# Condition-Based Maintenance: Algorithms and Applications for Embedded High Performance Computing

[1]Bonnie Holte Bennett, [2] Knowledge Partners of Minnesota, Inc., [3] George Hadden, Ph.D.

[1]University of St. Thomas, 2115 Summit Avenue OSS 301, St. Paul, MN 55105, 651-962-5509
bhbennett@stthomas.edu
[2] 9 Salem Lane, Suite 100, St. Paul, MN 55118-4700, 612-720-4960
Bbennett@kpmi.com
[3] Honeywell Technology Center, 3660 Technology Drive, Minneapolis, MN 55418, 612-951-7769
hadden@htc.honeywell.com

Abstract. Condition based maintenance (CBM) seeks to generate a design for a new ship wide CMB system that performs diagnoses and failure prediction on Navy shipboard machinery. Eventually, a variety of shipboard systems will be instrumented with embedded high-performance processors to monitor equipment performance, diagnosis failures, and predict anticipated failures. To illustrate the general principles of our design, we are currently building a research prototype of the CBM system that applies to the shipboard chilled water system.

## 1 Introduction

System health management is a concerted effort with applicability to airline maintenance operations and support, refineries and other chemical processing facilities, commercial aircraft, and sensor interface modules for the space shuttle operational maintenance system. The current effort is a three-year demonstration of concept scheduled to conclude in late 1999.

A team is accomplishing this work with members from industry, government labs, and academia. Our partners include The Georgia Institute of Technology, and PredictDLI. This paper will concern work on knowledge fusion performed by Knowledge Partners of Minnesota, Inc. (KPMI), rule-based system analysis by Graduate Programs in Software (GPS) at the University of St. Thomas, St. Paul, Minnesota, and system integration by Honeywell technology Center (HTC).

The CBM program is extremely ambitious. Eventual implementations will instrument thousands (perhaps tens of thousands) of replaceable parts on shipboard systems. Fleet-wide, thousands of embedded processors will collect millions of data points per second of data from tens of thousands of locations each. These data will be processed by local software, and the results will be passed on to local "data concentrators" (DCs) that will concentrate reports from a number of different sensors via a variety of specialized analyzers, for example DLI's Expert Alert expert system for vibration analysis. Results from hundreds of DCs per ship will be correlated at a system level in another processor, the Prognostic/Diagnostic Monitoring Engine (PDME). The result is evident: significant data loads, multiple embedded processors, and critical high performance computing needs.

We are currently designing and refining a Machinery Prognostic and Diagnostic System (MPROS) system architecture with open interfaces to provide machinery condition and raw sensor data to other shipboard systems such as ICAS (Integrated Condition Assessment System, a product from IDAX). We have implemented an Object-Oriented Ship Model (OOSM) and knowledge fusion algorithms to integrate the data at PDME level.

The knowledge fusion algorithms currently use class heuristics and Dempster/Schafer probability reasoning to scrutinize incoming reports for correlation with others. We expect to implement a Bayesian Network probability theory when sufficient data exists for a priori dependence calculations. Our current prognostic extrapolation uses worst-case scenarios for failure prediction, but next generation software will use more complex failure analysis using historical data, and learning to refine its estimates over time.

The high-performance computing issues related to this work are challenging, as described above, the data rates are extremely high, data overload is rampant, and we are working to articulate the specific challenges here.

## 1.1 MPROS

The ONR CBM system, called MPROS (for Machinery Prognostics and Diagnostics System), is a distributed, open, extensible architecture for housing multiple on-line diagnostic and prognostic algorithms. Additionally, our prototype contains four sets of algorithms aimed specifically at centrifugal chillers. These are:

1. DLI's (a company in Bainbridge Island, Washington that has a Navy contract for CBM on shipboard machinery) vibration based expert system adapted to run in a continuous mode.

2. State Based Feature Recognition (SBFR), an HTC-developed embeddable technique that facilitates recognition of time-correlated events in multiple data streams. Originally developed for Space Station Freedom, this technique has been used in a number of NASA related programs.

