCVM architecture allows modules to be activated or scheduled for execution asynchronously (i.e., invoked by the recording of a data event). It also allows interrupts that abort task activities in favor of higher priority activities.

Other interesting features: NA

LANGUAGE: NA

HARDWARE: LISP machine

STATUS

System: Prototype

Data used: NA

Performance rating: NA

Size: NA

FUTURE DIRECTIONS: Future expansions to the prototype include

- allowing data objects to acquire unavailable data and notifying the module initiating the request when these become available
- having a way of determining a problem's importance when dealing with multiple problems
- developing hypothesis management and belief revision mechanisms
- representing more explicit knowledge about the data
- developing structures that permit multiple treatment tasks to coordinate their respective recommendations
- allowing dynamic planning in uncertain environments
- generate realtime response by
  - devising efficient event detection (possibly through parallelism)
  - providing for full interruptibility
  - exploring alternative methods of control

3.1.7 Diagnostic Event Analyzer (DEA) [10]

DEVELOPER: Laboratory for Intelligent Systems in Process Engineering, Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts

DOMAIN AND DESIGN GOALS: An automated system for diagnosing malfunctions in a continuous process. Its objectives include

- developing a system that has the ability to reason about multiple malfunctions
- ability to port large portions of the KB from one plant to another
- integrate causal diagnosis with that based on analyzing quantitative constraints

ARCHITECTURE: Refer to Fig. 7.

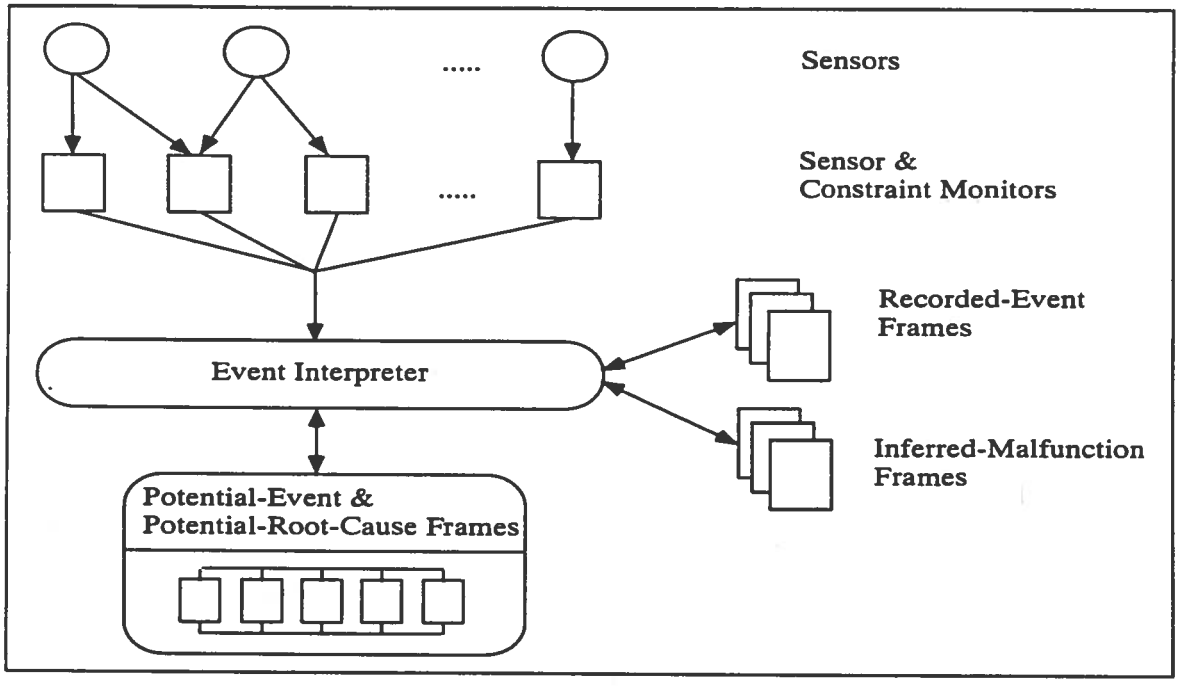


Figure 7: Overall architecture of DEA

FEATURES

Knowledge representation: The event interpreter consists of a small set of generic rules and LISP functions.

The other information used by DEA is represented by objects of different types implemented using the frame representation. Objects are related to each other forming a hierarchical relationship where objects in the lower layers of the hierarchy inherit attributes from the higher layer objects. A typical frame contains attributes that describe the properties of the frame, values for these attributes, and a set of facets that describe the domain of the attribute value. These form the declarative knowledge about the object. Methods and demons form the procedural knowledge about

the object. Methods are procedures associated with the object such as calculating outputs given the input values. Demons, on the other hand, are functions associated with a frame slot or attribute. These are triggered when the slot or attribute values change, such as a change in a critical sensor value.

Object types include

- monitors. There is a separate monitor assigned to each sensor, quality attribute, or constraint of interest. A monitor watches a particular measurement or performance measure and triggers the event interpreter should it receive a significant event. DEA has two types of monitors: EF (exponential filter) and SQC (statistical quality control).
- potential root causes. User-defined system malfunctions.
- potential events. These objects contain the information on the root causes of, and relationships between, events.
- recorded events. This object represents an occurrence of a potential event.
- inferred malfunctions. These represent a group of recorded events that can be attributed to a single malfunction.

The first three object types form the static KB containing information on failure modes, causes, and effects. The last two objects form the dynamic KB containing interpretation of observed events.

Working memory organization: NA

Focusing mechanism: The system is always focused on interesting events and only objects that are relevant to these observed events are considered.

Inference method: This uses object-oriented programming. The sensor and constraint monitors watch their assigned sensor or measurement. When the appropriate slot receives a measurement, the demon associated with this slot verifies that the value meets its constraints. If not, other slot demons are activated that send an interesting event to the event interpreter. The reasoning mechanism attempts to find a direct cause for the event, taking into account previous and succeeding events. It applies rules that locate local cause and precursor links. Rule application is via GoldWorks which uses the Rete match algorithm [Forgy 82]. If a direct cause for the event is found, the system eliminates previous hypotheses that conflict with this new evidence. Hypotheses that do not exactly fit in with the inferred cause but that cannot be eliminated with certainty are placed in the *other possible root causes* slot.

Methods for handling uncertainty

Data uncertainty (input reliability required): NA

Knowledge uncertainty: NA

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: The potential event objects contain causal information for events that may be observed in the system. This is used by the reasoning mechanism when diagnosing the cause of observed events.

Interrupt handler/asynchronous inputs: NA

Other interesting features: The system designers believe that DEA has successfully addressed the following points.

- It explicitly stores and makes use of the order of events.
- It addresses the problem of variation in event order (i.e., it can diagnose faults wherein the events manifesting these do not come in the prescribed order).
- It can diagnose multiple faults and malfunctions.
- It integrates quantitative constraints and qualitative measurement deviations.
- The inference mechanism is transportable.

LANGUAGE: DEA is implemented using GoldWorks.

HARDWARE: The system runs on an IBM PC/AT under MS-DOS using 6 MB of memory.

STATUS

System: Prototype

Data used: NA

Performance rating: The current implementation using LISP and an IBM PC/AT is found to be too slow for many practical applications. It is found to perform at 15–30 s/event.

Size: NA

FUTURE DIRECTIONS: Focus will be on the following issues:

- deriving the potential event objects directly from a process flow sheet
- building an English-like explanation facility
- integrating advisory functions into DEA that will make suggestions on repair actions following diagnosis

3.2 Examples of RTES Shells and Tools

The following are examples of shells or tools that aid in developing RTESs. Figure 8 summarizes their features.

Name	System status	High perf.	Guaranteed Resp. time	Focus mech.	Noisy data	Knowl. uncert.	Truth mtce	Temp./ Causal/ Spatial	Interrupt handler	Asynch. input	Compile env.	Garb.coll/ Archiving	Continuous operation
MXA	Experimental			✓	✓	✓			✓		✓		✓
HEXSCON	Experimental	✓	✓	✓		✓		T,S	✓	✓	✓		✓
Comdale/C	Commercial	✓		✓	✓	✓		T	✓	✓	✓	A (data)	✓
PDS	Testing	✓			✓	✓		T				A (data)	✓
Picon	Commercial	✓		✓	✓		✓	T	✓	✓		A (data)	✓
G2	Commercial	✓		✓		✓	✓	T,S	✓	✓		A (data)	✓
RTES	Commercial							T		✓			✓

Figure 8: RTES shells and tools

3.2.1 The MXA Shell [3]

DEVELOPER: SPL International, United Kingdom (under contract to the Admiralty Surface Weapon Establishment)

DOMAIN AND DESIGN GOALS: To design a tool to aid research into using AI technology to solve sensor fusion problems when integrating data from multiple military sensors

ARCHITECTURE: NA

FEATURES

Knowledge representation: A KS is a group of condition-action rules. It takes the form

KNOWLEDGE_SOURCE ks_name
IS INITIALLY ks_status
PRIORITY priority_num
...list of rules...
END_KNOWLEDGE_SOURCE

A rule, on the other hand, takes the form

RULE rulename
IS conditions
ACTION actions
END_RULE

There are metalevel KSs whose function is to prioritize and schedule the domain KSs. In addition to KSs and rules, built-in PASCAL functions may be used.

In the types of applications handled by MXA there may be several instances of things in the scene being interpreted, such as several planes or ships in the scene. The knowledge representation language therefore has a feature which implements pattern-matching to the conditions of the rules by set definition. When a rule is invoked, a set is formed consisting of all hypothesis structures that meet the constraints as applied to the current state of the blackboard.

Working memory organization: It uses the blackboard model where the blackboard entries are the hypotheses created. The system maintains creation time and time of last update for each entry. These entries are connected by binary evidential links to explicitly record the supports of a hypothesis. Each hypothesis available to the rules belongs to a predefined class. It has two parts:

1. the *hidden* structures which are used by the system to maintain control. This includes information on the likelihood, the time-stamp, and the link fields (to show relationships between hypotheses) of the hypothesis.
2. the *visible* part that consists of the user-defined attributes in a form similar to Pascal data declarations. These are the attributes that are directly accessible to the user.

There are two blackboards maintained. One blackboard contains the hypotheses made in solving the domain problem of interpreting the scene based on sensor input. The other is the control blackboard that contains plans on KS invocation.

Focusing mechanism: There are two mechanisms employed. The first makes use of metalevel or control knowledge, embodied in the meta-KSs, to guide the inference process. These meta-KSs have access to fields in the blackboard that are normally hidden, to control fields associated with KSs (the KS status and the priority fields), and to the internal scheduling mechanisms. The second focusing mechanism is the capability of the system of creating hypotheses that represent expectations which constrain the forward chaining via a generate-and-test strategy.

Inference method: Forward chaining (event driven) and backward chaining (goal driven)

Methods for handling uncertainty

Data uncertainty (input reliability required): Sensor data are entered into the blackboard as hypotheses, and uncertainty about these is expressed using likelihoods (as in other inferred hypotheses).

Knowledge uncertainty: Hypotheses have likelihoods attached to them, and evidential links have degrees of support attached to them. Lazy-evaluation Bayesian methods are used to propagate these likelihoods.

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: NA

Interrupt handler/asynchronous inputs: The control blackboard contains information on the current objective of the system and plans on how to achieve this. This objective may be interrupted to tackle some new higher-priority objective. When an interrupted plan is resumed, a review of previous steps performed is done to make sure that they are still valid.

