

Conf - 9110272--1

PNL-SA--19551

DE92 002701

Received by OSTL
NOV 12 1991

TOWARDS THE DEVELOPMENT OF MULTILEVEL-
MULTIAGENT DIAGNOSTIC AIDS

R. C. Stratton
D. B. Jarrell

October 1991

Presented at the
2nd International Workshop on
Principles of Diagnosis
October 14-16, 1991
Milano, Italy

Work supported by
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

TOWARDS THE DEVELOPMENT OF MULTILEVEL-MULTIAGENT DIAGNOSTIC AIDS

Rex (Trav) C. Stratton
Don B. Jarrell
Pacific Northwest Laboratory
999 Battelle Boulevard
Richland, Wa. 99352

ABSTRACT

Presented here is our methodology for developing automated aids for diagnosing faults in complex systems. We have designed these aids as multilevel-multiagent diagnostic aids based on principles that should be generally applicable to any complex system. In this methodology, "multilevel" refers to information models described at successive levels of abstraction that are tied together in such a way that reasoning is directed to the appropriate level as determined by the problem solving requirements. The concept of "multiagent" refers to the method of information processing within the multilevel model network; each model in the network is an independent information processor, i.e., an intelligent agent.

1.0 INTRODUCTION

Presented here is our methodology for developing automated aids for diagnosing faults in complex systems. We undertook this research at the Pacific Northwest Laboratory (PNL)¹ for the U.S. Department of Energy and the U. S. Nuclear Regulatory Commission. We have designed these aids as multilevel-multiagent diagnostic aids based on principles that should be generally applicable to any complex system. In this methodology, "multilevel" refers to information models described at successive levels of abstraction that are tied together in such a way that reasoning is directed to the appropriate level as determined by the problem solving requirements. The concept of "multiagent" refers to the method of information processing within the multilevel model network; each model in the network is an independent information processor, i.e., an intelligent agent.

1.1 ROOT CAUSE ANALYSIS

Our research in fault diagnosis at PNL grew out of our work in root-cause analysis (RCA). RCA is the process of determining the fundamental cause for the degradation or failure of an artifact [1]. RCA consists of two major activities: fault diagnosis and root-cause evaluation (RCE). These activities and their relation with each other are shown in Figure 1 in relation to a "plant" within which a complex process-system can be found. The purpose of fault diagnosis is to characterize and specify faults, i.e., determine the plant events and conditions that are associated with a specific symptom. Then RCE is used to determine the cause of the events and condition.

1.2 GENERAL APPROACH

It is our opinion that software development in general is evolving from an ad hoc activity to an engineering discipline. At a minimum, the classical life-cycle approach to software system development includes tasks such as problem definition, conceptual design, design, construction, certification, implementation, and maintenance. Intelligent systems, on the other hand, modify these tasks by requiring additional activities to be performed. In our

¹ Operated for the U.S. Department of Energy by Battelle Memorial Institute under Contract DE-AC06-76RLO 1830.

approach, we categorize these activities as 1) determining knowledge requirements, 2) constructing models, and 3) developing the requirements for representation schemes

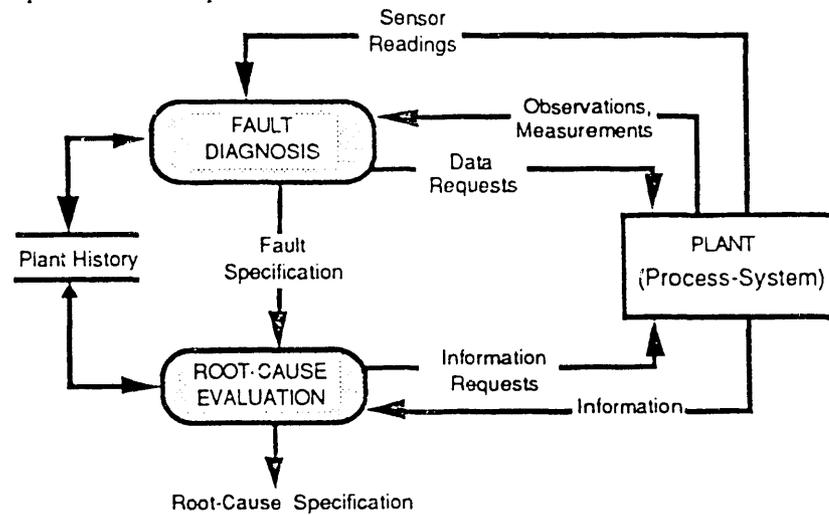


Figure 1. Schematic of the activities in root-cause analysis

1.2.1 Knowledge Acquisition

We group knowledge for fault diagnosis into two broad groups: cognitive task and process-system knowledge. Cognitive task knowledge is knowledge of how to perform a task such as fault diagnosis. This knowledge includes inference methods, control strategy, procedures, and methods or criteria for making decisions. The process-system knowledge consists of knowledge about the process-system structure, function, constraints, physics, and faults. Process-system knowledge is used with plant state and event data to develop information about the behavior of the plant.

During the reasoning process, the knowledge contained in these two groups are related to each other, back and forth, to draw conclusions about the process-system, as illustrated in Figure 2. This paper focuses on the determination and specification of process-system knowledge.

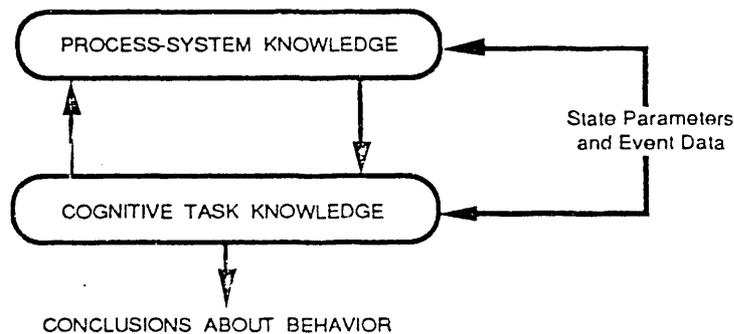


Figure 2. Interaction between the process-system and task models

Domain knowledge is acquired in both the knowledge acquisition and model construction activities. However, the knowledge acquisition activity is the primary and initial mechanism for acquiring domain knowledge. Model construction is a secondary mechanism for acquiring knowledge. During model construction additional knowledge requirements may be discovered and existing knowledge may be refined and modified.

1.2.2 Model Construction

By model we mean a representation of a specified reality which captures some essential aspects of the reality within a framework of a representation method. The model of the reality provides a means of exploring the properties of that reality. This definition is an adaptation of the definition for mathematical models presented by J.L. Casti [2]. The important issues concerning models are that 1) they capture essential aspects 2) in an appropriate

representation 3) in order to explore properties of the reality. This means that the essential properties and the purpose of the model must be understood. It also means that the method for representing the model must allow for inferencing that accomplishes its purpose.

For fault diagnosis, the essential properties are knowledge of the process-system and how to perform a diagnosis. In our methodology for developing automated aids for fault diagnosis, this knowledge is represented as models using quantitative calculus, a qualitative calculus, predicate logic, and intelligent agents.

As might be expected, because the major categories of knowledge are cognitive task and process-system, the major categories of model development are the same. In this chapter, however, we only discuss the modeling of process-system knowledge. Preliminary work on modeling the diagnosis task is presented in Stratton and Jarrell [1]. The model construction activity is used to construct fault-association models. Quantitative and agent models are primarily developed in the problem definition activity.

1.2.3 Representation Scheme Requirements Development

Once the process-system knowledge is represented as a system of quantitative and qualitative models, it is necessary to determine the knowledge representation requirements in order that the models can be implemented in a software framework. Because we are using a model-based reasoning approach, the representation scheme will have a general requirement that it provide a means for the knowledge to be organized and executed as a system of successively abstract and integrated models which function interactively as required during problem solving.

Representation scheme requirements are determined by analyzing the cognitive and process-system knowledge. These requirements generate a specification which the representation scheme should satisfy. The representation scheme should provide methods for representing and organizing information as well as methods for performing inference and reasoning control.

2.0 KNOWLEDGE REQUIREMENT ISSUES

Before discussing the major subjects of this chapter (knowledge determination/acquisition, model construction, and representation scheme determination), we feel it is important to first discuss some general issues associated with knowledge requirements for intelligent system. Specifically, in this section we discuss 1) knowledge requirements for process-system control and diagnosis, 2) the relationship between knowledge requirements for different cognitive tasks, and 3) interaction between quantitative and qualitative knowledge.

2.1 PROTOTYPE SYSTEM

In illustrating and discussing the concepts to be presented in this section and the remainder of this chapter, we will use a cooling system of the kind used in nuclear reactor service water systems as a prototype of a process-system. A diagram of the prototype is shown in Figure 3. The function of this process-system is to cool the process fluid routed through the shell side of the heat exchanger. The system consists of a pump, three valves (V1, V2, and V3), a heat exchanger, interconnecting piping, and instrumentation.

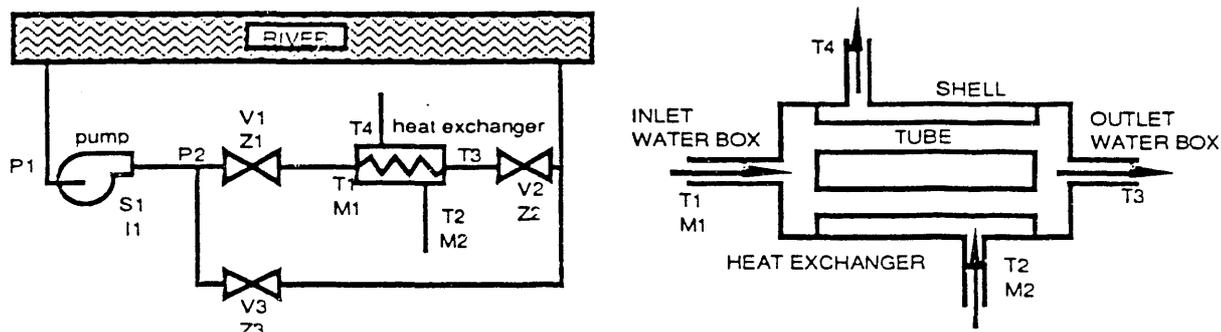


Figure 3. Sc-system, which is in the cooling mode, as illustrated in Figure 4.