3. Wavelet-Neural Network (WNN) diagnostics and prognostics developed by Professor George Vachtsevanos at Georgia Tech. This technique is, like DLI's, aimed at vibration data, however, unlike DLI's, their

algorithm will excel in drawing conclusions from transitory phenomena rather than steady state data.

4 Fuzzy Logic diagnostics and prognostics also developed by Georgia Tech which draws diagnostic and prognostic conclusions from non-vibrational data.

Since these algorithms (and others we may add later) have overlapping areas of expertise, they may sometimes disagree about what is ailing the machine. They may also reinforce each other by reaching the same conclusions from similar data. In these cases, another subsystem, called *Knowledge Fusion*, is invoked to make some sense of these conclusions. We use a technique called *Dempster-Shafer Rules of Evidence* to combine conclusions reached by the various algorithms. It can be extended to handle any number of inputs.

MPROS is distributed in the following sense: Devices called *Data Concentrators* (DC) are placed near the ship's machinery. Each of these is a computer in its own right and has the major responsibility for diagnostics and prognostics. The algorithms described above run on the DC. Conclusions reached by these algorithms are then sent over the ship's network to a centrally located machine containing the other part of our system – the *Prognostic/Diagnostic/Monitoring Engine* (PDME). KF is located at the PDME. Also on the PDME is the *Object Oriented Ships Model* (OOSM). The OOSM represents parts of the ship (e.g. compressor, chiller, pump, etc.) and a number of relationships among them (part-of, proximity, kind-of, etc.). It also serves as a repository of diagnostic conclusions – both those of the individual algorithms and those reached by KF. Communication among the DC's and the PDME is done using DCOM (Distributed Common Object Modules), a standard developed by Microsoft.

The MPROS program is designed to comprise two phases. The first phase, which we have just finished, will see MPROS installed and running in the lab. The second phase will extend MPROS's capability somewhat and install it on a ship. The current plan is to install it on the Navy Hospital Ship, Mercy, in San Diego.

## 1.2 Mission

Our central mission in this project is to design a *shipwide* CBM system to predict remaining life of all shipboard mechanical equipment. However, implementation of such a system would be much too ambitious for the current project. In light of this, we have chosen to illustrate the general principles of our design by implementing it in a specific way on the Centrifugal Chilled Water System. The result of this philosophy is that occasionally we will choose a more general way of solving a problem over a "centrifugal chiller-specific" solution.

## 2 Why Centrifugals?

There were two main reasons for our choice of centrifugal chillers: System complexity and commercial applicability.

These A/C systems combine several rotating machinery equipment types (i.e. induction motors, gear transmissions, pumps, and centrifugal compressors) with a fluid power cycle to form a complex system with several different parameters to monitor. This dictates the requirement for a correspondingly complex and versatile monitoring system. Dynamic vibration signals must be acquired using high sampling rates and complex spectrum and waveform analysis. Slower changing parameters such as temperatures and pressures must also be monitored, but at a lower frequency and can be treated as scalars rather than vectors as with vibration spectra.

All of these monitored parameters and analysis techniques are combined using a versatile diagnostic system. The final product has the inherent capability of diagnosing not just the whole A/C system, but each of its parts as well, making it a potentially very useful tool for monitoring any pump, motor, gearset, or centrifugal compressor in the fleet.

Secondly, the selection of A/C system as the subject will provide a high probability of commercial applicability of the resultant monitoring system. There are a great deal of facilities industrial, military, commercial and institutional that use large centrifugal chiller based A/C systems throughout the US and the world.