Other interesting features:

- MXA double compiles the KB. First the MXA compiler compiles the source, which is in the MXA language, into one Pascal program which is then recompiled by the Pascal compiler of the host computer.
- User information or requests for an explanation are treated as hypotheses of *request* class and are put on the blackboard.
- Identified drawbacks and omissions in MXA are nonincremental compilation, lack of consistency checking of rules, and no property inheritance.

LANGUAGE: A knowledge representation language which is an extension to Pascal.

HARDWARE: The shell runs on a VAX with a BBC micro as the user interface to give color graphics.

STATUS

System: Experimental. It may receive data from a realtime simulator, from a file containing time-stamped data, or directly from the sensors.

No. of installations: NA

Locations of installations: NA

Installation information:

- A demonstration signal understanding system using a realtime simulator has been built.
- It is being used for experimentation in intelligent alarm analysis.

FUTURE DIRECTIONS: NA

3.2.2 The Hybrid EXpert System CONtroller (HEXSCON) [26]

DEVELOPER: SRI International, California

DOMAIN AND DESIGN GOALS: To design a tool that deals with control problems encountered in military and advanced industrial applications. Its design goals include

- having a capacity of 5000 rules
- using a microcomputer with 512 K memory
- having a response time of 10–100 ms
- being able to handle 1000 objects
- being able to function with uncertainty

ARCHITECTURE: Refer to Fig. 9.

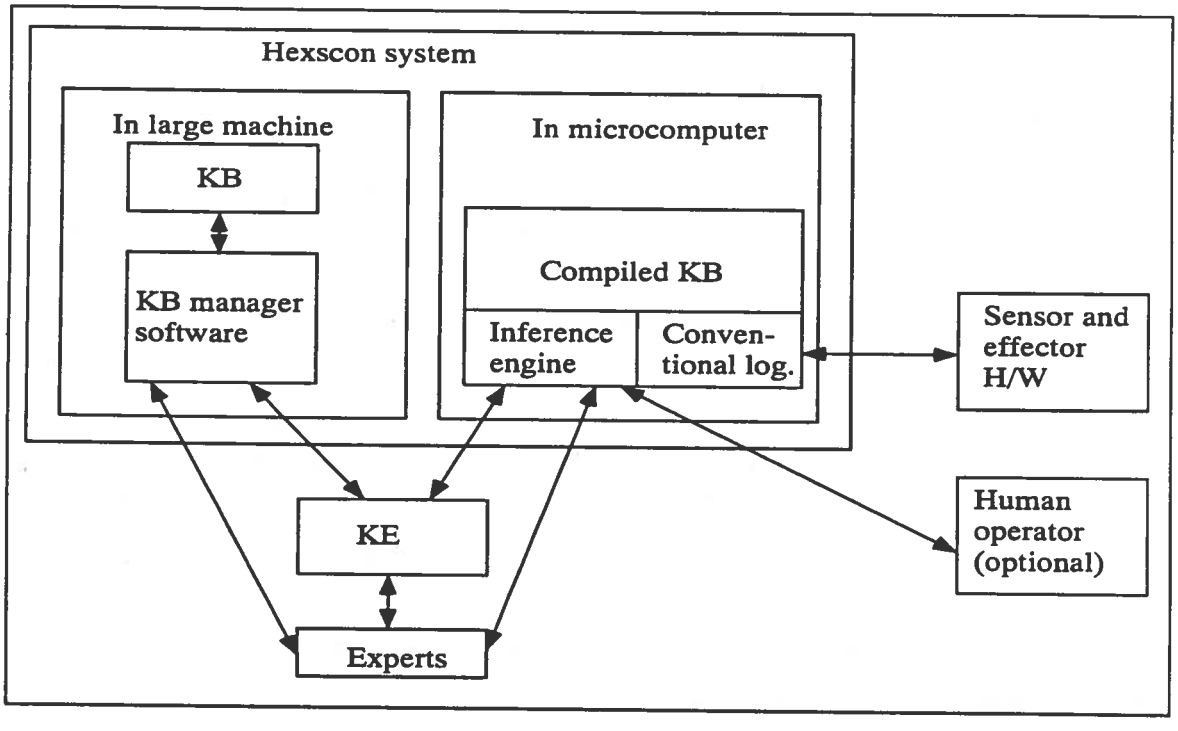


Figure 9: Architecture of HEXSCON

FEATURES

Knowledge representation: The system uses two types of knowledge: one embodied in if-then rules and the other using conventional knowledge. There is a translator that interfaces these two parts.

Working memory organization: Uses object planes identified by an object plane number composed of the object number and a time-stamp (see Fig. 10 and the section on WM).

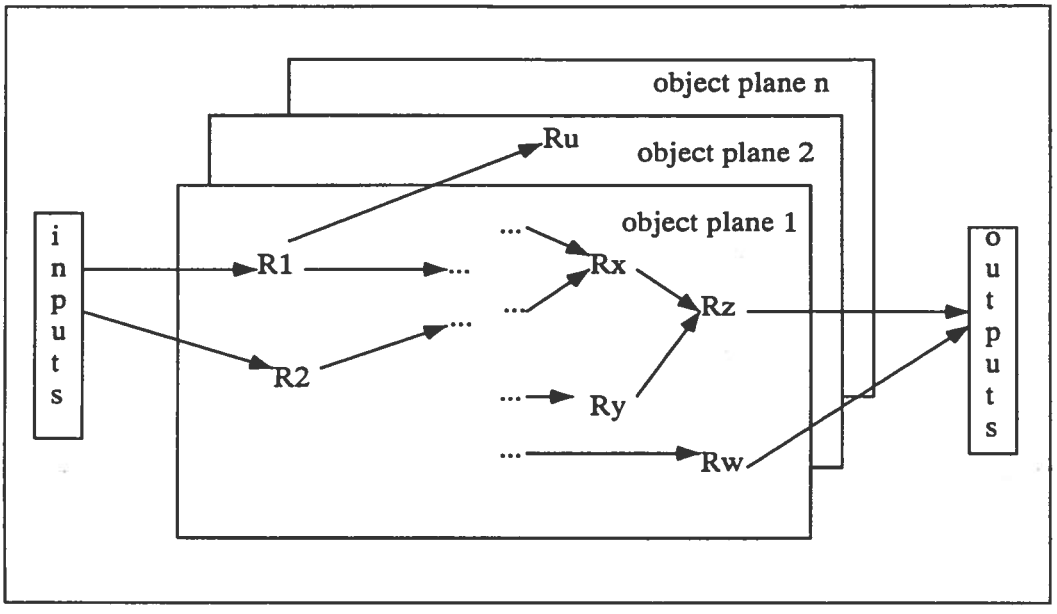


Figure 10: Logical representation of multiple objects in HEXSCON

Focusing mechanism: It can focus attention on certain areas by using the control knowledge which controls how the inference engine operates on the heuristic knowledge and by using prioritization.

Inference method: It uses a technique called progressive reasoning that allows reasoning at different levels, depending on the response time required. This response time may range from reflex-like responses of 10 ms to responses requiring complex decisions of 20 min. The system currently has four levels. The first level is for the conventional realtime operating system. The second to the fourth levels are for the KB part. After each level, the system determines if there is sufficient time to go on to the next level. If this is the case, it goes and does reasoning on the next level, otherwise, it gives the response as determined at the current level. Different levels use different types of knowledge.

In addition, it uses forward chaining with backward chaining.

Methods for handling uncertainty

Data uncertainty (input reliability required): NA

Knowledge uncertainty: Multiple lines of reasoning are combined using the Schafer-Dempster theory to combine evidence. Uncertainty is represented using the three attributes belief, confidence, and importance. Belief and confidence distinguish between uncertainty and lack of information. Importance indicates the relative importance of facts in a rule. These three are used to help in rule selection. It uses two types of inferencing to propagate the inferencing parameters, belief and confidence. The first, linear inferencing, propagates inferencing parameters through a rule using a fixed algorithm. The second, nonlinear inferencing, propagates parameters that are not necessarily related in a fixed way to the condition

parameters of a rule; but instead, are set by values in the rule itself (refer to [26] for more information).

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: Object planes may reflect the situation in the past, present, or future. Past and future planes are stored on disk and the present object planes are kept in memory. These are used to handle basic time and spatial reasoning.

Interrupt handler/asynchronous inputs: The KB part runs as a single task in a real-time system with two types of interrupts. The first is the process interrupt which is equivalent to an interrupt in conventional realtime operating systems. The second is the logic interrupt which is an interruption of the KB component part way through its reasoning process.

Other interesting features: NA

LANGUAGE: Pascal

HARDWARE: The shell runs on a microcomputer.

STATUS

System: Experimental. It may operate in simulation mode which is used in KB development and realtime control mode which is used when running the tests.

No. of installations: NA

Locations of installations: NA

Test cases:

1. Goal of obtaining maximum squeeze with minimum sophistication. This used an 8-bit micro (8085) with 64 K memory, around 2000 (1650 actually used) rules. Response time was 10–30 s.
2. Goal of obtaining maximum speed with moderate sophistication. This used a 16-bit micro (8086) with 250 K memory and around 5000 (250 actually used) rules. Response time was 0.25–0.5 s.
3. Goal of obtaining maximum expertise with maximum sophistication. This used a 16-bit micro (8086) with 512 K memory and around 4000 (350 actually used) rules. Response time was 1–10 s.

FUTURE DIRECTIONS: NA

3.2.3 Comdale/C Process Control System [37]

DEVELOPER: Comdale Technologies Inc., Toronto, Ontario

DOMAIN AND DESIGN GOALS: To provide a tool for the development and application of ESs to process control. The design goals are to offer flexibility in knowledge representation and explanation, ease of use, expandability, and transparency of knowledge.

Knowledge is acquired from the operators.

ARCHITECTURE: It is composed of several modules: controller, database, operator, display, scheduler, alarm, and expert, which may be configured by the developer. The controller is the inference engine. The database is the working memory. It possesses a data manager that manages the supply and retrieval of data by the other modules in the system. The operator module provides the interface with the operator. The display module provides user-configurable screens to display variable values. The scheduler takes care of managing time-dependent tasks, such as logging variables and executing tasks at specified times. The alarm module monitors alarm states of all types of variables. The expert module coordinates all requests for access to any of the other modules.

FEATURES

Knowledge representation: Uses heuristic rules in if-then form. A rule trace facility and the ability to set break points within rules are used to test the KB.

Working memory organization: A database of assertions. An assertion is a keyword-triplet composed of object, attribute, and value tokens.

Focusing mechanism: Library functions and control units exist to help focus the system. For example, there is the *apply rule function* that focuses the system on certain rules when a certain situation arises, like a sensor giving a certain value. Attributes are associated with rules in order to determine the situations they are interested in. Also, there is the *scan control unit* that specifies a list of rules to execute for a given situation.

Inference method: Forward and backward chaining are both used. The developer writes the rules in a certain way to enable the forward and backward chaining process and, depending on the data and information, the inference engine will use either process.

Methods for handling uncertainty

Data uncertainty (input reliability required): The Comdale/C library of functions includes one that ignores or filters out noisy data.

Knowledge uncertainty: Each keyword-triplet in working memory has a *degree of certainty* associated with it. A mechanism for defining fuzzy values for keyword-triplets is provided and these are used when determining the degree of certainty for this. Each rule premise has a *degree of truth* associated with it that is calculated from the degree of certainty of the associated keyword-triplet plus the predicate used in the premise. For instance, the degree of truth of a premise

using the predicate *is* may be different from that of a premise with the same keyword triplet but using the predicate *is definitely*.

The *net degree of truth* for a rule is calculated by first determining all the degrees of truth of each premise. Then, depending on the logical operators used, the net degree of truth may be the minimum degree of truth of the premises for an *and* operator or may be the maximum degree of truth for an *or* operator.