In operation, the pump draws water into the system from the river and routes it through the tube side of the heat exchanger. A different system then routes the process fluid through the shell side of the heat exchanger. The process-system has several modes. These modes are defined by the pump speed (two speeds) and the settings of the valves. The instrumentation for this system is, as shown in Figure 3, labeled as follows: Z indicates position; for

example, Z1, Z2, and Z3 indicate the position of V1, V2, and V3 respectively. Mass flow rate is indicated by M: M1 is the coolant mass flow rate and M2 is the process fluid mass flow rate. Pressure then is indicated by P: P1 is the pump inlet pressure and P2 is the pump discharge pressure. Finally, temperature is indicated by T: T1 and T3 indicate the tube inlet and outlet temperatures, respectively, and T2 and T4 indicate the shell inlet and outlet temperatures, respectively.

Initial understanding of knowledge requirements is developed via analysis of task (control and diagnosis) scenarios. In the following, we discuss control and diagnostic scenarios with respect to our prototype process-system, which is in the cooling mode as illustrated in Figure 4. In the figure, the process fluid is shown being cooled; the components of the system are in the following states: the pump is running at 100% capacity, V1 is 50% open, V2 is fully open, and V3 is closed. State variables have the present reading: M1 = 100,000 lbm/hr and T4 = 600°F.

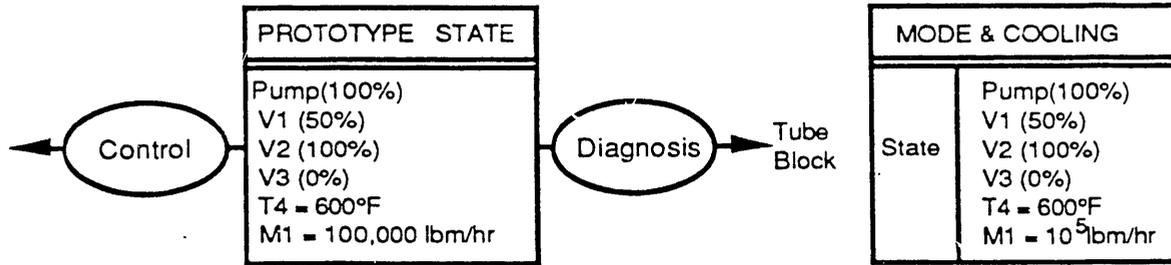


Figure 4. Prototype state vector and associated control and diagnostic results

The control task analyzes the process-system data and determines that the outlet temperature of the process fluid has exceeded a limit value of 570°F. The control response under this scenario is to increase the opening of valve V1 to 75%, as shown in Figure 4, the effect of which is to increase the cooling mass flow. Listed in Table 1 is a sample of the types of knowledge needed to perform this task.

	CONTROL KNOWLEDGE	DIAGNOSTIC KNOWLEDGE
Process-system	<ul style="list-style-type: none"> Controlling variable and its real-time and setpoint values Mass flow affects cooling Valve position affects mass flow V1 controlling component and has capacity Mechanism to actuate the valves 	<ul style="list-style-type: none"> Component operating states are unchanged Cooling mass flow decreases Flow affects cooling Heat transfer area affects cooling Heat rate relations Fault-association relations
Cognitive	<ul style="list-style-type: none"> How to recognize control requirements How to determine controlling component and its capacity How to determine state change requirements How to change controlling component state 	<ul style="list-style-type: none"> How to recognize a fault How to localize a fault How to identify a fault

Table 1. Samples of types of knowledge needed to perform control task analysis

In this scenario, as the control task is being accomplished, the diagnostic task analyzes the process-system data and concludes that a fault is present in the heat exchanger and that the faulty state is a tube block. The basis for this diagnosis is that the cooling mass flow has unexpectedly decreased, and the calculated heat rates are not in agreement with each other. Listed in Table 1 is a sample of the types of knowledge needed to perform this task. (As a side issue, think about an advanced control system that dynamically alters its control strategy based on diagnostic input.)

2.3 SOME OBSERVATIONS ABOUT KNOWLEDGE REQUIREMENTS

The above scenarios provide a sample of the kinds of knowledge used to perform control and diagnosis. Analysis of this knowledge implies some general conclusions about knowledge requirements: 1) intelligent systems

need knowledge of both the artifact reasoned about and the reasoning task itself and 2) each task contains knowledge unique to itself and knowledge that is common to some other cognitive tasks. Both of these conclusions are illustrated in Figure 5.

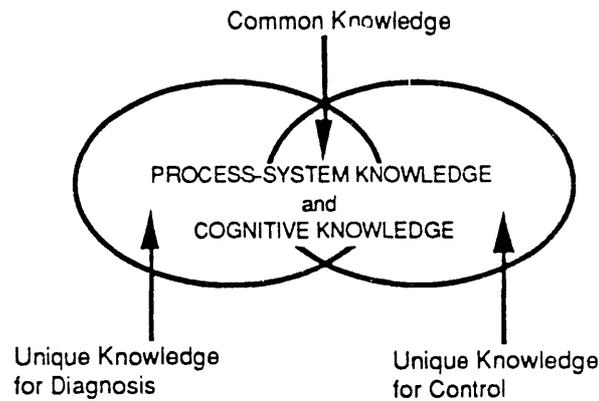


Figure 5. Characteristics of knowledge within and between tasks

Additionally, a word of caution is needed concerning knowledge resolution and its effect on task performance. That is, having knowledge of the value of a state variable, such as mass flow rate, may not be effective in the performance of diagnosis until it reaches a threshold value because of the resolution required of the variable in the task. We will illustrate this concern in the following example of the diagnosis of a tube block in the heat exchanger, the prototype process-system.

A block in the tube of the heat exchanger causes the mass flow to decrease due to the increase in flow resistance. Also the calculated heat rates are not equal because the wrong value of the heat transfer surface area is used to calculate heat rates (the design value is used instead of the actual value). However, in practice the degree of the degradation caused by the block may be so small that it is masked by the resolution of the measured parameters.

In the case of the tube block, if 1 tube out of 326 is blocked, this means that the cross-sectional area for flow and the heat transfer surface area have decreased by 1/326 of their original values. The nominal value of the mass flow rate is 170,000 lbm/hr. A single tube block will decrease the mass flow approximately 520 lbm/hr, or approximately .3%, which is well below the resolution of the mass flow rate sensor. The same reasoning applies to the heat-rate calculations based on the change in the outlet temperatures that is due to the small percentage change in the heat transfer surface area.

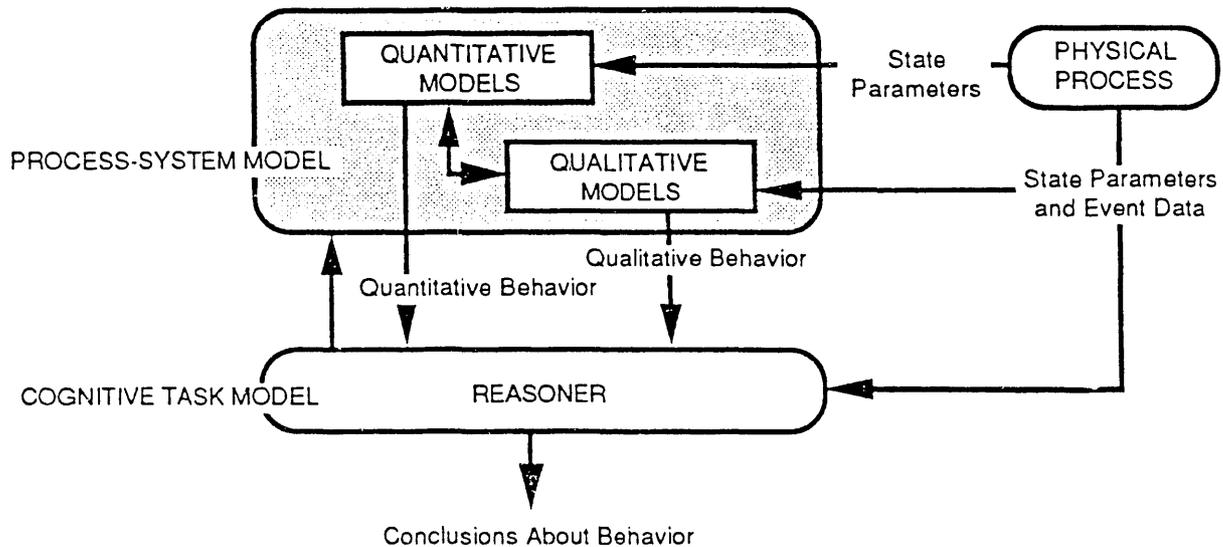


Figure 6. Interaction between process-system qualitative and quantitative models

This phenomenon suggests that in real world diagnostic aids the effect of sensor resolution must be understood in context of the knowledge requirements and the cognitive task. Knowledge must be analyzed to determine whether it has a threshold restriction. That is, the knowledge value must exceed a certain threshold value before it can be used effectively in problem solving.

Also indicated by the task scenarios is that process-system knowledge exists in at least two forms, quantitative and qualitative, which interact during the process of problem solving. This interaction is schematically illustrated in Figure 6. The process-system model shown in the figure is, therefore, composed of both quantitative and qualitative models. This model executes based on values of process-system state variables and events. The qualitative and quantitative models develop information about qualitative and quantitative behavior, respectively. In some cases, the models interact based on the information they develop. For instance, a qualitative model may be used to examine mass flow rate to determine whether it is increasing, decreasing, or constant. Based on the results of the examination, the qualitative model may signal a quantitative model to calculate the heat rates needed to provide further information for the diagnosis. Both qualitative and quantitative models will be discussed further in Section 3.0

3.0 ACQUISITION OF PROCESS-SYSTEM KNOWLEDGE

Process-system knowledge is acquired from system documentation, operation and maintenance records, text books, and system experts. In this section, we discuss our method for acquiring and modeling process-system knowledge for the development of automated fault diagnostic assistants. This method consists of two major activities--problem definition and model construction--and their associated subactivities, as illustrated in Figure 7. The problem definition activity determines initial information about the process-system. Model construction uses this information to construct information models via constraint, fault-class, and qualitative analysis.