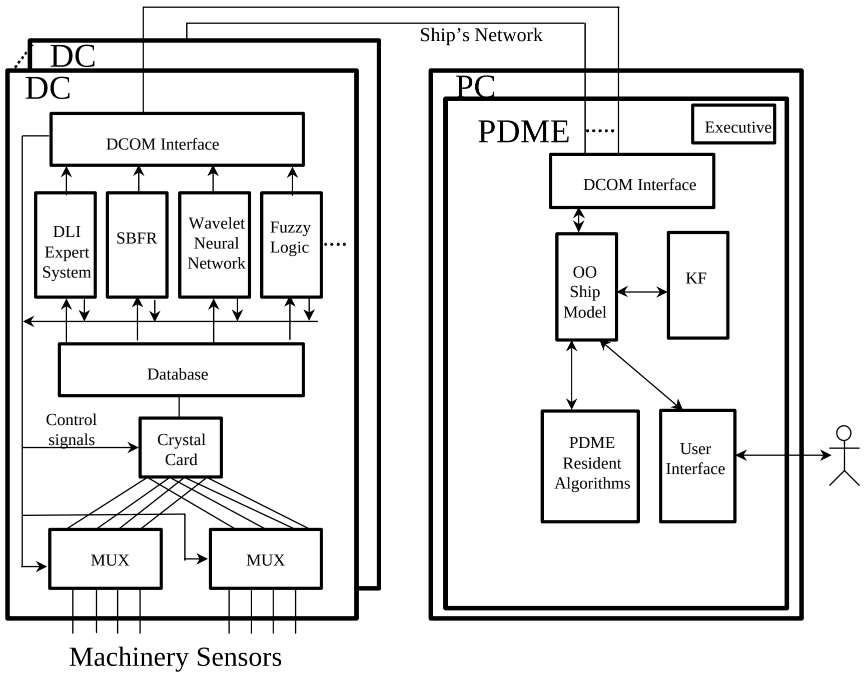


Fig. 1. The MPROS System

## 3 Software

Figure 1 shows a diagram of the MPROS system. Here we describe the various parts.

### 3.1 PDME

The Prognostic/Diagnostic Monitoring Engine (PDME) is the logical center of the MPROS system. Diagnostic and prognostic conclusions are collected from DC-resident algorithms as well as PDME-resident algorithms. Fusion of conflicting and reinforcing source conclusions is performed to form a prioritized list for the use of maintenance personnel.

The PDME is implemented on a Windows NT platform as a set of communicating servers built using Microsoft's Component Object Model (COM) libraries and services. Choosing COM as the interface design technique has allowed us to build some components in C++ and others in Visual Basic, with an expected improvement in development productivity as the outcome. Some components were prototyped using Microsoft Excel and we continue to use Excel worksheets and macros to drive some testing of the system. Communications between DC components and PDME components depend on Distributed COM (DCOM) services built into Microsoft's operating systems.

### 3.2 User interface

As shown in Fig. 2. , an interface to the MPROS conclusions has been built. The sample screen shown indicates that for machine A/C Compressor Motor 1, six condition reports from four different knowledge sources (expert systems) have been received, some conflicting and some reinforcing.

After these reports are processed by the Knowledge Fusion component, the predictions of failure for each machine condition group are shown at the bottom of the screen.

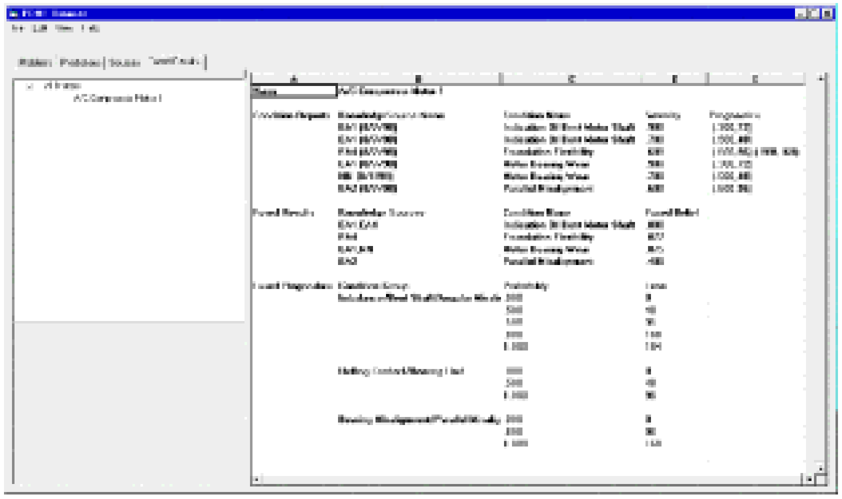


Fig. 2. MPROS User Interface

This display is updated as new reports arrive at the PDME and are accumulated in the OOSM.