The rule is fired if the net degree of truth exceeds a specified confidence level. The degree of certainty associated with a concluded keyword-triplet depends on the predicate in the conclusion, the net degree of truth for the whole rule premise, and the certainty factor specified in the rule for this conclusion. For example, if the predicate is *is* then the concluded triplet gets a *degree of certainty = net degree of truth * (confidence factor specified for this concluded triplet / 100)*.

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: It deals with lags in operator decision-making and control action (when one has to wait to make adjustments to control actions to avoid overcompensating adjustments or masking their effects) by having the capability of scheduling future events.

Interrupt handler/asynchronous inputs: The alarm module monitors the values received by the database module to check for alarm conditions. If the developer has provided rules to deal with these alarm conditions, these rules may in turn interrupt the controller module. It is not clear how this interruption is done.

Other interesting features:

- The shell may be embedded in another application.
- The explanation facility uses customized questions and explanations. A decision tree is built to show the rules that have been applied. In addition, the developer may use various Comdale library functions like Paint to produce graphics. These are stored in files that can be called from the KB, thus enhancing the explanation given to the user.
- The KB is compiled and a cross-reference is produced to aid in knowledge verification.
- Comdale Technologies offers a four-day training course on expert systems and the Comdale/X shell (the off-line version of Comdale/C) and it was found that, after this course, the students can build some pretty good prototype systems.
- Examples of process control functions handled by the shell include handling alarms, maintaining setpoints, and halting processes when necessary.
- The largest rule set built is composed of around 500 rules. It is found to make decisions slightly ahead of the operator. The decisions made by the system are found to correspond to the operator's decisions.

LANGUAGE: It is written in C.

HARDWARE: The tool runs on an IBM PC/AT and 386 compatibles under XENIX (which is being phased out) or QNX (and DOS for the off-line version, Comdale/X). There is an interface to the Bailey network 90, Foxborough I/A (using the X25 protocol), and Optomux (using RS-422).

STATUS

System: Commercial

No. of installations: NA

Locations of installations: Polaris Mine, Cominco Ltd., North West Territories, Wabush Mines, Labrador

Installation information:

- Polaris Mine. As of December 1987, development of the Polaris Expert is near completion. The operator interface uses a VT-220 terminal and a Roland 1012 printer. The plant interface is via an RS-232 and RS-422 serial communications with a PDP 11/23 and an Opto-22 data acquisition hardware. The ES runs on a Compaq Deskpro 286.

  The PDP 11 outputs %solids and assays from seven process streams and seven flotation air monitor readings that are updated every 6 min. The Polaris Expert acquires data from 100 sensors with the operator entering some data, such as froth color.

  The Polaris Expert has over 530 input, output, inferred, and concluded variables. History on these variables are kept for up to 3.5 days of plant operation. It contains over 400 rules and requires 800 KB of RAM. The scan cycle (period when the controller looks at the plant and makes decisions) is 2 to 5 min.

  The first version of the Polaris Expert runs off-line (i.e., it does not control the flotation circuit. Instead, it gives suggestions on control settings to the operator). It is currently being fine-tuned and the KB modified to run on-line.

- Wabush Mines. This is a RTES that controls the grinding circuit of a semi-autogenous grinding mill. It is currently being developed and it is estimated that the final version will have around 100 rules. This RTES interfaces with a Foxboro 7621 (the interface is written in C). It receives data from 15 sensors and sends data to seven actuators. This implementation runs under QNX and is scheduled for testing on May 1, 1989, with possible implementation in June 1989.

FUTURE DIRECTIONS: For the immediate future, Comdale Technologies is working on an on-line graphics package for Comdale/C. It has been found that the current package is adequate and no extensions to the system are planned.

3.2.4 Process Diagnostic System (PDS) [32]

DEVELOPER: Carnegie-Mellon University and Westinghouse Electric Corporation, Pennsylvania

DOMAIN AND DESIGN GOALS: A rule-based architecture for the on-line realtime diagnosis of malfunctions in machine processes. A goal for this includes providing machine technicians with a portable and inexpensive diagnostic tool.

ARCHITECTURE: NA

FEATURES

Knowledge representation: It uses rules organized into an inference network. PDS has several special rule types, such as belief and parametric alteration rules. Belief rules are used by the inference engine to propagate belief. Parametric alteration rules are used to alter the definition of any node or rule in the system.

Working memory organization: Hypotheses, sensor readings, and malfunctions are represented as nodes in a network. The node contains information, such as the level of belief in the node being true (MB), the level of disbelief in the node being true (MD), the level of certainty (CF) for the node, which may be a combination function of MB and MD, and other information pertinent to the node type. For example, for a sensor node there are slots for the reading-value and the reading-time that store the current reading for the sensor.

In addition, there are slots to show the links from and to the node. For example, there is a slot for supporting rules which specifies other rules for which this node is the hypothesis; and there is a slot for supported rules which specifies rules for which this node is evidence.

Focusing mechanism: NA

Inference method: Forward chaining

Methods for handling uncertainty

Data uncertainty (input reliability required): PDS attempts to reason about its sources of information and their veracity. It handles two issues that arise in the application of ESs to the analysis of sensor-based data and these are spurious readings and sensor degradation. Techniques used in PDS include retrospective analysis to handle spurious readings and metadiagnosis to handle sensor degradation.

In retrospective analysis, readings from a sensor or values for any node are stored and analyzed to derive ways to refine the rule base. Types of analysis include time series analyses, such as rate of change (first derivative), averages, filtering, and curve smoothing. These analyses are done both at the front end of diagnostic systems and during the diagnosis itself.

Sensor degradation is handled at the machine level by the placement of redundant and overlapping sensors. At the diagnosis level, PDS uses metadiagnosis to detect sensor degradation. It uses rules to monitor the behavior of a sensor.

If it is sensed that the sensor is malfunctioning, it uses parametric alteration rules attached to the corresponding sensor schema to modify this by reducing its weight in the conjunctive evidence of a belief rule.

In addition, in order to analyze and react to readings from redundant, overlapping sensors, PDS provides the composite sensor schema which combines multiple sensors into a single composite sensor.

Knowledge uncertainty: If a rule has a compound antecedent or condition, the belief rules are used to determine the MB, MD, and CF of the resulting hypothesis or action. Information in the belief rules are used to combine the MB, MD, and CF of each of the constituents of the rule condition in order to derive the MB, MD, and CF of the resulting action or hypothesis. This method of belief propagation is similar to the one used in Mycin [Shortliffe 76].

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: Some trend analysis is done on monitored data.

Interrupt handler/asynchronous inputs: NA

Other interesting features:

- Its explanation facilities are primitive. It uses canned text.
- It can be connected to a text-to-speech converter (the PROSE 2000 board by Telesensory, Inc.), allowing it to *speak* its explanations.

LANGUAGE: PDS is written using the Schema Representation Language (SRL) which is implemented in FranzLISP.

HARDWARE: PDS was developed using a VAX 780 under VMS. A version of this tool is being implemented on a four-board package (CPU, memory, graphics I/O, and text-to-speech converter) that fits in a hand-carry suitcase.

STATUS

System: Testing

No. of installations: NA

Locations of installations: Westinghouse Power Generation, Orlando, Florida

Installation information:

- Prototype. It started with 10 sensors, 44 rules, and 29 intermediate hypotheses to indicate seven malfunctions. It took one month to develop and test. Testing was done using 150 sets of test data generated in four groups. Group 1 tested the rule interactions. Group 2 tested the response of the system to malfunctions previously diagnosed by the experts. Group 3 tested the response of the expert to the malfunctions diagnosed by the system. Group 4 comparison-tested the responses of the experts and the system to data neither had *seen* before. Problems encountered in this phase included
  - convincing the necessary experts to participate.

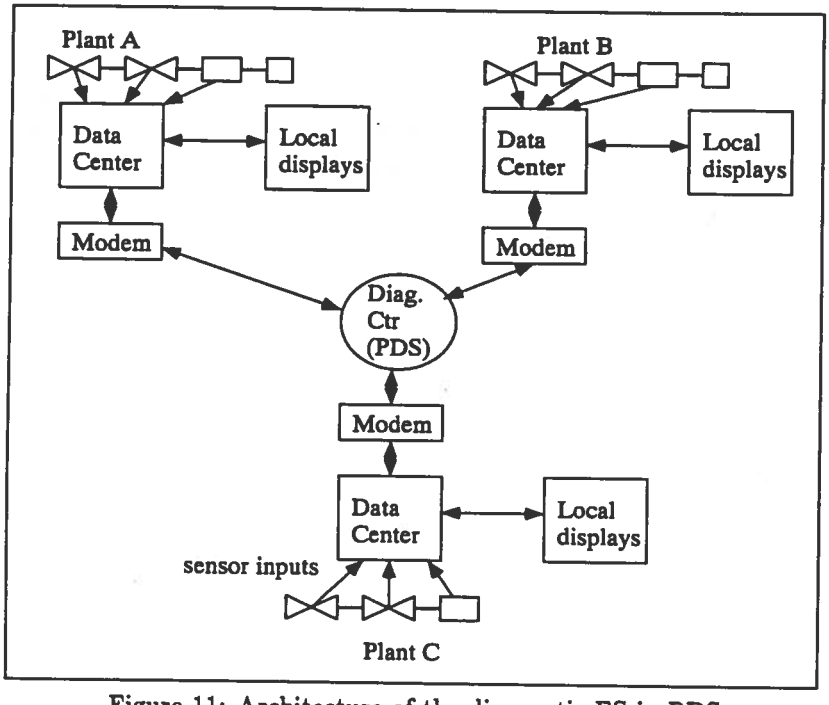


Figure 11: Architecture of the diagnostic ES in PDS

- experts had a hard time thinking about the steps they took to make diagnoses. They usually went from a sensor reading to a final malfunction in one step.
- the actual gap in the knowledge of the experts. The knowledge of sensor malfunction and the forms it may take is extremely limited.
- the sensors used for the on-line test malfunctioned more often than desired and so it was difficult to obtain data to test anything but the malfunction detection rules.
- the disagreement among the experts on certain rules of diagnosis.

- A diagnostic expert system that is presently remotely diagnosing from the Westinghouse Power Generation headquarters in Orlando, Florida seven steam turbine generators at the Texas Utility Generating Co. The architecture for this system includes a centralized diagnostic center (developed using PDS) interfacing with multiple data centers (refer to Fig. 11). The data centers transmit data from the turbine generator to the diagnostic center and take care of displaying trends of any points being monitored versus time as well as other monitored points. They maintain databases that store information for up to a week. The centralized diagnostic center translates sensor input to a diagnosis and recommendation for corrective actions. It has the ability to discriminate false alarms from real emergency as seen from several incidents. It can store information in its database for periods longer than a week.

  The diagnostic expert runs on a VAX 11/780. For data acquisition, a Computer Products type RTP data acquisition system tied to a VAX 11/725 processor is

used. The operator interface uses a Professional 350 personal computer and color touch screen CRT.

Readings from around 110 different sensors in a gas-cooled generator and auxiliaries are transmitted through telephone lines or satellite to the diagnostic center. The system is able to identify 350 conditions using 1300 rules.

FUTURE DIRECTIONS: Future plans include improving the explanation capabilities, both in the language generation aspect and in the range of questions it can answer.

3.2.5 Process Intelligent CONtrol (PICON) [15]

DEVELOPER: LISP Machines, Inc. and GigaMOS Systems, Inc., Massachusetts (vendors)

DOMAIN AND DESIGN GOALS: It is a RTES tool for developing process control applications with the following requirements: high speed, context-sensitive rule activation, efficient recycling of memory elements no longer needed and maintenance of sensor history, interactive acceptance of command sequences from the operator, and communication between multiple experts.