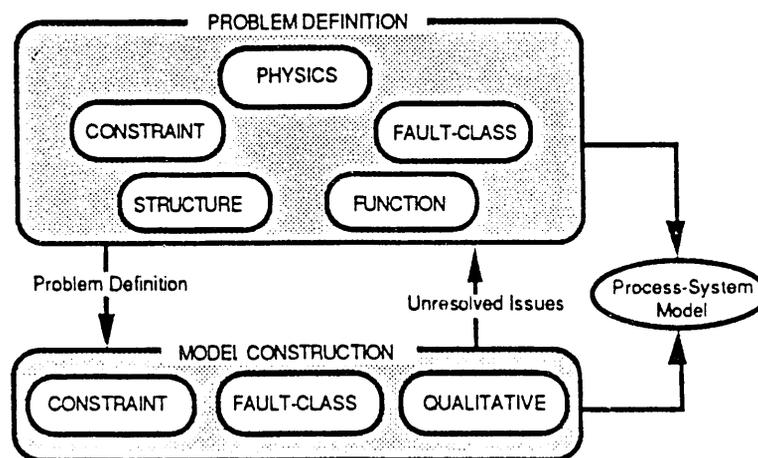


Figure 7. Process-system knowledge acquisition activities

These activities are not necessarily performed sequentially. In fact, there is a large amount of interplay between activities and between subactivities within an activity. The interplay between physics and fault-class determination serves as an example. An initial examination of the heat exchanger physics suggests that heat rate, mass flow rate, and conservation of mass and energy are sufficient to characterize the physics. However, when determining fault-classes it was determined that Asian clam infestation can be a principle cause of heat exchanger blocking and that knowledge of the fluid chemistry is necessary to determine the potential for this kind of fault.

3.1 PROBLEM DEFINITION

3.1.1 Structure Determination

Structure is the actual makeup and environment of the process-system and is expressed in terms of composition, connectivity, and object class information. Structure knowledge is used to partition the problem for analysis, provide a list of elements that directly or indirectly participate in faults, and determine system functions. Structure is expressed at different levels of abstraction described as an organized collection of subsystems, components, and parts. (Parts makeup components, subsystems are comprised of components, and subsystems are

combined to define systems.)

Composition explicitly defines the elements that makeup the process-system. The principal composition relation between structural elements is the "composed-of" relation. The prototype cooling system is composed-of "develop kinetic energy," "route fluid," and "exchange heat" subsystems. The exchange heat subsystem is composed-of the heat exchanger which is composed-of shell, tubes, inlet water box, outlet water box, cooling fluid, and cooled fluid.

Connectivity specifies how structural elements are connected via the "connected-to" relation. There are at least two types of connectivity: physical and environmental. Physical connectivity expresses the direct or explicit connection between elements. The system pump is connected-to inlet and outlet pipes, the pump outlet pipe is connected-to valve (V1) and so forth. Instrumentation connected-to components are also defined by the connectivity relation, e.g., pressure sensor P1 is connected-to the pump inlet, current sensor I1 is connected-to the pump motor windings, and speed sensor S1 is connected-to the pump shaft. Environmental connectivity expresses an implicit connection between structural elements. An uninsulated pipe may be environmentally connected-to a pump motor in that the pipe can develop surface condensation which would drip onto the motor windings and contacts.

Object class information documents generic knowledge about structural elements from a class perspective. The object class knowledge is viewed as typical knowledge associated with a structural element. An example of generic knowledge is illustrated by examining the parts that makeup components, e.g., a valve consists of a body, bonnet, stem, and disk and a pump consists of a case, shaft, seals, impeller, and motor.

3.1.2 Function Determination

Function defines the purpose of an artifact. In this context, an artifact is a process-system, subsystem, component, or component part. As an example, the function of the prototype system is to cool an internal process fluid and the function of the pump is to move river water through the tube side of the heat exchanger. Function can be expressed hierarchically.

Process-system functions are determined from the physics and structural elements of the process-system. Function determination can be viewed as a combination of both goal- and data-directed analysis. In the goal-directed analysis, the process physics is analyzed to determine the dependent and independent parameters. From this analysis, functions are derived that abstract and relate the associations between these parameters. Data-directed analysis begins with the process-system components and determines the functions of each component. For further discussion concerning the use of process-system functions in reasoning, see Stratton [5], Stricklin [6], and Moorthy [7].

3.1.3 Constraint Determination

We broadly define a constraint as a confinement or restriction. In this context, assumptions and requirements are not constraints but they do impose constraints. Constraints are developed from the problem to be solved and the capabilities of the software system to be built. They are used to specify and bound the process and relevant physics and impose requirements on the reasoning task.

CONSTRAINT TYPE	CONSTRAINT SPECIFICATION
Physics Properties	cooling fluid incompressible cooling fluid single phase
Process Parameters	temperature T1 = constant pressure P1 > pressure P2
Sensor Capability	pump sensor set [S1, P1, P2, I1] S1 accuracy ±5% S1 resolution ±10°F

Table 2. Prototype systems example of natural constraints

Constraints can be group as natural and synthetic. Natural constraints are developed from physics properties, process parameters, and sensor capability. Synthetic constraints arise in consideration of the computing environment and funding level. The synthetic constraints restrict the size and scope of the problem to be solved. In

an implicit way, the synthetic constraints impose restrictions and refinements on the natural constraints. Examples of natural constraints (see Table 2) are fluid characteristics (physics properties), constancy of and relation between state variables (process parameters), and sensor set size, accuracy, and resolution (sensor capability).

Determination of constraints is performed by an analysis of the process-system characteristics and the automation system requirements. Process-system analysis examines structure and behavior of the physical system and process physics. Automation requirements analysis determines the desired capabilities and limits of the software system. Constraints are then defined based on the process-system structure, behavior, and actual and desired limits and capabilities.

3.1.4 Physics Determination

Process-system physics consists of the quantitative relations that express the process-system thermodynamics, hydrodynamics, chemical dynamics, and electromagnetic properties. These relations are determined from an analysis of the physical system, chemical processes, constraints, faults, and reasoning requirements. The physics defines expected process behavior and specifies fundamental process models from which fault models are developed. Some of the prototype process physics are illustrated in Table 3.

PHYSICS CLASS	PHYSICS PROPERTIES	Definition of Terms: ρ = density; v = average fluid velocity at Ac; Ac = fluid cross section area; Cp = fluid heat capacity; Tout and Tin are fluid outlet and inlet temperatures; U = heat transfer coefficient between tubes and shell fluids; As = total tube surface area; LMTD = log mean temperature difference; and qshell, qtubes, and qxf are the heat rates for the shell fluid, tube fluid, and between fluids.
Hydrodynamics	Mass Flow Rate $M = \rho v Ac$	
Thermodynamics	Heat Flow Rate $q = M Cp (T_{out} - T_{in})$ $q = U As LMTD$ Conservation of Energy $q_{shell} = q_{tubes} = q_{xf}$	

Table 3. Example prototype physics properties

3.1.5 Fault-class Determination

In the context of a process-system, a fault is a condition that mars, flaws, or defects the process-system structure or process resulting in unexpected behavior. A fault can be viewed as a dynamic redesign of the process-system brought about by a degradation mechanism. Fault and degradation mechanisms are varied. The purpose of fault-class determination is to discover the types of faults and associated degradation mechanisms. Additionally, fault-class determination specifies potential location of faults within the process-system structure.

FAULT CLASS	FAULT	FAULT LOCATION
Block	Flow path Leak plug	Shell Inlet waterbox Outlet waterbox Tubes
Leak	Fluid containment Block erosion	
Heat Transfer Coefficient (U) Degradation	Tube U degradation	Interior tube surface Exterior tube surface

Table 4. Heat Exchanger fault-classes, faults, and location.

Fault-classes are discovered by component and process degradation analysis and operation and maintenance experience. Like function determination discussed above, fault-class discovery is both goal- and data- directed. In goal-directed analysis, the components and processes are analyzed to determine what can malfunction and how these malfunctions can be brought about. Analysis of experience gained from operation and maintenance provides a data-directed method for determining faults, causes, and locations.

The results of both kinds of analyses are combined to provide a specification of faults, fault mechanisms, fault-classes, and fault locations. Table 4 specifies the heat exchanger fault-classes, faults and locations. Fault mechanisms are discussed in Jarrell [8].

3.2 MODEL CONSTRUCTION

In the following we discuss model construction subactivities: constraint, fault-class, and qualitative analysis. These subactivities are used to develop both quantitative and qualitative information models of the process-system. Additionally, we provide an illustration of how the quantitative and qualitative models interact to develop on-line knowledge that is later used in diagnosis.

3.2.1 Constraint Analysis

The purpose of constraint analysis is to determine reasoning implications and requirements associated with the process-system constraints. This is necessary because the reasoner must be able to recognize its limitations in context of the constraint envelope (the set of constraints determined during problem definition). When inside the envelope, the reasoner should be able to recognize this situation and function as per design. However, when the constraint envelope has been breached, then the reasoner must recognize the breach, revise its reasoning capability, and notify the user.

Constraint analysis is performed by determining the implications of the constraints, analyzing the process-system relations that define the constraint implication, determining constraint reasoning requirements, and developing a strategy for handling constraint violations. Examples of the implication determination and analysis are illustrated in Table 5. The reasoning requirement for the single phase fluid constraint is that the reasoner must understand the relation between temperature, pressure, and fluid phase. Based on this relation and the real-time state of the fluid, the reasoner determines whether the fluid has violated the single-phase constraint.

CONSTRAINT	IMPLICATION	ANALYSIS
Single-phase fluid Tube-pressure > Shell-pressure	No boiling No shell to tube leak	Temperature-pressure relation Inter-part relation

Table 5. Example prototype constraints, implications and analysis

The final part of constraint analysis is to determine the reasoner response in the event of a constraint violation. There are several alternatives for dealing with constraint violations. The reasoning scope can be reduced such that reasoning domains not affected by the violation remain in effect and capabilities affected are shut off. Other alternatives are that the reasoner can continue reasoning with lower belief values and disclaimers, or the reasoner can shut down all together. The strategy selected is contingent on how well the constraint violation is understood with respect to the process-system physics and fault-associations.

3.2.2 Fault-Class Analysis

The fault-class analysis activity develops fault-association models in context of faults, constraints, and physical structure. This activity can also be viewed as an analysis of known fault scenarios, e.g., a block in a system component. It is performed by analyzing actual and calculated component behavior for known faults.

Fault-class analysis proceeds as follows. Select a known fault and determine the relevant physics. For each physics relation, determine the actual and calculated values of the dependent variables. Then use the actual values and their relations to determine the logical relations that exist between the calculated values. The interest in the calculated and sensed values is founded on the understanding that these are the values the operator or automated system see.