## 3.3 Status

The PDME has been implemented and running since August, 1998. This work has included a number of specification, analysis and design tasks. An RF survey was completed on the USNS Comfort, a Navy Hospital ship in San Diego. A failure effects mode analysis (FMEA) was completed and used to select 12 candidate failure modes. This work is being integrated with industry standards such as Machinery Management Open Systems Alliance (MIMOSA), and related work at Penn State and BBN. Significant completed milestones include:

- Demonstrated an object-oriented Level5 Object ™ simulation of Carrier Chiller startup.
- Instrumented a lab with an optical RIMS (Resonant Integrated Micro-Structure – a type of MEMS) for temperature, and are investigating it for accelerometer.
- Installed a state of the art DLI Fulltime ™ system on CVN-72 (Lincoln), the Comfort, and at an HTC lab.
- Completed phase 1 design and implementation.
- Integrated a vibration rule-base, and state based feature recognition for diagnosis.
- Implemented knowlege fusion with prognostic reporting for “time to failure” estimates.
- Designed for integration of Wavelet Neural Net and Fuzzy Logic analysis from Georgia Tech.

## 3.4 Going Forward

Preparation plans for shipboard deployment include continued testing and monitoring, as the installed system will be disconnected from our labs for months at a time.

# 4 Object Oriented Ship Model

## 4.1 Objectives

The Object Oriented Ship Model (OOSM) is a persistent repository for machinery state information used for communication between the various prognostic and diagnostic software modules. In addition to diagnostic and prognostic reports, the OOSM also models the physical, mechanical and energy characteristics of the machinery being monitored. Exposing an integrated programming interface to all of this information eases the development of new knowledge-based algorithms for diagnostics and prognostics.

## 4.2 Object Model

An object-oriented design was chosen to provide a consistent interface to all the developers using the OOSM to store, retrieve and monitor changes to information of interest to MPROS components.

Entities in the OOSM are modeled as objects with properties and relationships to other entities. Some of the OOSM objects represent physical entities such as sensors, motors, compressors, decks, and ships while other OOSM objects represent more abstract items such as a failure prediction report or a knowledge source. Some common properties include name, manufacturer, energy usage, capacity, and location. Common relationships include “part-of”, whole and refers-to.

As part of easing the use of an object-oriented model for developers, the persistence is entirely managed in the background. In fact, save for retrieving the first object in a connected graph of objects, no understanding of the persistence mechanism is necessary.

## 4.3 Contents

While the preceding section described the form of objects in the OOSM, this section briefly describes the contents. We have modeled a portion of the information about the system under observation in the OOSM. This includes information about the motors, compressors and evaporators in the chillers we are working with. We have also modeled relationships between the failure predictions and the relevant equipment to provide for a future expert system to analyze interactions between equipment subsystems.

As we expand the scope of equipment this system is expected to monitor, the contents of the OOSM will expand to match.

### 4.4 Applications Programming Interface (API)

As mentioned in preceding sections, a consistent API for developers has been defined. Briefly, it consists of functions to retrieve specific object instances, to view the values of properties, to update their properties and relationships, and to create and delete instances. This API, based upon COM, has been used from C++, Visual Basic, and Java programs to work with the OOSM.

### 4.5 Events

An event model has been implemented for the OOSM, which allows client programs to be notified of changes to property or relationship values without the need to poll. This facility is provided by OLE Automation events, making it usable from C++, Visual Basic and Java. The Knowledge Fusion component uses this to automatically process failure prediction reports as they are delivered to the OOSM. The PDME browser also uses events to update it's display for users.

### 4.6 Database Mapping

Persistence of object state in the OOSM is implemented using a relational database. Object types are mapped to tables and properties and relationships are mapped to columns and helper tables. This mapping approach has helped in system debugging and has proven to be very reliable in operation.

### 4.7 Implementation

The implementation of the OOSM is C++ using Microsoft's Active Template Library (ATL) and Active Data Objects (ADO) library. This approach was chosen because of the control over object lifetime, performance, and reliability.

### 4.8 Status

The OOSM has been implemented and running since August, 1998.

### 4.9 Going Forward

As we prepare for shipboard deployment, we plan to exercise scenarios that will likely be encountered. For example, power supply and communications are stable in our labs but may not be the same on board the ships. Simulating the range of problems that may arise will let us improve robustness to the point of long-term unattended operation.