ARCHITECTURE: PICON has four parts: the inference engine, AI-BASE, the knowledge base, and the data supplier. The inference engine applies the rules and requests for data that it needs. The AI-BASE is the interface with the engineer who enters, revises, and edits the knowledge in the ES. The KB contains the two types of knowledge: the schematic diagram and a rule base. The data supplier supplies sensor values to the inference engine. During on-line operation this is usually RTIME. During testing this may be the PICON simulator. In some applications this may be some other type of data supplier.

The inference engine, the KB, and AI-BASE run on the Lambda LISP machine. RTIME runs in the UNIX portion of Lambda/Plus and this receives and sends data to the distributed process control system.

FEATURES

Knowledge representation: In PICON, a schematic diagram is used to describe the layout of the plant. Each item in the diagram is taken as an object with an associated attribute table.

In addition, heuristic knowledge is represented using rule frames which combine the rule and frame schemes. PICON supports different types of rules, such as

- if-then rules whose action part is carried out if its conditions are true
- unconditional rules (i.e., they do not have an *if* part) which carry out actions whenever they are activated, either by backward chaining or because they are associated with some attribute
- initial rules perform actions when a KB is first run or when it is restarted after being stopped with a halt command
- whenever rules test whether a value varies by some specified amount and executes its actions whenever the new value goes outside the specified limit

When these rules are created an associated attribute table describing them is also created. This contains attributes that relate to each rule, such as problem type, process unit, or category, dynamic attributes, such as certainty and data currency, and attributes relating to the use of the knowledge, such as the scan interval and priority.

To aid in testing the KB, PICON provides a way to retrieve knowledge by attributes associated to a heuristic frame or schematic object. It also provides a simulator that

takes user-defined simulation statements, which are entered in the same way as the rules. When the user specifies that he or she wants to receive data from the PICON simulator, the system uses these statements to simulate sensor values.

Working memory organization: NA

Focusing mechanism: PICON may operate in two modes. First, while in normal mode, the system scans key processing information to monitor performance and to detect problems. This is done by firing rules that determine possible significant events. When a problem is detected, the system goes into the second mode called the alert interval. It will focus on the detected problem, and logic rules and procedures are invoked to diagnose this.

Inference method: PICON uses forward chaining when initial data are plentiful, the number of possible conclusions are large, and all the hypotheses must be deduced. Backward chaining is used when data acquisition is expensive and when one or more hypotheses must be validated. It allows prior scheduling of events through either the event scheduler or the cycle system. The event scheduler initiates specific activities at specified times in the future; whereas the cycle system initiates activity at regular intervals. It is loosely procedural. It does not use pattern-matching to decide which rules to apply; instead, rules have explicit suggestions as to what rules apply to certain states or a database query facility is used to retrieve rules that apply to the current situation.

Methods for handling uncertainty

Data uncertainty (input reliability required): If a sensor fails to provide data on demand, the system repeatedly tries to get this value for several minutes; then it passes control back to the sensor frame (in the KB) and performs the actions specified in the slot *on data supplier problems.*

Knowledge uncertainty: NA

Consistency maintenance mechanism: Sensor variables are assigned a currency interval which is the length of time after a sensor is read that the reading is considered valid. After such time, this reading is considered expired and will have to be reread before it can be used.

Temporal/causal/spatial reasoning: Variables are time-stamped and rule syntax allows referrals to time intervals, thus allowing the initiation of specific activities at specific times in the future.

Interrupt handler/asynchronous inputs: During on-line operation data are usually supplied by RTIME which runs on a separate processor from PICON. RTIME takes care of acquiring data on a timed basis. It may perform parallel processing on accessed data using algorithms specified by PICON. When it observes a significant condition (as specified by PICON), it (software) interrupts PICON.

Other interesting features:

- PICON provides a schematic capture tool that allows the user to build a process schematic, which is a very familiar starting point for process engineers, using

icons. This permits P and I diagrams to be quickly entered with a full sensor database, which is a considerable aid in the initial stages of knowledge acquisition.

- Rule entry is also interactive using menus which lead the engineer through the filling out of the attribute list associated to the rules.
- PICON has reasonably extensive facilities for historical data archiving and retrieval. However, only values of variables are kept.

LANGUAGE: PICON is written in LISP and RTIME is written in C.

HARDWARE: PICON may run on the TI Explorer, LMI Lambda/Plus, and it was being extended to work on a VAX under VMS. RTIME operates on a 68010 processor that is running UNIX. RTIME interfaces with the distributed process control system via RS-232 type or Ethernet type links. It interfaces with PICON on the LISP processor via the serial STREAMS interface and shared memory.

STATUS

System: Commercial

No. of installations: NA

Locations of installations: Texaco Chemicals, Port Arthur, Texas (May 1985), pilot plant of Johnson Controls (August 1985), Exxon Chemicals, Bayway, New Jersey (September 1985), Oak Ridge National Labs., MIT Chemical Engineering Department, Lockheed, Rockwell, Leeds and Northrup, Bell Labs, etc.

Installation information:

- APEX, an alarm management RTES in a coal-fired power station at Thorpe Marsh, United Kingdom. It monitors operations of the coal milling plant and associated boilers of the station. The control system is fully digital and consists of a series of PDP 11 computers running control programs written in a proprietary language called CUTLASS. The plant has some 100 control variables, 200 computer-based alarms, and 100 hard-wired alarms. The final KB had 1600 statements of which 600 were if-then rules.

  The group who implemented this system give the following comments.

  - PICON provides many but not all of the features and facilities that are required for realtime process management.
  - It is a tool that is easy to use for non-AI specialists.
  - Its weaknesses include the lack of a class structure or inheritance mechanism when representing sensors, even though these appear to use some frame or object-based representation scheme. Also, the attributes associated with process icons have little or no built-in knowledge about the functions of the item being represented by the icon.
  - Software quality assurance issues must be satisfied before ESs can be used in a critical role [McClelland 88].
  - There is a question on whether ESs have enough bandwidth to handle the large amounts of data produced by a large system [McClelland 88].

- The SICON prototype built by LMI. Its objective is to illustrate PICON's applicability to the electrical-power-system task and specifically, to provide intelligent satellite control. SICON consists of dual solar arrays with associated positioning sensor, steering system, and power-monitoring devices. The system uses redundant sun-sensing devices to target solar arrays.

  It has several groups of rules. One group checks sun-sensor integrity. Another group implements standard steering control for solar arrays. A third, using meta-diagnostics, determines whether both arrays are operating, or if any array has failed. There are other rules that deal with battery charging and reconditioning.

  This prototype uses data from the telemetry tapes of a functioning satellite generated by either the satellite itself or a highly accurate ground simulation program.

  SICON provides communication between multiple ESs. Cooperating PICONs are organized hierarchically with a master supervisory system providing oversight and summarized information regarding the operation of the component ESs. It uses the MOBY technique to provide virtualization of globally consistent memories of unlimited size by software and firmware functions invisible to the users. The PICONs may regard variables within another PICON as sensor values and perform inferencing tasks based on these.

FUTURE DIRECTIONS: To handle applications such as plant optimization, overall plant and corporation scheduling, plant design, and other high-level planning.

3.2.6 Gensym Real-Time Expert System (G2) [30]

DEVELOPER: Gensym Corporation, Cambridge, Massachusetts

DOMAIN AND DESIGN GOALS: A tool for designing ESs that monitor and control complex realtime functions.

ARCHITECTURE: Refer to Fig. 12.

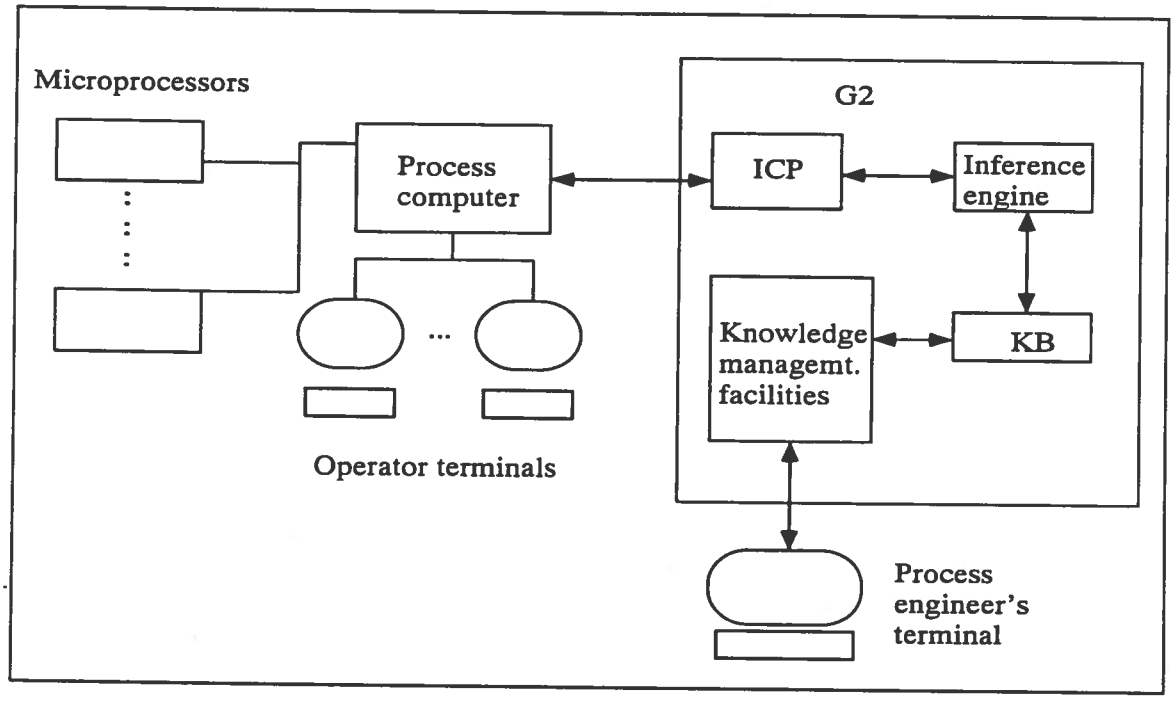


Figure 12: Structure of G2

FEATURES

Knowledge representation: Knowledge about the application is described in G2 using objects, rules, and formulae. Each entity of interest is described by an object and its associated attribute table. A schematic diagram composed of objects and connections between them is used to describe the model of the application. The connections, be they pipes, wires, or some abstract relationship between the objects, are taken as separate objects with their own attribute table.

Rules are used to embody an expert's knowledge of what conclusions to draw from conditions and how to respond to these. Each rule has an antecedent that lists the conditions when the rule is applicable, a consequent that lists what action(s) to take given the condition, and an attribute table that contains attributes like its scan interval, what objects it is interested in, its category, etc.

A third form of knowledge representation uses formulae to describe the algorithms to use when calculating values of variables. Variables are attribute values found in the attribute tables describing objects that are taken as special kinds of objects with their

own attribute tables. In the attribute table of a variable, one may find information such as what data source to use when determining the value for the variable, what formula to use to calculate its value, the last recorded value of a variable, and its default update and validity intervals. Formulae found in the attribute table of a variable are called specific formulae. G2 also has another kind of formula which they call generic formulae which applies to a whole class of variables.

Working memory organization: Values for variables or for objects that may have values are stored in two databases. There is the simulation database which contains all values calculated by the simulator and the inference engine database which contains current variable values and all inferred values. The simulation formulae are continuously being run and the values in the simulation database are continuously being updated. When the value for a needed variable is expired, a new value is obtained from its specified data source; which may be from the simulation database, the inference engine database, or from other data servers connected to, for example, a sensor or a process control computer.