The actual values are determined using known theoretical behavior of the parametric variables based on the specified fault. If the fault under consideration is a heat exchanger tube block, then the actual heat transfer area, A_s , and therefore heat transfer, q_{xf} , will decrease. However, the calculated values are determined using design and sensed values of the variables. For the same block fault, the calculated value of q_{xf} will be based on the designed value of A_s and the designed A_s is constant in context of any fault. The following development of a heat exchanger fault-

association illustrates how fault-class analysis is used to develop fault-associations. The fault in the following illustration is a block of 20% of the tubes in the prototype heat exchanger.

The physics relations for determining the actual heat rates (subscript "a") are

$$q_{shell(a)} = M_{shell} C_p \Delta T_{shell}$$

$$q_{tube(a)} = M_{tube} C_p \Delta T_{tube}$$

$$q_{xf(a)} = U A_s LMTD$$

$$q_{tube(a)} = q_{shell(a)} = q_{xf(a)}$$

The physics relations for determining the calculated heat rates (subscript "c") are

$$q_{shell(c)} = M_2 C_p(\text{design}) \Delta T_{shell}(\text{sensor}) \quad q_{tube(c)} = M_1 C_p(\text{design}) \Delta T_{tube}(\text{sensor})$$

$$q_{xf(c)} = U(\text{design}) A_s(\text{design}) LMTD(\text{sensor})$$

$$q_{tube(c)} = q_{shell(c)} = q_{xf(c)}$$

Using the above relations, we compare the actual and calculated values to determine the relation between calculated heat rate:

$$M_{tubes} = M_1$$

$$M_{shell} = M_2$$

$$A_s < A_s(\text{design})$$

$$U = U(\text{design})$$

$$q_{tube(a)} = q_{tube(c)}$$

$$q_{shell(a)} = q_{shell(c)}$$

$$q_{xf(a)} < q_{xf(c)}$$

The analysis results in the following logical relation:

$$(q_{tube(c)} = q_{shell(c)} < q_{xf(c)}) \rightarrow \text{tube block.}$$

The tube block fault has multiple diagnoses. One diagnosis is that the designed cross-section flow area has been reduced or a second is that a previously existing leak is plugging.

We recognize that this analysis is not complete and that there are possibly other faults that could develop the same relation between heat rates. Therefore, a more appropriate expression is

$$(q_{tube(c)} = q_{shell(c)} < q_{xf(c)}) \rightarrow \text{tube block or unknown.}$$

Relations developed via this technique can provide direct knowledge of a fault as shown in the above expression or indeterminate knowledge of a fault as the following illustrates. Analysis of the heat-rate expressions for blocks in either the inlet and outlet water boxes determines that for these faults the heat rates are equal. For a non-faulted heat exchanger, the heat rates are also equal. Therefore, logical expressions for these fault-classes must be preceded with the qualifier "possible":

$$(q_{tube(c)} = q_{shell(c)} = q_{xf(c)}) \rightarrow \text{possible inlet water box block}$$

$$(q_{tube(c)} = q_{shell(c)} = q_{xf(c)}) \rightarrow \text{possible outlet water box block.}$$

3.2.3 Qualitative Analysis

The purpose of qualitative analysis is the same as that for fault-class analysis, i.e., to derive fault-association models that specify process-system behavior in context of faults, constraints, and physical structure. This activity is performed by developing qualitative models from quantitative models and then analyzing the qualitative models in context of constraints and faults. Development of the qualitative mass flow-rate relation as it applies to a generic heat exchanger demonstrates this process. The quantitative relation for mass flow is

$$M = \rho v A_c.$$

Based on the constraint that ρ is a constant, the mass flow relation and its derivative can be qualitatively expressed as

$$M = [v A_c]$$

$$\partial M = [v \partial A_c + A_c \partial v].$$

The " ∂ " notation denotes the qualitative time derivative and expressions between brackets, $[\]$, are evaluated in a qualitative sense [9]. The fluid velocity in a heat exchanger changes only as a consequence of a change in area, not as a result of the device adding or removing kinetic energy in other ways, which allows the velocity term to be eliminated. The final expression for the qualitative mass flow derivative is then

$$\partial M = \partial A_c.$$

The above expression specifies a causal relation between the fluid mass flow rate and the flow cross-section area. Analysis of this expression with the previously determined fault classes results in the following logical expressions:

$(\partial M = 0) \rightarrow (\partial A_c = 0)$:normal behavior

$(\partial M < 0) \rightarrow (\partial A_c < 0)$:faulted behavior

$(\partial M > 0) \rightarrow (\partial A_c > 0)$:faulted behavior.

Analysis of the relation between the fault-classes and the heat exchanger structure results in

$(\partial A_c < 0) \rightarrow$ block of a design flow path or plug of an existing leak

$(\partial A_c > 0) \rightarrow$ leak in a design flow path or erosion of an existing block.

3.2.4 Quantitative and Qualitative Model Interaction

The above methods of knowledge acquisition have resulted in a set of quantitative (physics relations) and qualitative (constraint relations and fault-associations) relations that specify the process-system behavior during normal and faulted operation. The following illustrates how these models might interact during the operation of the prototype system.

Initially, prior to time t_2 , both sets of models are quiescent. At t_2 , the qualitative model set is activated to analyze the new state of the mass flow rate, $M_1(t_2)$, as shown in Figure 8. It is determined that the qualitative mass flow derivative, ∂M_1 , is less than zero which implies that a fault is present, $\text{fault}(\text{present}, t_2)$. The qualitative model sends its diagnostic findings to the task reasoner and signals the quantitative model that a fault is present.

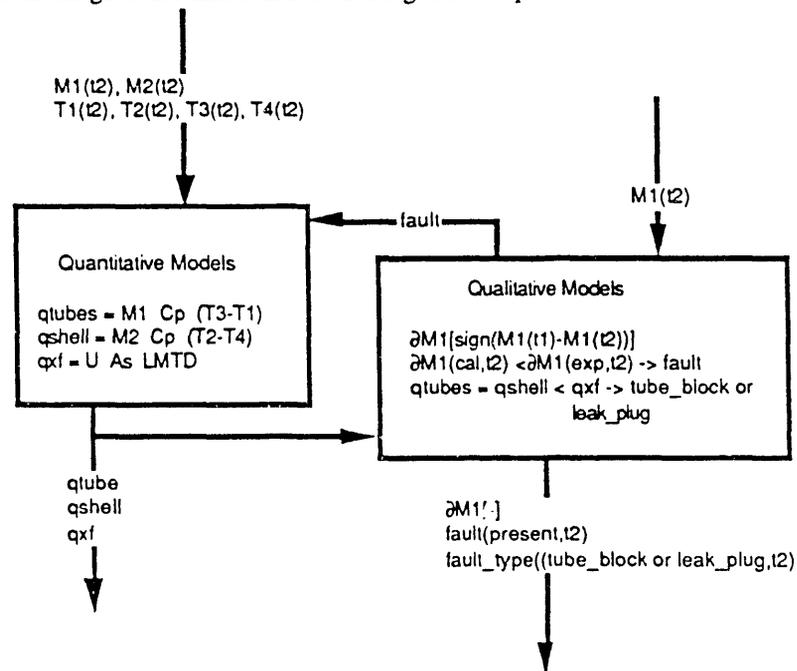


Figure 8. Interaction between quantitative and qualitative model during a block fault

The quantitative model calculates heat rates using the complete heat exchanger state vector, $(M_1(t_2), M_2(t_2), T_1(t_2), T_2(t_2), T_3(t_2), T_4(t_2))$. The heat-rate information is sent to both the qualitative model and the task reasoner. The qualitative model evaluates the new state information to determine additional diagnostic information. Qualitative heat-rate analysis indicates that there is either a block in the tubes or plugging of a previously existing leak. This new information is sent to the task reasoner and both models return to a quiescent mode.

4.0 REPRESENTATION SCHEME REQUIREMENTS

Subsequent to process-system knowledge acquisition we develop the intelligent diagnostic aid conceptual design which is then used to specify the representation scheme requirements. In this section, we illustrate elements of the conceptual design and discuss what we mean by representation scheme requirements and how these requirements relate to the conceptual design and how they are developed.

4.1 REPRESENTATION SCHEME ISSUES

By representation scheme we mean a language used to implement an information processing task in software. (See Levesque [10] for a broad general discussion of knowledge representation and reasoning.) The representation scheme must provide a capability for documenting and executing both qualitative and quantitative models. This means that it must provide a capability for representing and organizing information and methods for performing inference and controlling reasoning. These capabilities, shown in Table 6, specify the elements that makeup a representation scheme.

CAPABILITY	ELEMENT	EXAMPLES
Documentation	Representation	Structure Function Behavior Task know how
	Organization	Agents Hierarchies
Reasoning	Inference	Implication Inheritance Discrepancy Fault-Association
	Reasoning Control	Fault Recognition Fault Localization Fault Identification Fault Specification

Table 6. Representation scheme capabilities, elements, and examples

Fault diagnosis knowledge and reasoning requirements place constraints on the elements of the representation scheme. The representation element defines the methods for recording and interpreting information based on a defined syntax and semantics. This element must allow for documenting the process-system structure, function, behavior, and knowledge of how to diagnose. The organization element provides methods for organizing and abstracting information. It must allow for the specification of intelligent agents and agent hierarchies. The inference element provides methods for knowledge computation and must provide the capability for performing implication, inheritance, discrepancy analysis, and fault- association reasoning. The reasoning control element provides methods for determining what computations to perform and when to perform them. The remainder of Section 4.0 discusses in some detail the elements of the fault diagnostic representation scheme. The discussion for the most part will be in context of the prototype system previously defined.