# 5 Knowledge Fusion

## 5.1 Objectives

This section describes the objectives for knowledge fusion.  It will start by describing the challenges presented by the data received as input for knowledge fusion.  Then, it will continue to describe the functions required to deal with these inputs.

Knowledge fusion is the coordination of individual data reports from a variety of sensors.  It is higher level than pure "data fusion" which generally seeks to correlate common-platform data.  Knowledge fusion, for example, seeks to integrate reports from acoustic, vibration, oil analysis, and other sources, and eventually to incorporate trend data, histories, and other components necessary for true prognostics.

The knowledge fusion components must be able to accommodate inputs which are incomplete, time-disordered, fragmentary, and which have gaps, inconsistencies, and contradictions.  In addition, knowledge fusion components will have to be able to collate, compare, integrate, and interpret the data from a variety of sources.

In order to this, it must provide inference control that accommodates a variety of input data, and fusion algorithms with the ability to deal with disparate inputs.

Knowledge fusion follows the general format:

1 New reports arriving to the PDME are posted in the OOSM.

2 New reports posted in the OOSM generate "new data" messages to the knowledge fusion components.

3 The knowledge fusion components access the newly arrived data from the OOSM. They perform knowledge fusion of diagnostic reports and knowledge fusion of prognostic reports.

4 Conclusions from the knowledge fusion components are posted to the OOSM and presented in user displays in the graphical user interface.

## 5.2 Implementation

To date, two levels of knowledge fusion have been implemented one for diagnostics and one for prognostics.

## 5.3 Knowledge Fusion for Diagnostics

The current approach for implementing knowledge fusion for diagnostics uses Dempster-Shafer belief maintenance for correlating incoming reports. This is facilitated by use of a heuristic that groups similar failures into logical groups.

Dempster-Shafer theory is a calculus for qualifying beliefs using numerical expressions. Specifically, given two sets of beliefs. For example, given a belief of 40% that A will occur and another belief of 75% that B or C will occur, it will concluded that A is 14% likely, “B or C” is 64% likely and there is 22% of belief assigned to unknown possibilities. This maintenance of the likelihood of unknown possibilities is both a differentiator and a strength of Dempster-Shafer theory. It was chosen over other approaches like Bayes Nets because they require prior estimates of the conditional probability relating two failures. The data is not yet available for the CBM domain.

The system was augmented by a heuristic grouping similar failures into logical groups. This facilitates processing and streamlines operation. The reason for this is that Dempster-Shafer analysis looks at each failure in light of every other failure possible, and this is required to produce the likelihood of unknown possibilities. In the CBM case, this is inadequate because this would assumes mutual exclusivity of failures, in other words, it assumes that any one failure precludes any other failures. However this is not the case in CBM, there can, in fact, be several failures at one time, and two or more of them might be independent of one another. Thus, we developed the concept of logical groups of failures. Failures, which are all part of the same logical groups, are related to each other (for example, one group might be electrical failures, another lubricant failures, etc.).

Moreover, failures within a group might be mistaken for one another, so they are logically related and should share probabilities when they are both under consideration. Note that this does not preclude multiple failures within a group to all be suspect concurrently, it simply ensures that they are tracked correctly and weighted correctly given similar failures which might be related to them.

## 5.4 Knowledge Fusion for Prognostics

Prognostics are defined in this system as time point, probability pairs, and lists of these pairs. So for example, a prognostic of (3 months, .1) would indicate that the system has a 10% likelihood of failure within 3 months time from now. Subsequently, a prognostic list of ((2 weeks, .1) (1 month, .5) (2 months, .9)) would indicate a likelihood of failure of 10% within 2 weeks, 50% at 1 month and 90% in 2 months. Knowledge fusion for prognostics is the combination of these lists of time and failure likelihoods. Our approach in phase one has been to combine the lists taking the most conservative estimate at any given time period, and interpolating a smooth curve from point to point.

So, for example, if we have a prognostic for a given component that indicates it will perform well for 3 months, then experience some trouble making it as likely to break as not by 4 months, and almost surely break by 5 months: ((3 months, .01) (4 months, .5) (5 months. 99)) and we need to combine this with another report showing that the same component will experience some small trouble at 4 1/2 months ((4.5 months, .12)), then we will ignore the second report, and stick with the first which is more conservative.