Focusing mechanism: See the focusing and scanning methods used by the inference engine to invoke rules.

Inference method: The inference engine can invoke a rule by

- scanning. This is a method for monitoring conditions in a plant or other application by regularly invoking rules to evaluate the conditions. This is accomplished by specifying a scan interval in the attribute table associated with the rule.
- focusing. This is a method for concentrating on a particular object or class of objects by invoking all rules associated with that (class of) object(s) (i.e., all rules with this as its focal object or class as specified in its attribute table).
- invoking. This is a method for concentrating on categories of rules by invoking all the rules associated with a category (i.e., all rules with this as its focal category in its attribute table).
- backward chaining.
- forward chaining.
- event-driven activity. One form of G2 rule, the whenever rule, is invoked when its conditions are satisfied; for example, when a sensor variable gets an unrequested value from its data server.
- activating an object. If an object has an activatable workspace associated with it that may contain items such as rules, formulae, and other objects; this object may be activated when all items in its workspace are to be considered and be made inactive at all other times.
- wakeup. When an invoked rule cannot be immediately evaluated, for example, there is no currrent value for a variable; the rule is put to sleep and is awakened when a value for the variable is received. At this time the rule tries to complete again.

Methods for handling uncertainty

Data uncertainty (input reliability required): NA

Knowledge uncertainty: Variables may have fuzzy truth values which indicate the degree of certainty in the truth of the value of the variable attribute. A simple combining function is used to propagate the truth value of a rule's antecedent to its consequent. For example, if the antecedent is a simple logical expression without any connectors such as *and, or* and *not*; then the conclusion takes the truth value of the antecedent. If the antecedent is a compound expression that uses *and* connectors, then the compound antecent and the conclusion take the minimum truth value of the expression.

Consistency maintenance mechanism: To handle data that are not durable in time, G2 associates a time stamp and a validity interval to variables (which is how data received from the outside are stored). The time stamp gives the time at which the data were received and the validity interval states the duration for which this data are valid. Once the data value has expired (or its validity interval is exceeded), this is not used by the inference engine. It, instead, requests a new value for the variable.

In similar fashion, information inferred by the inference engine that may also be stored as variables, also have time stamps and validity intervals and are treated in much the same way as variables associated with external data.

Expiration times may also be propagated through the inference process. For example, given that a conclusion is reached based on several time-sensitive variables, it may take on the earliest expiration time of these variables.

Temporal/causal/spatial reasoning: Variables are time-stamped and rule syntax allows referrals to time intervals thus allowing the initiation of specific activities at specific times in the future. Types of expressions that may be used by G2 include history expressions and animation expressions.

In history expressions, the average value of a variable, its rate of change, its value, integral in seconds, or interpolated value may be checked for a specified time (for example, during the last 10 min, between 1 and 3 h ago, during the last 2 min, etc.); thus allowing some form of temporal reasoning.

Animation expressions refer to the position of an object, its height, its width, degree of rotation, and its location relative to other objects; this allows some form of spatial reasoning.

Variables are assigned a validity interval which is the length of time for which a variable's value is considered valid and after which time a new value must be obtained if the variable is accessed.

Interrupt handler/asynchronous inputs: In G2, asynchronous events, such as receiving unrequested sensor values, are handled by using *whenever rules* (refer to event-driven activity under inference method).

The G2 scheduler controls the events in G2. It takes rules to be fired and divides them into component tasks which are very small and take little time to execute. While G2 is executing a task, this cannot be interrupted, even by a higher priority task. Each

task, however, is given a certain amount of time to complete. If it has not completed by the end of this time, it is deferred.

Other interesting features:

- G2 provides two parameters that control how it keeps time and how it monitors events in time. For example, the user may force G2 to run in real time, in simulated time (where a simulated second may be longer than a second in real time), or as fast as possible (which, depending on the G2 application, may run faster or slower than real time). To monitor events in time, the user may force G2 to give priority to mouse and keyboard interface or to make the keyboard, mouse, inference engine, G2 simulator, and other data servers receive equal shares of the available time.
- As mentioned in the types of rule invocation, a rule may be put to sleep should it require a value that is not currently available. G2 provides two modes of execution of the action portion of the rules. In the simultaneous mode, all actions within the rule take place at the same time, and none before the rule has all of its expressions evaluated. Should a rule awaken, in this mode the antecedent and all actions in the consequent are re-evaluated. On the other hand, using the in order mode, each action within the rule completes in sequence and the antecedent and other previously executed actions in the consequent are not re-evaluated when the rule is awakened.
- G2 provides an icon-based graphical interface which aids in building the KB, especially when describing the model of the application. For rule creation, an English-like grammar is used together with an interactive menu-driven grammar editor.
- G2 provides the Intelligent Control Protocol (ICP) that may be used to hook multiple G2's together and that is used to communicate with external data sources, such as sensors, actuators, and process control computers.
- G2 provides a simulator which may be used for debugging and knowledge validation.

LANGUAGE: G2 is written in CommonLISP.

HARDWARE: G2 may run on the TI Explorer, Symbolics 3620, the Sun 3 work station, any HP9000 series machine, Macintosh II, and the Dec VAX.

Interface between the process computer and G2 uses the ICP. In most of the installations, G2 has had to be custom-fitted in order to communicate with the process computer.

STATUS

System: Commercial

No. of installations: NA

Locations of installations: Monsanto, Sauget, Illinois, Reliable Water Co., Boston, Massachusetts

Installation information: Site 1: Reliable Water Co., Boston. This implementation controls a reversible-osmosis water purification process that involves opening and closing valves, turning on and off pumps, and other operations. It runs on two Macintosh IIs which provide a redundant computer configuration so that if one fails, the other may take control. The delivery system includes an input/output capability of 48 points on the process, a KB of around 1000 rules, with the base scan time of the system at 100 ms, and with rule execution rate at around 100 rules per second.

FUTURE DIRECTIONS: NA

3.2.7 Real-Time Expert System (RTES) [41]

DEVELOPER: RTS Real Time Systems Inc., Ontario

DOMAIN AND DESIGN GOALS: To provide a shell for the development of rule-based data acquisition and control applications.

ARCHITECTURE: It is composed of a realtime multitasking operating system, an inference engine, communication tasks, and a user interface allowing on-line editing of the KB as well as its utilization.

FEATURES

Knowledge representation: RTES uses rules of the form *variable = expression.* When a rule is triggered, RTES evaluates the expression on the right-hand side and then assigns the result (arithmetic or logical) to the variable on the left-hand side. The expressions use operands (variables and constants) along with a wide range of operators that include arithmetic, logical, comparisons, delays, indexing, and trigonometric functions. The variables are given names chosen by the designer of the application. For example, the rule *heater=temp* $<$ *min*, causes the *heater* to be true when *temp* is less than *min.*

In addition to rules, RTES allows the creation and execution of tasks which are a list of instructions very similar to BASIC code. Tasks may execute concurrently with the inference engine, thus allowing the simultaneous performance of continuous and batch control.

RTES also allows the implementation of loop control using the proportional, integral, and derivative (PID) algorithm. The parameters of the loop are part of the KB, thus allowing self-tuning and adaptive loop control strategies.

Working memory organization: The rules are memory resident. Memory is dynamically allocated during the KB editing process. This mechanism is transparent to the user.

Focusing mechanism: NA

Inference method: The RTES inference engine is data driven (forward chaining).

Methods for handling uncertainty

Data uncertainty (input reliability required): Data communication protocols reject invalid data by using the checksum or the cyclic redundancy check algorithms. Rules may be devised to ignore or signal erratic information.

Knowledge uncertainty: NA

Consistency maintenance mechanism: NA

Temporal/causal/spatial reasoning: RTES has a built-in scheduler that allows the entrance or removal of events at specified times of the day. In addition, date and time are system variables that are automatically maintained by RTES and may be used in rules like any other variables.

Interrupt handler/asynchronous inputs: All communications (serial ports, keyboards, printer) are handled by interrupts.

Other interesting features:

- An alert generator allows user-defined alert conditions that display a desired text at the bottom of the screen. Alert events may be logged to a disk file and/or printer.
- A batch file execution facility may contain a series of operator commands, including KB editing.
- A trend recording facility allows the system to keep track of specified variables at user-defined intervals. The resulting file is compatible with spread sheets.
- A series of variables may form a cascading block similar to shift registers.
- The user may define context dependent help files and alarm extension files to further a didactic purpose and advise on appropriate actions.
- A recipe download/upload facility makes RTES practical for applications where the set-up undergoes frequent changes.
- All inputs may be simulated by keystroke commands when a data acquisition and control subsystem is not present (or disconnected), thus allowing process simulations.
- A built-in, on-line color display generator allows the creation of customized dynamic displays to facilitate the use of the application by operating personnel. Variables may be displayed in different formats (numerics, bars, color change, text arrays).

LANGUAGE: RTES is written in Assembly language.

HARDWARE: The tool runs on an IBM PC/XT/AT or 100% compatibles using PC-DOS or MS-DOS Version 2.0 or later. It may interface with most data acquisition and control subsystems (such as OPTOMUX and PAMUX) and most programmable controllers (such as Allen Bradley, Mitsubishi, Modicon, Siemens, and Texas Instruments).

STATUS

System: Commercial. Currently distributed in Canada, U.S.A., and Western Europe

No. of installations: Around 200

Locations of installations: Ontario, Quebec, Alberta, Florida, Mexico, United Kingdom

Installation information: Typical installations are

- substation monitoring and control (Ontario Hydro)
- process control (Magna–Polyrim, Toronto, Ontario)
- machine control (Big-O, Exeter, Ontario)
- batch processing (BASF–Inmont, Brampton, Ontario)

FUTURE DIRECTIONS: RTS Inc. is continuing R&D to enhance RTES graphics capabilities and make it suitable for robotics and machine vision applications.

4 Conclusions

In conclusion it can be said that there is much work being done in the area of RTESs at universities, in industry, and in government agencies. Some of these RTESs are specific to a domain or application, whereas some are generic shells and general architectures for RTESs. It is often claimed that for applications wherein the RTES is replacing an old, existing system, the RTES is as good as or better than the existing system. For systems that aid operators in, for example, diagnosing faults, the approach of using ESs appears to be a promising and fruitful way of improving current techniques.

The following will review the issues brought up in Section 2 and will point out some observations and shortcomings inferred from the literature on the existing RTESs surveyed.

1. *Time constraints.* The issue of responsiveness has been addressed by most RTESs, with some form of intelligent scheduling using control strategies and metalevel knowledge, focusing mechanisms, parallelism, and VLSI technology being actively explored. The problem for hard realtime systems of guaranteeing response time is rarely addressed. Systems that do address this, HEXSCON [26] and DVMT [35, Lesser et al. 88], are investigating reasoning at different levels. These invoke different types of knowledge depending on the response times required, ranging from reflex-type reactions to in-depth inferencing.

2. *Critical decisions.* A good knowledge of the domain is needed for the system to make reliable decisions. It has been found that encoding only experiential and heuristic knowledge present the problems of [Kramer & Finch 88, Venkat & Rich 88]

   - knowledge acquisition bottleneck. Since this type of knowledge is obtained from domain experts, the acquisition of knowledge is found to be a bottleneck in ESs development and to require significant time and effort.
   - lack of experts. Without readily accessible domain experts, as is often the case in the process domain, experiential knowledge is not available.
   - lack of generality. It is found that experiential knowledge does not port well from one installation to another. For instance, process plants rarely have the same layout and knowledge must be built from scratch for each implementation. In addition, it is inflexible and hard to maintain.
   - reliability and completeness. This type of knowledge encodes known situations. It can lead to erroneous conclusions or total failures when dealing with unexpected and novel situations.