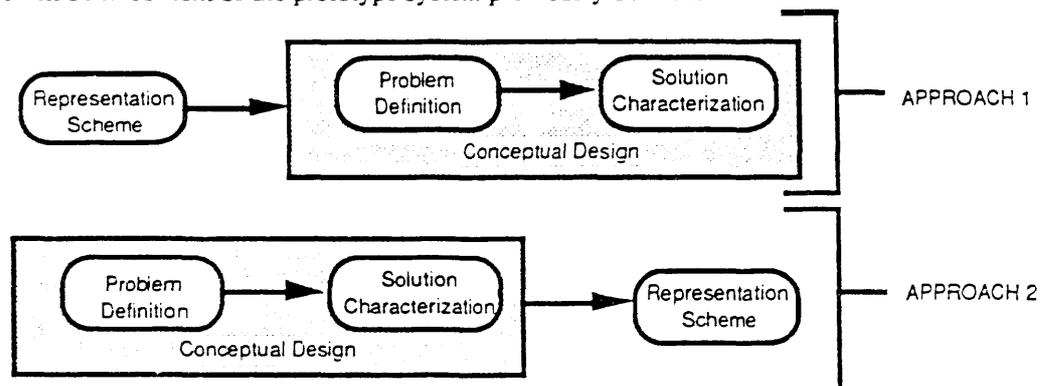


Figure 9. End points of the representation scheme selection spectrum

There are a large number of languages, shells, and tools--i.e., representation schemes--available for implementing information processing systems. Each scheme potentially provides a different depth and breath of representation scheme elements (not all representation schemes are equal). This situation suggests that there are multiple ways, a spectrum of approaches, in which to select a representation scheme to implement intelligent diagnostic aids. The two ends of the spectrum are illustrated in Figure 9.

In Approach 1, the representation scheme is acquired prior to determining what information processing task is to be automated. In this approach, the problem and associated solution targeted for automation do not levy constraints on the representation scheme selection. In Approach 2, the conceptual design is developed and analyzed to identify representation scheme requirements. Then the representation scheme is acquired.

The representation scheme selected for implementing a software system constrains the scope, capability, flexibility, and efficiency of the software system. In Approach 1, the representation scheme biases the development and implementation of the problem solution. In Approach 2, the problem and conceptual solution bias the representation scheme. It is our perspective that Approach 2 should be used when developing intelligent diagnostic aids. Approach 1 is analogous to purchasing a cross-cut saw and then deciding that the purpose of the saw will be the cut down trees. Whereas in Approach 2 you would first determine that the purpose is to cut down trees and then specify the purchase of a chain saw.

4.2 REPRESENTATION ELEMENT

The representation element defines the methods for recording and interpreting information based on a defined syntax and semantics. It is used to document the process-system structure, function, behavior, and knowledge of how to diagnose. Table 7 lists the categories of process-system information to be documented and includes examples of each. The "How-to-Diagnose" category contains two types of information, as shown in the table.

INFORMATION CATEGORY	INFORMATION EXAMPLE
Structure	Pump, Valves, Pipes, and Heat Exchanger
Function	Control Fluid Heat, Develop Fluid K.E., Route Fluid, and Exchange Heat
Behavior	Heat Rates, Mass Flow, and Fault Associations
How to Diagnose	1. Fault Recognition, Localization, Identification, and Specification 2. Discrepancy Analysis, Fault-Association, Meta-Analysis

Table 7. Categories of process-system information

4.2.1 Information to be Represented

Two types of information need to be represented. The first is information about diagnosis tasks: fault recognition, localization, identification, and specification. The second is information about how tasks are accomplished and includes discrepancy analysis, fault-association analysis, and meta-analysis. The relation between these two types of information is illustrated by examining how discrepancy analysis is used in the fault-recognition task. As the name implies, the purpose of the fault-recognition task is to determine the presence of a fault in the process-system. Discrepancy analysis is one means of determining a fault's presence. This type of analysis evaluates (or simulates), based on real-time state vector information, the state of the process-system. It then compares this simulated state to the actual or expected state to determine whether there is any disagreement which indicates the presence of a fault.

4.2.2 Representation Methods

There are many methods for representing information. The methods that we use are summarized in Table 8 and include a qualitative calculus, quantitative calculus, structured logic, and agent objects. Each method has a set of features defined as a set of operators and operands.

The quantitative calculus representation uses the real number space to define its operands, i.e., numbers from negative infinity to positive infinity. Operators include addition, subtraction, multiplication, differentiation, integration, and etc. The quantitative relation for the mass flow rate, $M = \rho v A_c$, is an example of information represented in a quantitative calculus.

There has been extensive work done in the development of qualitative calculus [11]. The theory of a qualitative calculus is an ongoing area of research in artificial intelligence. The qualitative calculus we are presently using is based on the calculus specified by de Kleer and Brown [9]. For a discussion on how we interpret and use their qualitative calculus, see Stratton and Jarrell [1]. This representation has three symbols in its number space: -1, 0, and 1. The 0 symbol refers to values of 0, -1 symbol refers to values less than zero, and the 1 symbol refers to numbers greater than zero.

REPRESENTATION METHOD	METHOD FEATURES	
Quantitative Calculus	Number space Operators Example	$-\infty$ to $+\infty$ $+, -, /, *, dx, \text{integration, etc.}$ $M = \rho v A_c$
Qualitative Calculus	Number space Operators Example	$-, 0, +$ subset of quantitative operators $\partial M = \partial A_c$
Logic	Truth space Operators Example	true, false conjunction, disjunction, negation, and implication $T > 328F \text{ and } P = 100\text{psi} \rightarrow \text{state}(H_2O, \text{steam})$
Agents	Symbol space Features Example	name, attributes, and relations develop K.E. agent

Table 8. Representation methods and their features

The operators in the qualitative representation are a subset of the quantitative operators. Because a qualitative calculus does not have magnitude certain quantitative operation can not be performed, e.g., the addition of a $X = [-1]$ and $Y = [1]$. An example of a qualitative expression is the qualitative time derivative of the mass flow equation: $\partial M = \partial A_c$. This expression is interpreted to mean that the direction of the change in the mass flow is the same as the direction in the change of the cross sectional flow area.

The logic representation is Prolog, a version of predicate calculus. Its value space consists of true and false with operators of conjunction, disjunction, negation, and implication. The following expression is an example of a logic expression:

$T > 328F \text{ and } P < 100\text{psi} \rightarrow \text{state}(H_2O, \text{steam})$.

This expression is interpreted to mean that H_2O molecules are in the state of steam when the environments temperature is greater than 328°F and the pressure is less than 100 psi.

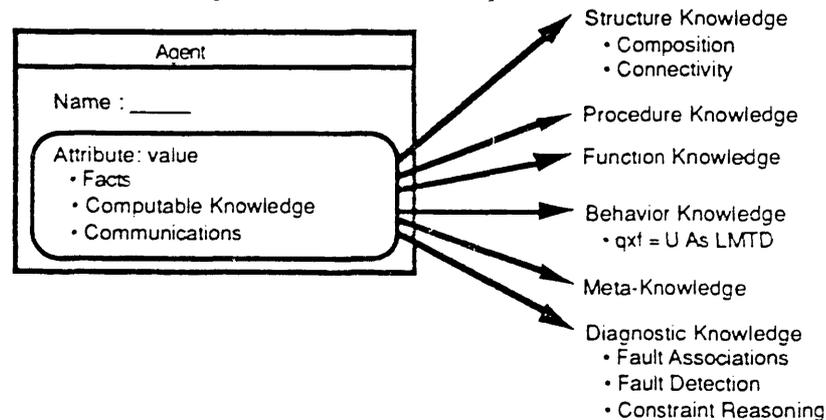


Figure 10. A generic process-system agent

An agent representation method consists of a symbol space in which agents are defined with features consisting of a name, attributes, value, and relations. An example of an agent is the "Develop K.E." agent. Figure 10 illustrates a generic agent containing process-system information.

4.3 ORGANIZATION

The organization element provides methods for abstracting and organizing information. Organization is developed around the notion of intelligent agents and hierarchies of agents. Agents were discussed in the previous section.

Hierarchies organize information based on object description and class abstraction. There are several kinds of hierarchies and for each there are multiple ways in which the hierarchy can be defined, as shown in Stratton [5], Mesarovic [12], Chandrasekaran [13], Minsky [14], and Patil [15]. Structure and content of hierarchies are influenced by the purpose of the hierarchy and the modeler's perspective of the domain to be modelled. We presently define a hierarchy as a tree structure in which each parent node has one or more children and, except for leaf nodes, each child has one parent (leaf node connections are discussed later). We use both component object (based on object description) and process-system function hierarchies (based on class abstraction).

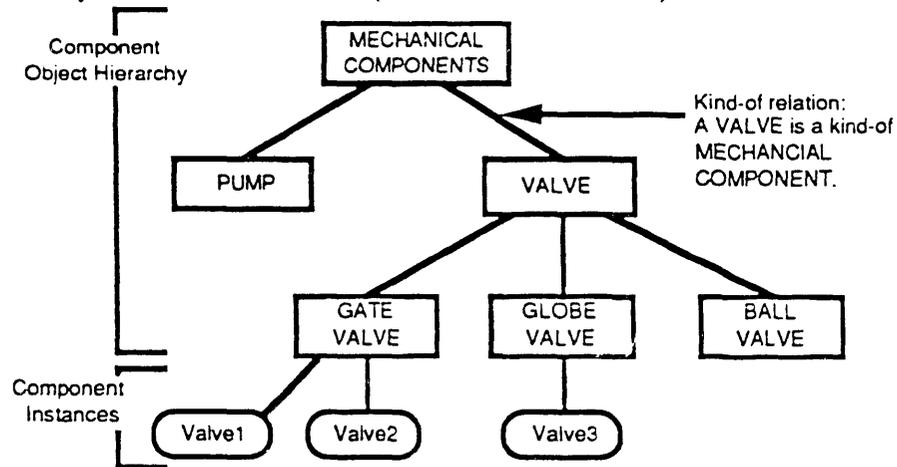


Figure 11. Example of an object description hierarchy based on components

Object hierarchies are used to organize generic taxonomical knowledge about component classes and instances in a "kind-of" relationship, e.g., a valve is a kind-of mechanical component as illustrated in Figure 11. Component classes describes object types such as pumps and valves and component instances describe actual objects that exists somewhere in the real world (valve 1, valve 2, and valve 3). The purpose of the object hierarchy is to organize generic information about component classes and specific information about component instances. The component class hierarchy is relatively static and changes only when new components classes are determined or when new components are constructed.

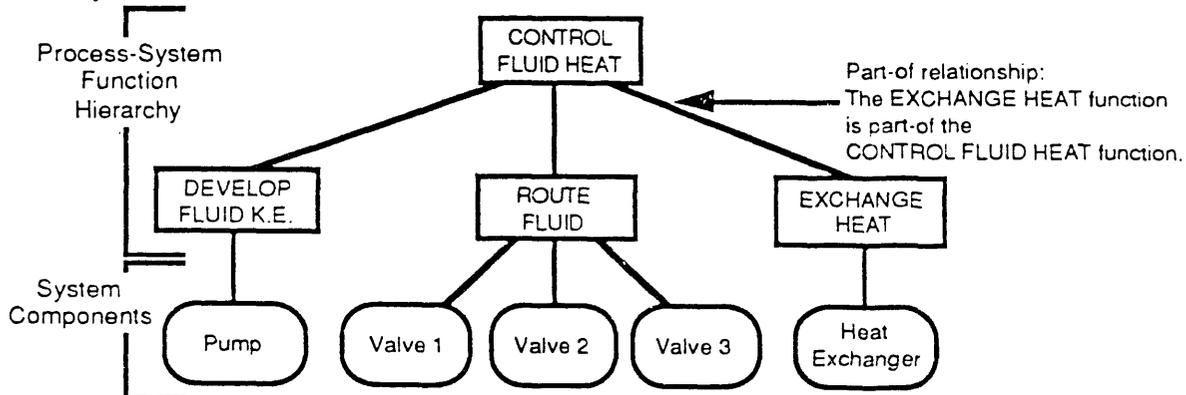


Figure 12. Example of a class abstraction hierarchy based on function

Function hierarchies organize knowledge around process-system function in a "part-of" relationship, e.g., the function to exchange heat between the process fluids is part-of the function to control fluid heat (Figure 12). This hierarchy specifies the function and sub-functions of the process-system, specifies their relationships, and as-

sociates system components to ground functions. These hierarchies are developed by decomposing the physical system into functions and subfunctions, generally based on the notion of system and subsystem. Function decomposition proceeds until ground functions are determined and specified. A ground function is the function just before the leaf nodes. Route Fluid is a ground function to which the leaf nodes Valve 1, Valve 2, and Valve 3 are connected.

Leaf nodes are the physical components that combined to provide for the functions specified at the ground level. Leaf nodes can be further functionally decomposed within the node itself and be expressed as functions based on physical parts, e.g., the pump can be modeled as a motor (develop kinetic energy), bearings (stabilize), and impeller (develop pressure differential).

The function hierarchy is unique for each process-system. The structure once defined is static except for physical system design changes. However, values of the function attributes are dynamic, e.g., performance requirements and state values. It is expected that, during system operation, components change state (on, off, failed, etc.) based on performance demands and component condition.

We view object hierarchies as "libraries of knowledge" containing generic information about objects and function hierarchies as models of "designed systems" knowledge containing information about a real world process-system (Figure 13).

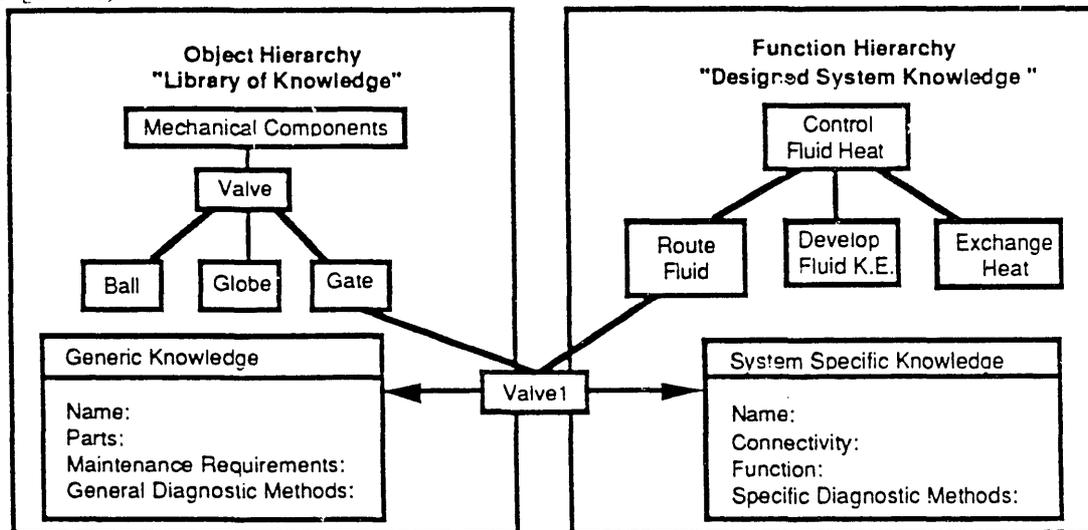


Figure 13. Relation between object and function hierarchies

Component instances exist in both hierarchies. Valve 1 in the object hierarchy, for instance, is the child of the "gate valve" node while in the function hierarchy it is the child of the "route fluid" node. Valve 1, as an object leaf node, contains information about the generic structure and behavior of the valve. However, as a function leaf node, it contains knowledge concerning its function in the designed system as well as physical connections, real-time state, and behavior knowledge. Component nodes can be viewed as the intersection of the knowledge contained in each type of hierarchy.

4.4 INFERENCE

The inference element provides methods for knowledge computation by means of implication, inheritance, discrepancy analysis, and fault-association reasoning. They are used to draw conclusions and develop new knowledge from existing knowledge. The following discusses generic and task dependent inference methods. Generic inference methods are fundamental methods of inference and can be of use in a wide variety of cognitive tasks. Task dependent inference methods for the most part are specific inference methods used in a single or a small set of related cognitive task.

4.4.1 Generic Inference Methods

Implication and inheritance are generic inference methods (reasoning task independent). Implication is a logic operation that determines the truth of a logic expression and is either goal or data directed. Goal-directed inference determines the truth of a goal by determining the truth of its conditions. If the conditions are true, then so

is the goal; however, if any condition is false, then the truth of the goal is not known and the statement is false. The general form of a goal statement and an example are as follows:

General Form: (goal) is implied by (conditions)

and, the example,

Route Fluid is in heat-exchanger-header mode IS IMPLIED BY
 valve 1 is in the open state AND
 valve 2 is in the open state AND
 valve 3 is in the closed state.

Data-directed inference executes in the reverse of goal-directed. If a data clause is determined to be true, then the reasoner fires to determine whether any conclusion based on the data clause can be set to true. The general form of a logical data statement and an example are as follows:

General Form: (data) implies (conclusion)

and, the example,

Valve 1 is in the open state AND
 Valve 2 is in the open state AND
 Valve 3 is in the closed state IMPLIES
 Route Fluid is in heat-exchanger-header mode.

Inheritance inference propagates knowledge between classes, subclasses, and instances. The valve class hierarchy shown in Figure 11 is used to illustrate this inference method. The class Valve has knowledge of generic valve structure and optional implementation features of the structure (see Table 9). The parts of a valve are body, bonnet, disk, stem and operator. Implementations features of a valve disk are gate, ball, or plug.

OBJECT	ATTRIBUTE	VALUE
Valve	Parts Disk Operator	body and bonnet and disk and stem and operator gate or ball or plug pneumatic or manual or motor
Gate Valve	Parts Disk Operator	body and bonnet and disk and stem and operator gate pneumatic or manual or motor
Pneumatic Gate Valve	Parts Disk Operator	body and bonnet and disk and stem and operator gate pneumatic

Table 9. Inheritance of object structure and implementation features

By assigning "Gate Valve" as a sub-class of the "Valve" class, it inherits the valve class structural knowledge and modifies it as appropriate. Knowledge of the generic parts remains the same; the disk-type attribute is defined as a gate, and the operator attribute is a list of potential values. Valve 1 is represented as an instance of a pneumatic gate valve. Therefore, Valve 1 inherits valve class and gate valve sub-class knowledge and modifies the operator type (pneumatic).

4.4.2 Task Dependent Inference Methods

Discrepancy analysis and fault association are inference methods inherently specified by the fault-diagnosis reasoning task [3, 16, 17, 18, 19]. Discrepancy analysis evaluates the relation between the actual and the expected state in order to infer the presence of a fault as illustrated below:

$state(expected) \neq state(calculated) \rightarrow fault(present)$

$\partial M(calculated, t1) < \partial M(expected, t1) \rightarrow fault(present)$.

Fault-association inference is used to determine the location and cause of the fault using expressions that relate object state to fault state. Fault associations are predetermined implications that relate process-system physics, faults, and structure. An example of fault-association inference is

$q_{tube} > q_{shell} \text{ and } q_{xf} > q_{shell} \rightarrow fault(inlet_waterbox, leak_to_atm)$.

4.5 REASONING CONTROL

The reasoning control element provides methods for determining which computations to perform and when to perform them. The following discusses a generalized method for reasoning about faults and an example.

4.5.1 Generalized Reasoning Method

It is our belief that fault diagnosis is a combination of data and goal-directed reasoning and iterates between them based on the present state of the diagnosis. Fault diagnosis consists of subtasks that recognize, localize, identify, and specify the fault [1]. Each of these subtasks employs inference methods for reasoning about process-system knowledge to determine conclusions about the diagnostic state of the process-system.

The function hierarchy can be generalized as three levels of agents: top, intermediate, and ground agents (Figure 14). Diagnostic subtasks take place at different levels within this generalized structure. Each subtask is characterized by the type of analysis principally employed to perform it. Fault recognition is performed using discrepancy analysis, fault localization and identification are performed via fault-association analysis and meta-analysis, and fault specification is performed using meta-analysis and procedure look up.

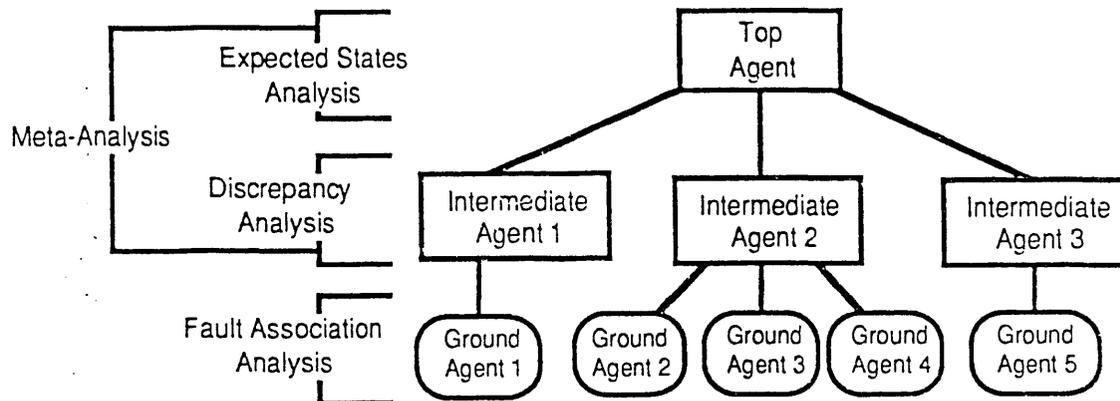


Figure 14. Generalized hierarchy of agents for performing diagnosis

The diagnosis cycle is initiated either by the notification of process-system operation requirements change or state vector collection. Initiation takes place at the top agent if a new operations requirement is received or at an intermediate agent if a state vector is collected. In either case, the diagnosis aid activities fault recognition by determining the expected state of the process-system and comparing it to the calculated state. If a discrepancy exists, then lower intermediate or ground agents are notified to initiate fault localization, depending on where the discrepancy was determined.