   As a solution to these problems, use of model-based reasoning from first principles is being investigated, even though this type of reasoning is time-consuming.

   There is much work underway in trying to develop robust knowledge representation schemes. The most promising use hybrid schemes that make the representation scheme fit the knowledge, instead of the knowledge fit the representation scheme.

   Regarding the issue of recovery after crashes, a few systems have archiving facilities for the data and variable values. These, however, do not archive the context (or the environment) in which the data or values are received or calculated. There is no record

kept of the decisions, and the reasons for these decisions, made by the system. This makes recovery after a crash and after-the-fact inspections by users very difficult.

3. *Continuous operation.* The memory management facilities required when in continuous operation and that take care of allocating and de-allocating memory are found to sometimes cause a degradation in the performance of the system. It is also found that garbage collection is a major problem especially for multiprocessor systems.

4. *Asynchronous events.* Although many of the existing RTESs can receive asynchronous inputs, it is not clear how quickly this input is actually processed. Many lack a facility for handling interrupts, especially in the case when the inference engine is currently processing a piece of knowledge such as a rule, a procedure, or method.

5. *Limited resources.* For a system to be able to efficiently manage its available resources, it has to know what each system component will require in terms of these resources. It was found that none of the surveyed RTESs take these resource requirements into consideration when planning and scheduling actions.

6. *Large KBs.* Some systems like PICON and G2 emphasize the KA phase (as in entering knowledge into the ES) by providing tools that facilitate the entry of knowledge. More work is needed to make these tools more intelligent and thus be a better aid to the knowledge builder, especially when building large KBs.

   RTESs may have to deal with multiple problems simultaneously. Parallelism is being explored to handle communication among multiple ESs to solve these problems or different parts of a single problem. When multiple ESs share information (allowing multiple ESs both read and write access to the same information), concurrency, integrity, and security issues must be addressed. For example, it must be determined which ES has authority to modify the information, how to schedule access to the information by concurrent ESs, how to secure information from unauthorized access. Much may be gained by incorporating previous work in the area of databases to address these.

7. *Uncertainty.* More robust techniques for verifying the reliability of data input are needed. In addition, it appears that methods currently employed to handle knowledge uncertainty are hard to use. The different measures of confidence given to working memory elements and the domain knowledge (as in the rules) are often subjective with no clear basis for them. These subjective assessments may vary widely. They are also quite closely tied in to the inferencing mechanism in the sense that tweaking these measures of confidence may cause unexpected changes in the inferencing.

8. *Dynamism and nonmonotonicity.* Truth and consistency maintenance is usually approached as a change in the likelihood of conclusions, decisions, and solution elements. Few systems go the route of finding assumptions, decisions, or conclusions that have to be adjusted and then adjusting these, together with all the assumptions, decisions, or conclusions these, directly or indirectly, support.

   Some existing RTESs attach a validity interval to data received and information inferred. Once this validity interval is exceeded the information is considered invalid and not used. The inference engine attempts to obtain more recent values for these.

9. *Temporal reasoning.* Temporal reasoning about the past and present in the form of some data analysis is done in some of the RTESs. Very few, though, can reason about possible futures. This may be attributed to the lack of knowledge, in the ES, on the dynamics of the subject domain. In addition, only very basic causal and spatial reasoning is done in a few of the existing RTESs.

10. *Integration of conventional and AI technique.* Most RTESs are able to perform some numeric computation in addition to symbolic computing. However, to provide true coupling of numeric and symbolic computing, an ES should not only be able to invoke numeric processes, but should also reason about the application or results of these numeric processes. For example, given several mathematical methods for handling uncertainty, an ES must be able to reason about and select which one to use.

11. *Interface to external environment.* For the human–machine interface, it is often required to describe the functional and structural components of the subject domain. This must go both ways, from the user to the ES (in the development phase) and from the ES to the user (in the runtime phase). This is not easily done in text form. Therefore, sophisticated graphical interfaces are increasingly becoming important.

    It is found that maintaining and augmenting KBs is often difficult. Tools to verify, validate, maintain, and augment the knowledge base must be developed that will take into account knowledge about the users, the subject domain, the current situations, and the interface capabilities of the system.

    Current explanation facilities usually give a straight trace of the knowledge executed by the ES. This may not be good enough for the operator to clearly understand the justifications for the system's decisions, thus causing him to be wary of these decisions.

    Other research being pursued is the incorporation of machine learning with RTESs where it may play a role in both the KA phase and in the actual run-time phase.

    An important consideration when building RTESs is portability and capability to integrate with existing systems, e.g., MIS, CAD databases, and process controllers. It is found that interface with such systems in existing RTESs is usually custom-fitted.

    It should also be pointed out that there is a trend towards using less expensive hardware, like PCs, to deliver ESs. This may be sensible because of the harsh environments that some RTESs may have to operate in.

As a final comment, it can be stated that although much has been done or is being done in the area of RTESs, more work and research is needed to develop techniques and methodologies to satisfactorily resolve the identified issues and problems.

5 Acknowledgements

I would like to thank the other members of the ARTEMIS project who read through several drafts of this report ant gave useful suggestions. In particular, Jack Brahan, Leslie Buck, Bob Orchard, and Rob Wylie from NRC; Rob Spring from the Noranda Technology Centre; Larry Paul and Roger Patterson from MPB Technologies, Inc.

6 List of Acronyms

AFO(s)	Activation Framework Object(s)
AI	Artificial Intelligence
CAD	Computer-Aided Design
DAG	Directed Acyclic Graph
DBMS	Data Base Management System
DDC	Direct Digital Control
DEA	Diagnostic Event Analyzer
EF	Exponential Filter
ESM	Electronic Surveillance Measures
ES(s)	Expert System(s)
FTA	Fault Tree Analysis
HCVM	Heuristic Control Virtual Machine (FMC & Teknowledge)
HMI	Human Machine Interface
ICP	Intelligent Communications Protocol (Gensym Corp.)
ICU	Intensive Care Unit
JES	Job Entry System (IBM)
KA	Knowledge Acquisition
KB	Knowledge-Base(d)
KE	Knowledge Engineer
KR	Knowledge Representation
KS(s)	Knowledge Source(s)
LCL	Linguistic Control Language (LINKman)
LTM	Long Term Memory
MB	Measure of Belief
MCCF	MVS Communications Control Facility (IBM)
MD	Measure of Disbelief
MIS	Management Information Systems
MVS	Multiple Virtual Storage (IBM)
NA	information Not Available from the literature surveyed
PRECARN	PRECompetitive Applied Research Network
PSMS	Problem State Monitoring System
RTES(s)	Real Time Expert System(s)
SQC	Statistical Quality Control
STM	Short Term Memory
TMS	Truth Maintenance System
VM	Virtual Memory
WM	Working Memory
WME(s)	Working Memory Element(s)

Figure 13: Acronyms used

References

[1] Hasp/Siap: an ocean surveillance signal understanding ES.

Nii, H.P., Feigenbaum, E.A., Anton, J.J., and Rockmore, A.J. Signal-to-symbol transformation: HASP/SIAP case study. The AI Magazine, Spring 1982.

[2] Stammer: an ocean surveillance signal understanding ES.

Bechtel, R., Morris, P., and Kibler, D. Incremental deduction in a real-time environment. CSCSI 1980.

[3] MXA: a blackboard multiple-expert shell for prototyping sensor-based continuous interpretation systems.

Tailor, A. MXA — a blackboard expert system shell. Blackboard Systems. R. Engelmore and T. Morgan (eds.), Addison-Wesley Publishing Co., 1988.

Stammers, R.A. The MXA shell. The Proceedings of the 4th Technical Conference of the British Computer Society, Specialist Group on Expert Systems, M.A. Bramer (ed.), December 1984.

[4] An Electronic Support Measures (ESM) prototype: a prototype system designed to improve the performance of shipboard systems which detect and analyze electromagnetic emissions from radar emitters carried by platforms such as ships and aircraft.

Hood, S.T., and Mason, K.P. Knowledge-based systems for real-time applications. Applications of Expert Systems, J. Ross Quinlan (ed.), 1987.

[5] Automated Load Forecasting Assistant (ALFA): an ES that assists in forecasting short term demand for electricity.

Jabbour, K., Vega-Riveros, J.F., Landsbergen, D., and Meyer, W. ALFA: automated load forecasting assistant. WESTEX 1987.

[6] Annie & Violet: diagnosis and machine health monitoring ES shells.

Milne, R. On-line diagnosis applications. KBS 87: Online Publications, Pinner, UK, 1987.

[7] Cooker: a process monitoring and operator advisory system for batch manufacturing processes.

Allard, J.R., and Kaemmerer, W.F. The goal/subgoal knowledge representation for real-time process monitoring. AAAI 1987.

[8] Sinter Advisor (SA): an ES to diagnose moisture level problems in a sinter plant.

Brew, P.J., and Catlett, J. SA: an expert system for trouble-shooting a smelter. Applications of Expert Systems, J. Ross Quinlan (ed.), 1987.

[9] SCript ANalyst (SCAN): a script-based ES for battlefield monitoring.

Laskowski, S.J., and Hofmann, E.J. Script-based reasoning for situation monitoring. AAAI 1987.

[10] Diagnostic Event Analyzer (DEA): a system for automated diagnosis of malfunctions in continuous processes.

Kramer, M. Automated diagnosis of malfunctions based on object-oriented programming. Journal of Loss Prevention in the Process Industries, April 1988.

[11] Reactor: an ES that assists operators in the diagnosis and treatment of nuclear reactor accidents.

Nelson, W.R. Reactor: an expert system for diagnosis and treatment of nuclear reactor accidents. AAAI 1982.

[12] A Flight Expert System (FLES): a flight ES to assist pilots in monitoring, diagnosing and recovering from in-flight faults.

Ali, M., Scharnhorst, D.A., Ai, C.S., and Ferber, H.J. A flight expert system (FLES) for on-board fault monitoring and diagnosis. USAF Workshop on AI Applications, 1986.

[13] Artifact: a Prolog shell for real time feedback control applications.

Francis, J.C., and Leitch, R.R. Artifact: a real-time shell for intelligent feedback control. The Proceedings of the 4th Technical Conference of the British Computer Society, Specialist Group on Expert Systems, M.A. Bramer (ed.), December 1984.

[14] YES/MVS: an ES that interactively controls an operating system as an aid to computer operators.

Milliken, K.R., Cruise, A.V., Ennis, R.L., Finkel, A.J., Hellerstein, J.L., Loeb, D.J., Klein, D.A., Masullo, M.J., Van Woerkom, H.M., and Waite, N.B. YES/MVS and the automation of operations for large computer complexes. IBM Systems Journal, Vol. 25, No. 2, 1986.

Kastner, J.K., Ennis, R.L., Griesmer, J.H., Hong, S.J., Karnaugh, M., Klein, D.A., Milliken, K.R., Schor, M.I., and Van Woerkom, H.M. A continuous real-time expert system for computer operations. KBS '86: Online Publications, Pinner, UK, 1986.

Griesmer, J.H., Hong, S.J., Karnaugh, M., Kastner, J.K., Schor, M.I., Ennis, R.L., Klein, D.A., Milliken, K.R., and Van Woerkom, H.M. YES/MVS: a continuous real time expert. AAAI 1984.

[15] Process Intelligent CONtrol (PICON): a LISP ES tool for building real time systems.

Collier, D., and Neal, P.W. The detection of abnormal operating conditions in a pulverising fuel mill using an on-line expert system. IEE Colloquium on Knowledge Based Advisory Systems for Monitoring and Supervisory Control, No. 83, October 1987.

Leinweber, D. Expert systems in space. IEEE Expert, Spring 1987.

Leinweber, D., and Perry, J. Controlling real-time processes in the space station with expert systems. Proceedings of SPIE International Society for Optical Engineering, Space Station Automation II, 1986.

Leinweber, D., and Gidwani, K. Real-time expert system development techniques and applications. IEEE Western Conference on knowledge-based engineering and expert systems, 1986.

PICON Users Guide. Version 1.0 Revision A. LISP Machine Inc., Cambridge MA., 1986.

Khanna, R., and Moore, R.L. Expert systems involving dynamic data for decisions. Second International Expert Systems Conference, 1986.

Gidwani, K.K., and Dalton, W.S. Innovative knowledge engineering for real-time expert systems. IFAC Software for Computer Control, May 1986.

Moore, R.L., and Kramer, M.A. Expert systems in on-line process control. Chemical Process Control 3, 1986.

Moore, R.L., Hawkinson, L.B., Knickerbocker, C.G., and Churchman, L.M. A real-time expert system for process control. IEEE 1st Conference on AI Applications, 1984.

[16] Stochasm: a fault detection and diagnostic system for the lubrication oil subsystem in a US navy suface ship gas turbine propulsion unit.

Malkoff, D.B. Real-time fault detection and diagnosis: the use of learning expert systems to handle the timing of events. Navy Personnel Research and Development Center, NPRDC TR 87-8, November 1986.

[17] An AI system for military communications: an ES implemented in hardware that can check a damaged network for status and availability of communications between any 2 nodes.

Ulug, M.E. A real-time AI system for military communications. IEEE 3rd Conference on AI Applications, 1987.

Ulug, M.E. VLSI knowledge representation using predicate logic and cubical algebra. Proceedings of the IEEE Phoenix Conference on Computers and Communications, February 1987.

Ulug, M.E., and Bowen, B.A. A unified theory of algebraic topological methods for the synthesis of the switching systems. IEEE Transactions on Computers, C-23(3), March 1974.

[18] Dantes: a real time assistant to network supervisors in carrying out troubleshooting activities.

Mathonet, R., Van Cotthem, H., and Vanryckeghem, L. DANTES An expert system for real-time network troubleshooting. IJCAI 1987.

[19] Computerized Emergency Action Level Monitor and the Reactor Emergency Action Level Monitor (CEALM/REALM): RTESs that monitor a nuclear power plant.

Touchton, R. Emergency classification: a real time expert system application. Proceedings of Southcon, 2321–2323, 1986.

[20] LINKman: a full supervisory control toolkit capable of performing real time rule-based closed loop control of complex multivariable processes.

Taunton, J.C., and Haspel, D.W. Personal communication, November 1988.

Haspel, D.W. LINKman kiln control system. Tirkua Conference, London, May 1988.

Haspel, D.W., and Taunton, J.C. High level control of cement kilns; a practical example of real time expert system technology. IEE Symposium on Practical Implementation of Expert Systems in Industry, December 1987.

Thomson, A.C. Real time artificial intelligence for process monitoring and control. BP International Ltd., Britannic House, Moor Lane, London, England.

Haspel, D.W., Lorimer, A.D.J., Southan, C.J., and Taylor, R.A. Blue Circle high level kiln control. IEEE Cement Industry Technical Conference, 1987.

Taunton, J.C. Rule based expert systems — an application in real time process control and optimisation. Second International Expert Systems Conference, 1986.

Haspel, D.W., and Taunton, J.C. Applying linguistic control to a process optimisation: a case study. KBS '86: Online Publications, Pinner, UK, 1986.

Taunton, C. Expert systems go on-line at Blue Circle. Sensor Review, April 1986.

Taunton, J.C. A rule based supervisory control system. IEE Colloquium on Real-time Expert Systems in Process Control, No. 107, November 1985.

[21] A rotary cement kiln supervisor: an ES that performs supervisory control over a cement manufacturing application.

Norman, P., and Naveed, S. An expert system supervisor for a rotary cement kiln. IEE Colloquium on Real Time Expert Systems in Process Control, No. 107, 1985.

[22] Fault AnaLysis CONsultant (FALCON): a diagnostic tool to monitor and analyze alarms in a chemical process plant.

Shirley, R.S. Personal communication. The Foxboro Co., MA, September 14, 1988.

Shirley, R.S. The practicalities of expert systems (for process control applications). The Foxboro Company, June 1988.

Shirley, R.S. Status report 3 with lessons: an expert system to aid process control. Proceedings of the IEEE Pulp and Paper Industry Technical Conference, 1987.

Shirley, R.S. An expert system for process control. Tappi Journal, May 1987.

Rowan, D.A. On-line fault diagnosis: initial success and future directions. Proceedings of the ISA 87 International Conference and Exhibit, 1211–1218, 1987.

Chester, D., Lamb, D., and Dhurjati, P. Rule-based computer alarm analysis in chemical process plants. Proceedings of the Micro-Delcon, 1984.

Chester, D., Lamb, D., and Dhurjati, P. An expert system approach to on-line alarm analysis in power and process plants. Computers in Engineering, 1984.

[23] Expert Navigator: an ES designed to manage navigation sensors on advanced tactical aircraft and to make critical equipment and mission decisions from an aircrew's perspective.

Pisano, A.D. and Jones, H.L. An expert systems approach to adaptive tactical navigation. IEEE 1st Conference on AI Applications, 1984.

[24] NAVEX: a navigation ES for the space shuttle.

Healey, K.J. Artificial intelligence research and applications at the NASA Johnson Space Center. The AI Magazine Vol. 7, No. 3, 1986.

Maletz, M.C. NAVEX: Space shuttle navigation expert system. Proceedings of the 1st Annual Artificial Intelligence and Advanced Computer Technology Conference, Long Beach, CA, 1985.

Marsh, A. NASA to demonstrate artificial intelligence in flight operations. Aviation Week and Space Technology, September 17, 1984.

[25] AT&T VLSI chip for approximate reasoning: an inference architecture implemented on a chip using VLSI technology that copes with uncertainty and performs approximate reasoning.

Togai, M., and Watanabe, H. Expert system on a chip: an engine for real-time approximate reasoning. IEEE Expert, Fall 1986, Vol. 1, No. 3.

Togai, M., and Watanabe, H. A VLSI implementation of fuzzy inference engine: toward an expert system on a chip. Proceedings of the 2nd Conference on AI Applications, IEEE Computer Society, 1985. (also in Information Sciences 38: 147–163, 1986)

[26] Hybrid EXpert System CONtroller (HEXSCON): a hybrid microcomputer-based ES intended to deal with control problems often encountered in military and advanced industrial applications.

Wright, M.L. HEXSCON: A hybrid microcomputer-based expert system for real-time control applications. IEEE Western Conference on Knowledge-based Engineering and Expert Systems, 1986.

Wright, M.L., Green, M.W., Fiegl, G., and Cross, P.F. An expert system for real-time control. IEEE Software, March 1986.

[27] A model-based fault diagnoser: a model-based approach to diagnosing faults in a mechanical plant.

Koukoulis, C.G. KBS for fault diagnosis in real-time. KBS '86: Online Publications, Pinner, UK, 1986.

[28] Real-time Expert System Club of Users (RESCU): an Alvey (UK)-sponsored project to build a realistic on-line demonstrator ES for real time process control.

Dulieu, M.R.W., and Leitch, R.R. Experiences in design and development of a practical real-time expert system. IEE Colloquium on KB Advisory Systems for Monitoring and Supervisory Control, No. 83, October 1987.

Bailey, M.G. RESCU: a KBS for quality control. Presented at KBS 87: Online Publications, Pinner, UK, 1987.

Shorter, D.N. The Alvey RESCU project — a progress report. IEE Colloquium on Real-Time Expert Systems in Process Control, No. 107, November 1985.

[29] Starplan: an ES for satellite anomaly resolution.

Siemens, R.W., Golden, M., and Ferguson, J.C. StarPlan II: evolution of an expert system. AAAI 1986.

[30] Gensym Real-Time Expert System (G2): a framework for developing systems that monitor and control complex real time functions.

Matthews, B., Lindenfelzer, P., Hawkinson, L., and Moore, R. Process control with the G2 real-time expert system. International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, 1988.

G2 User's Manual, Version 1.0. Gensym, Corp., Cambridge MA., 1988.

Wolfe, A. An easier way to build a real-time expert system. Electronics, Vol. 60, No. 5, March 1987.

[31] Expert System for Complex Operations in Real Time (ESCORT): an ES designed to relieve the cognitive load on users of information systems generating large volumes of dynamic data.

Sachs, P. Personal communication. PA Computers and Telecommunications, London, UK, December 1988.

Thomson, A.C. Real time artificial intelligence for process monitoring and control. BP International Ltd., Britannic House, Moor Lane, London, England.

Sargeant, R.A.E. Integration of real time expert systems. KBS 87: Online Publications, Pinner, UK, 1987.

Sargeant, R.A.E. Real-time expert systems for the oil and gas industry. KBS '86: Online Publications, Pinner, UK, 1986.

Sachs, P., Paterson, A.M., and Turner, M.H.M. ESCORT — an expert system for complex operations in real time. Expert Systems, Vol. 3, No. 1, January 1986.

Paterson, A., Sachs, P., and Turner, M. ESCORT: the application of causal knowledge to real-time process control. Expert Systems 85, Proceedings of the 5th Technical Conference of the British Computer Society, Specialist Group on Expert Systems, M. Merry (ed.), 1985.

[32] Process Diagnostic System (PDS): a rule-based architecture for online real time diagnosis of machine processes.

Osborne, R.L. Online, artificial intelligence-based turbine generator diagnostics. The AI Magazine, Vol. 7, No. 4, Fall 1986.

Osborne, R.L., Gonzalez, A.J., Bellows, J.C., and Chess, J.D. On-line diagnosis of instrumentation through artificial intelligence. Proceedings of the Instrument Society of America Power Symposium, 89–94. Research Triangle Park, N.C.: Instrument Society of America, 1985.

Osborne, R.L., Gonzalez, A.J., Emery, F.T., Hurwitz, M.J., McCloskey, T.H., and Edmonds, J.S. Increased power plant availability and reliability through online diagnosis based on artificial intelligence. Proceedings of the American Power Conference, 539–543, Chicago, IL, 1985.

Fox, M.S., Lowenfeld, S., and Kleinosky, P. Techniques for sensor-based diagnosis. IJCAI 1983.

[33] ADS: an ES that provides expert-level assistance to specialists in defining medically-oriented, sensor-based applications.

Hollander, C.R. and Reinstein, H.C. A knowledge-based application definition system. IJCAI 1979.

[34] Materials Composition Management (MCM): an ES that controls a chemical manufacturing process.

Raulefs, P., D'Amrosio, B., Fehling, M., Forrest, S., and Wilber, M. Real-time process management for materials composition. IEEE 3rd Conference on AI Applications, 1987.

D'Ambrosio, B, Fehling, M.R., Forrest, S., Raulefs, P., and Wilber, B.M. Real-time process management for materials composition in chemical manufacturing. IEEE Expert, Summer 1987.

[35] Distributed Vehicle Monitoring Testbed (DVMT): a vehicle monitoring ES.

Durfee, E.H., and Lesser, V.R. Incremental planning to control a blackboard-based problem solver. AAAI 86.

Lesser, V.R., and Corkill, D.D. The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. The AI Magazine, Vol. 4, No. 3, 1983. (also in Blackboard Systems, Engelmore, R., and Morgan, T. (eds.), Addison-Wesley Publishing Co., 1988).

[36] Qualitative Process Automation (QPA): a generic control system architecture applied to autoclave curing of composites.