The notified agents generate and analyze process-system information concerning the diagnosis. The diagnostic findings of these agents are then analyzed by higher level agents to determine the diagnosis. The diagnosis is either partially or uniquely identified. If it is partially identified, the partial diagnosis is specified along with data collection methods that can be used to further develop the diagnosis. If the diagnosis is uniquely identified, then it is specified and the diagnostic aid is quiescent.

4.5.2 An Example of Fault-Diagnostic Reasoning

Initially, time t_0 , the diagnostic aid is in a quiescent state polling the outside world to determine changes in system operation requirements or state variables. For this example at t_1 , the diagnostic aid is informed that the pump has been switched from mode 1 to mode 2 (Figure 15). A mode change specifies a change in operational requirements. The mode change is interpreted by the "control fluid heat" agent which informs lower level agents of expected process-system state changes. The "Develop Fluid K.E." agent is then informed to expect the pump speed to double and the "Exchange Heat" agent is informed to expect the mass flow rate at t_1 , $M_1(t_1)$, to increase to 1.7 times the present mass flow rate $M_1(t_0)$.

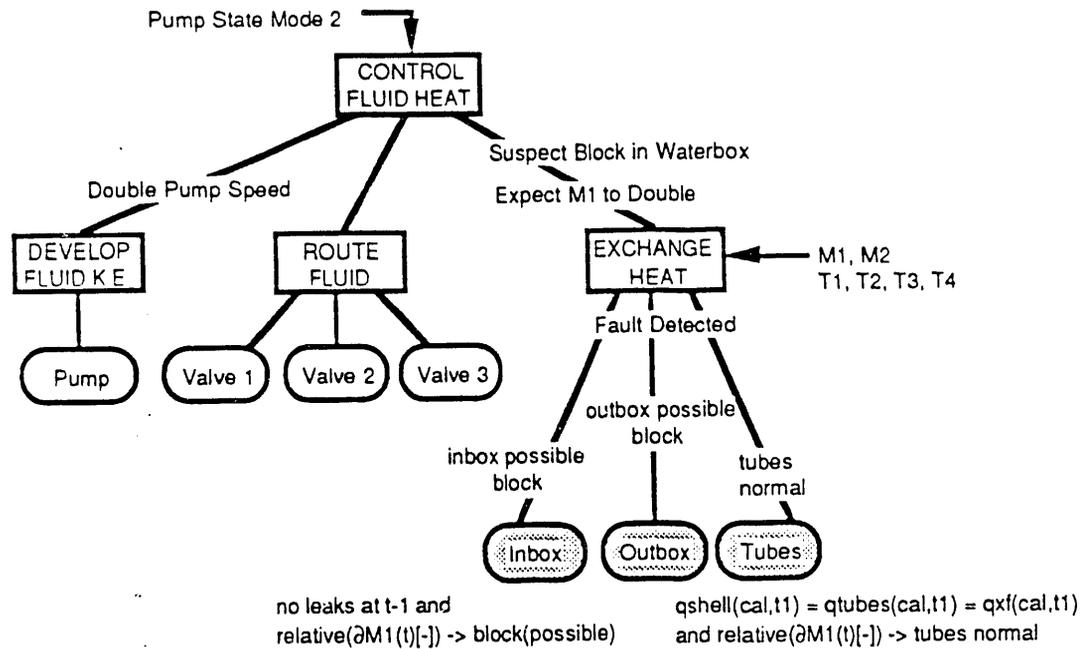


Figure 15. Prototype activity during fault diagnosis

The "Develop Fluid K.E." agent evaluates the new pump speed, motor current, and differential pressure against expected behavior and concludes that the pump is operating as expected. However, the "Exchange Heat" agent recognizes the existence of a fault when it determines that the new mass flow rate is 1.2 times the old mass flow rate instead of the expected 1.7.

Expected Change: $M1(t1)/M1(t0) = 1.7$

Calculated Change: $M1(t1)/M1(t0) = 1.2$

$M1(t1)/M1(t0)[calculated] \neq M1(t1)/M1(t0)[expected]$.

After recognizing a fault presence, the "Exchange Heat" agent sends a message to subagents to initiate fault localization. Each subagent generates additional state data which is analyzed to determine fault information. State generation and analysis is similar for both inbox and outbox agents. Each box agent determines that, based on the relative change in M1, there is a potential for either the flow path being blocked or a previous leak being plugged. Analysis of history implies that there are no leaks which is used to draw the conclusion that either or both waterboxes are possibly experiencing blocking in their flow paths:

State Generation:

$$\Delta M1(t1, relative) = M1(t1)/M1(t0)[calculated] - M1(t1)/M1(t0)[expected] = -1$$

State Analysis:

$$(\Delta M < 0) \rightarrow (\Delta A_c < 0)$$

$(\Delta A_c < 0) \rightarrow$ block of a design flow path or plug of an existing leak

history data -> no leaks

possible block and possible plug and no leak -> possible block.

The tube subagent also performs state generation and analysis. Tube mass flow rate analysis alone is not enough to draw any conclusions. However, the tube agent also evaluates and analyzes the heat rate relations. The heat rate for the tube and shell fluid are determined along with the rate of heat exchange between the fluids. Comparison of the heat rate is as expected, i.e., the magnitudes of all are equal. The tube agent concludes that it does not contain a fault:

State Generation:

$$q_{shell}(t1) = M2(t1) C_p (T2-T4)$$

$$q_{tube}(t1) = M1(t1) C_p (T3-T2)$$

$$q_{xf}(t1) = U A_s LMTD$$

State Analysis:

$\partial M1(t1, \text{relative}) \rightarrow$ possible block or leak plug

history data \rightarrow no leaks

$q_{\text{shell}}(t1) = q_{\text{tube}}(t1) = q_{\text{xf}}(t1) \rightarrow$ no block

possible block or leak and no leaks and no blocks \rightarrow no fault.

Each subagent informs the "Exchange Heat" agent that its analysis is complete. The "Exchange Heat" agent cannot identify the fault with certainty based on the collected data and generated information. It concludes that there is a block fault and that the fault is either in one or both of the waterboxes. The "Exchange Heat" agent informs its parent agent of its conclusions. The "Control Fluid Heat" agent informs the user of the diagnosis and suggests some non-intrusive non-destructive methods for collecting further data that can be used to develop a more specific diagnosis.

5.0 SUMMARY

Presented here is our methodology for developing automated aids for diagnosing faults in complex physical systems, e.g. reactor cooling system. We undertook this research at the Pacific Northwest Laboratory (PNL) for the U.S. Department of Energy and the U.S. Nuclear Regulatory Commission. We have designed these aids as multilevel-multiagent diagnostic aids based on principles that should be generally applicable to any complex system. In this methodology, "multilevel" refers to information models described at successive levels of abstraction that are tied together in such a way that reasoning is directed to the appropriate level as determined by the problem solving requirements. The concept of "multiagent" refers to the method of information processing within the multilevel model network; each model in the network is an independent information processor, i.e., an intelligent agent.

Our research in fault diagnosis grew out of our work in root-cause analysis (RCA)[1]. RCA consists of two major activities: fault diagnosis and root-cause evaluation (RCE). The purpose of fault diagnosis is to determine plant events and conditions that are associated with a specific symptom. Then RCE determines the cause of the events and condition.

We use an engineering approach for the development of intelligent aids. Requirements for intelligent systems development modify the classical life-cycle approach to software system development by requiring additional activities to be performed. In our approach, we categorize these activities as 1) knowledge requirements determination, 2) model construction, and 3) representation scheme requirements development.

The determination of knowledge requirements is fundamental to the development of any intelligent software system. This activity determines what information is needed in the problem solving activity and how it is to be used. We group knowledge for fault diagnosis into two broad groups: cognitive task and process-system knowledge.

Process-system knowledge uses plant (process-system) state and event data to develop information about the plant's behavior. Cognitive task knowledge uses general process-system knowledge and behavioral knowledge in addition to plant state and event data to develop conclusions about the plant behavior.

We chose models as the method to document and manipulate knowledge. Important issues concerning models are that 1) they capture essential aspects of the subject reality 2) in an appropriate representation 3) in order to explore properties of the reality [2]. For fault diagnosis, the essential properties are knowledge of the process-system (fixed and transient) and how to perform diagnosis. This knowledge is represented as models using quantitative calculus, a qualitative calculus, predicate logic, and intelligent agents.

Work supported by The U.S. Department of Energy under Contract DE-ACO6-76RLO 1830 and by the U.S. Nuclear Regulatory Commission

Representation scheme requirements are determined by analyzing the cognitive and process-system knowledge. These requirements generate a specification which the representation scheme should satisfy. The representation scheme should provide methods for representing and organizing information as well as methods for performing inference and reasoning control.

5.1 ACQUISITION OF PROCESS-SYSTEM KNOWLEDGE

Process-system knowledge is acquired from system documentation, operation and maintenance records, text books, and system experts. We present a method for knowledge acquisition that consists of two major activities: 1) problem definition and 2) model construction. Problem definition determines process-system structure, function, constraints, physics, and fault-classes. Model construction further develops the domain knowledge and constructs models via constraint, fault-class, and qualitative analysis.

5.1.1 Problem Definition

The problem definition activity analyzes process-system structure, function, constraints, physics, and fault-classes. Structural knowledge is determined by direction examination of the physical system. Structure is the actual makeup and environment of the process-system and is expressed in terms of composition, connectivity, and object class information. Composition explicitly defines the elements that makeup the process-system. Connectivity specifies how structural elements are connected. There are at least two types of connectivity: physical and environmental. Physical connectivity expresses the direct or explicit connections between elements. Environmental connectivity expresses implicit connections between structural elements. Object class information documents generic knowledge about structural elements from a class perspective.

Function defines the purpose of an artifact. Functions are determined from the physics and structural elements of the process-system. Function determination can be viewed as a combination of both goal- and data-directed analysis. In the goal directed analysis, the process physics is analyzed to determine the dependent and independent parameters. From this analysis, functions are derived that abstract and relate the associations between these parameters. Data-directed analysis begins with the process-system components and determines the functions of each component.

We broadly define a constraint as a confinement or restriction. Determination of constraints is performed by an analysis of the process-system characteristics and the automation system requirements. Process-system analysis examines structure and behavior of the physical system and process physics. Automation requirements analysis determines the desired capabilities and limits of the software system. Constraints are then defined based on the process-system structure, behavior, and actual and desired limits and capabilities. Constraints can be group as natural and synthetic. Natural constraints consists of physics properties, process parameters, and sensor capability. Synthetic constraints are classified as computing environment and funding-level constraints.

Process-system physics consists of the quantitative relations that express the process-system thermodynamics, hydrodynamics, chemical dynamics, and electromagnetic properties. These relations are determined from an analysis of the physical system, chemical processes, constraints, faults, and reasoning requirements.

In the context of a process-system, a fault is a condition that mars, flaws, or defects the process-system structure or process. The purpose of fault-class determination is to discover the types of faults and their potential location. Fault-classes are discovered by component and process degradation analysis and operation and maintenance experience. Fault-class discovery is both data and goal directed. In data-directed analysis, the components and processes are analyzed to determine which can malfunction and how these malfunctions can be brought about. Analysis of experience gained from operation and maintenance provides a goal-directed method for determining faults, causes, and locations.

5.1.2 Model Construction

Models are constructed based on constraint, fault-class, and qualitative analysis. The purpose of constraint analysis is to determine reasoning implications and requirements associated with the process-system constraints. Constraint analysis is performed by determining the implications of the constraints, analyzing the process-system relations that define the constraint implication, determining constraint reasoning requirements, and developing a strategy for handling constraint violations. The final part of constraint analysis is to determine the reasoner response in the event of a constraint violation.

The fault-class analysis activity develops fault-association models in context of faults, constraints, and physical

structure. It is performed by analyzing actual and calculated component behavior for known faults. Fault-class analysis proceeds as follows. Select a known fault and determine the relevant physics. For each physics relations determine the actual and calculated values of the dependent variables. Then use the actual values and their relations to determine logical relation that exist between the calculated values.

The purpose of qualitative analysis is the same as that for fault-class analysis, i.e., derive fault-association models that specify process-system behavior in context of faults, constraints, and physical structure. This activity is performed by developing qualitative models from quantitative models and then analyzing the qualitative models in context of constraints and faults.

5.2 FAULT DIAGNOSIS REPRESENTATION SCHEME REQUIREMENTS

By representation scheme we mean a language used to implement an information processing task in software. A representation scheme consists of methods for representing information, organizing information, performing inference, and controlling reasoning [3].

5.2.1 Representation

Representation defines the methods for recording and interpreting information based on a defined syntax and semantics. The representation provides for documenting the process-system structure, function, behavior, and knowledge of how to diagnose. The representation methods that we use include a quantitative calculus, qualitative calculus [4], structured logic, and agent objects [5]. Each method has a set of features defined as a set of operators and operands.

5.2.2 Organization

Organization defines the methods for organizing and abstracting information. It is developed around the notion of intelligent agents and hierarchies of agents [5,6]. Hierarchies organize information based on object description and class abstraction. Object hierarchies are used to organize generic taxonomical knowledge about component classes and instances in a "kind of" relationship. The purpose of the object hierarchy is to organize generic information about component classes and specific information about component instances. The component class hierarchy is relatively static and changes only when new components classes are determined or when new components are constructed.

Function hierarchies are organize knowledge around process-system function in a part-of relationship [7,8]. These hierarchies are developed by decomposing the physical system into functions and sub-functions generally based on the notion of system and subsystem. The function hierarchy is unique for each process-system. The structure once defined is static except for physical system design changes. However, values of the function attributes are dynamic.

We view object hierarchies as "libraries of knowledge" containing generic information of objects and function hierarchies as models of "designed systems" knowledge containing information about a real world process-system. Component instances exist in both hierarchies.

5.2.3 Inference

Inference defines methods for knowledge computation and provides the capability for performing implication, inheritance, discrepancy analysis, and fault association reasoning.

Implication and inheritance are generic inference methods (reasoning task independent). Implication is a logic operation that determines the truth of a logic expression and is either goal or data directed. Inheritance inference propagates knowledge between classes, subclasses, and instances.

Discrepancy analysis and fault-association are inference methods inherently specified by the fault diagnosis reasoning task. Discrepancy analysis evaluates the relation between the actual and the expected state and concludes the presence of a fault when there a discrepancy between the two. Fault-association inference is used to determine the location and cause of the fault using expressions that relate object state to fault state.

5.2.4 Reasoning control

Reasoning control provides methods for determining what computations to perform and when to perform them. Fault diagnosis is a combination of data- and goal-directed reasoning and iterates between them based on the present state of the diagnosis. Fault diagnosis consists of subtasks that recognized, localize, identify, and specify the fault.

Subtasks take place at different levels within the function hierarchy. The diagnosis cycle is initiated either by

the notification of a change in the requirements for the operation of the process-system or state vector collection. The reasoner when activated initiates fault recognition by determining the expected state of the process-system and comparing it to the calculated state. If a discrepancy exists, then intermediate or ground agents, depending on where the discrepancy was determined, are notified to initiate fault localization. The diagnostic findings of these agents are then analyzed by higher level agents to determine the diagnosis. The diagnosis is either partially or uniquely identified. If it is partially identified, the partial diagnosis is specified along with data collection methods that can be used to further develop the diagnosis. If the diagnosis is uniquely identified, then it is specified and the diagnostic aid is quiescent.

6.0 CONCLUSION

Our research indicates that a useful representation scheme for physical system diagnostic reasoning can be developed employing function and object hierarchies, task required inference, and task specified reasoning control. Hierarchies provide a structure for representing generic and specific physical system knowledge as well as organizing process behavioral and task knowledge. Formal modeling of the reasoning task determines the required task inference and control.

This research has focused on the use of the representation scheme for fault diagnosis. However, preliminary analysis indicates that this same scheme may be useful in automating control tasks. Also, because of the concurrency of node execution, software systems developed using this representation scheme can be parallelized and distributed.

7.0 REFERENCES

1. Stratton, RC, Jarrell, DB. A Knowledge Based Approach for Heat Exchanger Root-Cause Analysis. In Proceedings of Conference on Expert Systems Applications for the Electric Power Industry, Electric Power Research Institute, Orlando, Florida; 1989
2. Casti, JL. Alternate Realities; Mathematical Models of Nature and Man, John Wiley & Sons; 1989
3. Davis, R, Hamscher, W. Model-based Reasoning: Troubleshooting. Exploring Artificial Intelligence. Morgan Kaufmann Publishers, Inc., San Mateo, California; 1988
4. Proceedings of 1989 Workshop on Model Based Reasoning. Held in conjunction with The Eleventh Joint Conference on Artificial Intelligence, Detroit, Michigan; 1989
5. Stratton, RC. Artificial Intelligence Applications for Design Validation and Sneak Function Analysis. In Mohamed, SE and Hoover, MD (eds), Space Nuclear Power Systems, Orbit Book Company, Inc., Florida; 1984
6. Stricklin, J, Chandrasekaran, B, Bond, WE. Applying A Functional Approach for Model Based Reasoning. In Proceedings of 1989 Workshop on Model Based Reasoning. Held in conjunction with The Eleventh Joint Conference on Artificial Intelligence, Detroit, Michigan; 1989.
7. Moorthy, VS, Chandrasekaran, B. A Representation for the Functioning of Devices that Support Compilation of Expert Problem Solving Structures. In Artificial Intelligence in Maintenance, ed. Richardson, JJ. Noyes Publications, New Jersey; 1985
8. Jarrell, DB, Johnson, AB, Zimmerman, PW and Gore, ML. Nuclear Plant Service Water System Aging Degradation. Vol. 1, PNL-6560, NUREG/CR-5379, prepared for the U.S. Nuclear Regulatory Commission by Pacific Northwest Laboratory, Richland, Washington; June 1989
9. de Kleer, J, Brown, J. A Qualitative Physics Based on Confluences. Artificial Intelligence, Amsterdam; 1984; 24:7-84
10. Leveque, HJ. Knowledge Representation and Reasoning. Annual Review of Computer Science, Annual Reviews, Inc., Palo Alto, California; 1986
11. Weld, DS, deKleer, J. Readings in Qualitative Reasoning About Physical Systems. Morgan Kaufmann Publishers, Inc., San Mateo, California; 1990
12. Mesarovic, MD, Macko, D, Takahara, J. Theory of Hierarchical, Multilevel Systems. Academic Press, New York; 1970
13. Chandrasekaran, B. Towards a Taxonomy of Problem Solving Types. The AI Magazine, American Society of Artificial Intelligence, Winter/Spring; 1983; 4(1)
14. Minsky, M. The Society of Mind. Simon and Schuster, New York; 1985
15. Patil, RS. Artificial Intelligence Techniques for Diagnostic Reasoning in Medicine. Exploring Artificial Intelligence, Morgan Kaufmann Publishers, Inc., San Mateo, California; 1988
16. Lusk, EL, Stratton, RC. Automated Reasoning in Man-Machine Control Systems. In Kompass, EJ and Williams, TJ (eds), Learning Systems and Pattern Recognition. Technical Publishing Company, Indiana; 1983
17. Clancey, WJ. Heuristic Classification. Artificial Intelligence (North Holland, Amsterdam); 1985; 27(3)
18. de Kleer, J, Williams, BC. Diagnosis with Behavioral Modes. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan, Vol. 2.; Morgan Kaufmann Publishers, Inc., San Mateo, California; 1989
19. Stratton, RC., Jarrell, DB. Developing Integrated Quantitative and Qualitative Models for Reasoning About Physical Systems. In Proceedings of 1989 Workshop on Model Based Reasoning. Held in conjunction with The Eleventh Joint Conference on Artificial Intelligence, Detroit, Michigan; 1989.

END

**DATE
FILMED**

01/27/92