LeClair, S.R. Major, and Abrams, F.L. Qualitative process automation. Proceedings of the 27th Conference on Decision and Control, 558–563, December 1988.

Garrett, P., Lee, C.W., and LeClair, S.R. Major, Qualitative process automation vs. quantitative process control. American Control Conference, 1368–1373, June 1987.

[37] Comdale/C Process Control System: an ES development tool used for constructing process control applications.

Hall, M. Personal communication. Comdale Technologies, Ont., 14 November, 1988.

Harris, C.A., and Woo, E. Expert systems: application for process control. Presented at the First Canadian Conference on Computer Applications in the Mineral Industry, March 1988.

Musial, J. Fuzzy logic, computer-controlled machines catching on. The Northern Miner Magazine, January 1988.

Harris, C.A., and Kosick, G.A. Expert system technology at the Polaris Mine. Presented at the Canadian Mineral Processors Annual Operator's Conference, January 1988.

[38] Personal Consultant Series: an ES development tool.

Hadley, J. Expert system workstation for machine/process maintenance, service, and training. Proceedings of the ISA 87 International Conference and Exhibit, 1201–1210, 1987.

Darius, I.H., and Vion, JP, Rule based control: using process knowledge for better control. Proceedings of the ISA 87 International Conference and Exhibit, 827–831, 1987.

Carlson, K. Process-driven expert systems. Proceedings of the Third Annual Artificial Intelligence and Advanced Computer Technology Conference, April 1987.

[39] ONSPEC Superintendent: a PC ES development tool.

Spring, R., and Edwards, R. Real-time expert system control of the Brenda Mines grinding circuit. April 1988.

Schwartz, T.J. Real time processing on the PC. Spang Robinson Report, 3(7): 9–17, 1987.

[40] Activation Framework (AF): software framework that supports the implementation of real-time AI programs on multiple, interconnected computers.

Green, P.E. AF: A framework for real-time distributed cooperative problem solving. Distributed Artificial Intelligence, M.N. Huhns (ed.), Pitman, London, Morgan Kaufmann Pub., Inc., Los Altos, CA, 1985.

[41] Real-Time Expert System (RTES): a tool for the development of rule-based data acquisition and control applications.

Simtob, F. Personal communication. RTS Real Time Systems Inc., May 1989.

RTES — the real time expert system for smart automation, User Manual, Transduction Limited, 4th edition, November 1987.

[Allen 84] Allen, J.F. Towards a general theory of action and time. Artificial Intelligence, 23: 123–154, July 1984.

[Allen & Hayes 85] Allen, J.F., and Hayes, P.J. A common-sense theory of time. IJCAI 1985.

[Barr & Feigenbaum 81] Barr, A., and Feigenbaum, E.A. (eds.) The handbook of artificial intelligence, Vol. 1. William Kaufmann, Inc., 1981.

[Bennett 87] Bennett, M.E. Real-time continuous AI systems. IEE Proceedings, Vol. 134, Pt. D, No. 4, July 1987.

[Behan & Lecot 87] Behan, J., and Lecot, K. Overview of financial applications of expert systems. Proceedings of WESTEX 1987.

[Bobrow 85] Bobrow, D.G. Qualitative reasoning about physical systems. The MIT Press, Cambridge, MA, 1985.

[Bowen 87] Bowen, B.A. Real-time expert systems: a status report. AGARD Lecture Series No. 155, Knowledge-base Concepts and AI: Applications to Guidance and Control, 1987.

[Bylander & Mittal 86] Bylander, T.C., and and Mittal, S. CSRL: A language for classificatory problem solving and uncertainty handling. The AI Magazine, Vol. 7, No. 2, Summer 1986.

[Chandra & Punch 87] Chandrasekaran, B., and Punch, W.F. III, Data validation during diagnosis, a step beyond traditional sensor validation. AAAI 87.

[Cohen 84] Cohen, P.R. Heuristic reasoning about uncertainty: an artificial intelligence approach, PhD dissertation, Stanford University, 1984. (also published as Research Notes in Artificial Intelligence 2, Pitman Advanced Publishing Program, 1985).

[Cohen & Grinberg 83] Cohen, P.R., and Grinberg, M.R. A theory of heuristic reasoning about uncertainty. The AI Magazine, Vol. 4, No. 2, Summer 1983.

[Cruise et al. 87] Cruise, A., Ennis, R., Finkel, A., Hellerstein, J., Klein, D., Loeb, D., Masullo, M., Milliken, K., Van Woerkom, H., and Waite, N. YES/L1: integrating rule-based, procedural, and real-time programming for industrial applications. IEEE 3rd Conference on AI Applications, 1987.

[Davis 83] Davis, R. Diagnosis via causal reasoning: paths of interaction and locality principle. AAAI 1983.

[Doyle 79] Doyle, J. A truth maintenance system. Artificial Intelligence, Vol. 12, No. 3, 1979.

[Dvorak 87] Dvorak, D.L. Expert monitoring and control. Proceedings of the Third Annual AI and Advanced Computer Technology Conference, April 1987.

[Factor & Gelernter 88] Factor, M., and Gelernter, D. The parallel process lattice as an organizing scheme for realtime knowledge daemons. Yale University, March 1988.

[Factor et al. 88] Factor, M., Gelernter, D., Miller, P., and Rosenbaum, S. A prototype realtime knowledge daemon for ICU monitoring. Yale University, March 1988.

[Ferguson 86] Ferguson, J.C. Beyond rules, The next generation of expert systems. USAF Workshop on AI Applications, 1986.

[Finch & Kramer 87] Finch, F.E., and Kramer, M.A. Narrowing diagnostic focus by control system decomposition. AIChE Spring National Meeting, 1987.

[Forgy 82] Forgy, C.L. Rete: a fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence, Vol. 19, No. 1, 1982.

[Gupta 85] Gupta, A. Parallelism in production systems: the sources and the expected speed up. Fifth International Workshop, Expert Systems & Their Applications, Vol. 1, 1985.

[Hallam 87] Hallam, J. Autonomous vehicles as real-time expert systems. DAI research paper no. 332, Department of Artificial Intelligemce, University of Edinburgh, Edinburgh, Scotland, 1987.

[Hayes-Roth 84] Hayes-Roth, B. BB1: An architecture for blackboard systems that control, explain, and learn about their own behavior, Report no. STAN-CS-84-1034 (also numbered: HPP-84-16), Stanford University, December 1984.

[Hayes-Roth 85] Hayes-Roth, B. A blackboard architecture for control. Artificial Intelligence, 26: 251–321, 1985.

[Hayes-Roth & Lesser 77] Hayes-Roth, F., and Lesser, V.R. Focus of attention in the Hearsay-II speech understanding system. IJCAI 1977.

[Hayes-Roth et al. 88] Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R., and Seiver, A. Distributing intelligence within an individual. Report no. STAN-CS-88-1229 (also numbered KSL-88-50), Department of Computer Science, Stanford University, November 1988.

[Holmbad & Ostergaard 82] Holmbad, L.P., and Ostergaard, J-J. Control of a cement kiln by fuzzy logic. Fuzzy Information and Decision Processes, M.M. Gupta and E. Sanchez (ed.), North-Holland Publishing Company, 1982.

[Johnson et al. 88] Johnson, T., Hewett, J., Guilfoyle, C., and Jeffcoate, J. Expert systems markets and suppliers. Ovum Limited, 1988.

[Kaemmerer & Allard 87] Kaemmerer, W.F., and Allard, J.R. An automated reasoning technique for providing moment-by-moment advice concerning the operation of a process. AAAI 1987.

[Kramer & Finch 87] Kramer, M. A., and Finch F.E. Fault diagnosis of chemical processes. Knowledge-Based Systems Diagnosis, Supervision and Control, S.G. Tzafestas (ed.), Plenum Press, 1987.

[Kramer & Finch 88] Kramer, M. A., and Finch F.E. Development and classification of expert systems for chemical process fault diagnosis. Robotics & Computer-Integrated Manufacturing, Vol. 4, No. 3/4, 437–446, 1988.

[Laffey et al. 88] Laffey, T.J., Cox, P.A., Schmidt, J.L., Kao, S.M., and Read, J.Y. Real-time knowledge-based systems. The AI Magazine, Vol. 9, No. 1, Spring 1988.

[Leitch 87] Leitch, R.R. Modelling of complex dynamic systems. IEE Proceedings, Vol. 134, Pt. D, No. 4, July 1987.

[Lesser & Erman 77] Lesser, V.R., and Erman, L.D. A retrospective view of the Hearsay-II architecture. IJCAI 1977.

[Lesser et al. 75] Lesser, V.R., Fennell, R.D., Erman, L.D., and Reddy, D.R. Organization of the Hearsay-II speech understanding system. IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-23, 1975.

[Lesser et al. 88] Lesser, V.R., Pavlin, J., and Durfee, E. Approximate processing in real-time problem solving. The AI Magazine, Vol. 9, No. 1, Spring 1988.

[Maletz 87] Maletz, M.C. An architecture for real time knowledge-based control. Dearborn, Michigan, Society of Manufacturing Engineers, 1987.

[McClelland 88] McClelland, S. Giving AI real-time sensing. Sensor Review, January 1988.

[Minsky 68] Minsky, M., Matter, mind and models. Semantic Information Processing, M. Minsky (ed.), MIT Press, Cambridge, MA, 1968.

[Quillian 68] Quillian, M.R. Semantic memory. Semantic Information Processing, M. Minsky (ed.), MIT Press, Cambridge, MA, 1968.

[Quinlan 85] Quinlan, J.R. Induction of decision trees. The New South Wales Institute of Technology, School of Computing Sciences, Technical Report 85.6, 1985.

[Rader et al. 87] Rader, C.D., Crowe, V.M., and Marcott, B.G. Caps: a pattern recognition expert system prototype for respiratory and anesthesia monitoring. IEEE Western Conference on Knowledge-based Engineering and Expert Systems, 1987.

[Rodriquez & Rivera 86] Rodriguez, G., and Rivera, P. A practical approach to expert systems for safety and diagnostics. InTech, July 1986.

[Roffel & Chin 87] Roffel, B., and Chin, P. Computer control in the process industries. Lewis Publishers, 1987.

[Sauers & Walsh 83] Sauers, R., and Walsh, R. On the requirements of future expert systems. IJCAI 1983.

[Schank & Abelson 77] Schank, R.C., and Abelson, R.P. Scripts, plans, goals, and understanding: an inquiry into human knowledge structures. John Wiley & Sons, 1977.

[Shafer 76] Shafer, G. A mathematical theory of evidence. Princeton University Press, 1976.

[Shortliffe 76] Shortliffe, E.H. Computer-based medical consultation: MYCIN. New York: American Elsevier, 1976.

[Sloman 85] Sloman, A. Real time multiple-motive expert systems. Expert Systems 85, Proceedings of the 5th Technical Conference of the British Computer Society, Specialist Group on Expert Systems, M. Merry (ed.), 1985.

[Stefik & Bobrow 83] Stefik, M., and Bobrow, D. The LOOPS manual. Xerox Palo Alto Research Center, 1983.

[Taunton 87] Taunton, J.C. Real time process management. KBS 87; Online Publications, Pinner, UK, 1987.

[Venkat & Rich 87] Venkatasubramanian, V., and Rich, S.H. Model-based reasoning for fault diagnosis. AIChE, 1987.

[Venkat & Rich 88] Venkatasubramanian, V., and Rich, S.H. An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis. Special Issue of Computers and Chemical Engineering on AI, 1988.

[Zadeh 65] Zadeh, L.A. Fuzzy sets. Information Control, Vol. 8, 338–353, 1965.