If, however, the second report indicates a much higher likelihood of failure ((4.5 months, .95)) then this report would dominate, and the extrapolation of the curve beyond this point would indicate an even earlier demise of the component that the original which would be some time after 5 months.

## 5.5 Interfaces Provided

The general incoming report format may contain the following data fields (not all reports need use all fields):

- DC ID–Identifyer of the data concentrator source of this report.
- KS ID–Identifyer of the knowledge source generating this report.
- Timestamp–Time that the report was issued.
- Machine ID–Identifyer of the machine for which this report is generated.
- Machine condition–Type of failure (e.g., motor imbalance, motor rotor bar problem, pump bearing housing looseness, etc.).
- Severity–DLI issues a severity rating of slight, moderate, serious, or extreme.
- Belief–Some of DLI’s reports have a belief value less than 1.0 (on a scale of 0 to 1).
- Explanation–There may be an explanation about the diagnosis.
- Recommendation–This may indicate what action should be taken.
- Prognostic vector–This vector of time point, probability pairs indicate projected likelihood of failure in the given time period.

## 5.6 Knowledge Fusion Outputs

Diagnostic knowledge fusion generates a new fused belief whenever a diagnostic report arrives for a suspect component. This updates the belief for that suspect component and for every other failure in the logical group for that component. It also updates the belief of “unknown” failure for that logical group for that component.

Prognostic knowledge fusion generates a new prognostic vector for each suspect component whenever a new prognostic report arrives.

### 5.7 Resident Algorithms

As the reader can see from Figure 1, the PDME has the capability to host prognostic and diagnostic algorithms. Some reasons for placing the algorithms in the PDME rather than the DC include: the algorithm requires data from widely separate parts of the ship, the algorithm can reason from PDME resident components (a model-based diagnostic and prognostic system, for instance, might use only the OOSM), and so on.

Currently, our Phase 1 system does not place any diagnostic/prognostic algorithms in the PDME – all of them run in the data concentrators.

### 5.8 Data Concentrator

The data concentrator is a open architecture ODBC compliant relational database designed to store all of the instrumentation configuration information, machinery configuration information, test schedules, resultant measurements, diagnostic results, and condition reports. The database design is a commercially available database.

The DC software is coordinated by an event scheduler. It coordinates standard vibration test and including data acquisition and communication of the results. In similar fashion, the scheduler conducts wavelet and neural network testing and analysis, and state based feature recognition routines to collect and analyze process variables. The data is processed and then sent to an expert system DLL which applies stored rules for each equipment type and derives the diagnoses. The DLL then passes the results back to the DC database. Each of the components extract information from and store data in the DC database which is configured as a database server and can be accessed by client PC's on the network. In this way, the PDME or any other client can command the scheduler to conduct another test and analysis routine.

## 6 Prognostic/Diagnostic Algorithms

### 6.1 DLI Expert System

Currently, all standard machinery vibration FFT analysis and associated diagnostics in the Data Concentrator are handled by the DLI expert system. Over the past ten years, this system has been installed in hundreds of industrial and manufacturing facilities in addition to a variety of Navy ships. In one study, it was found that the system exceeds 95% agreement with human expert analysts for machinery aboard the Nimitz class ships. The DLI expert system provides an intermediate level of sensor and knowledge fusion. The frame based rules application method employed allows the spectral vibration features to be analyzed in conjunction with process parameters such as load or bearing temperatures to arrive at a more accurate and knowledgeable machinery diagnosis.

For example since some compressors vibrate more at certain frequencies when unloaded, the DLI expert system rule for bearing looseness can be sensitized to available load indicators (such as pre-rotation vane position) in order to ensure that a false positive bearing looseness call is not made when the compressor enters a low load period of operation. This kind of knowledge and sensor fusion is found again at the PDME that ascertains the relative believability of the various diagnostic systems and derives a reasonable conclusion. It is however advantageous for the vibration expert system to utilize all available known system responses when analyzing the vibration patterns because it minimizes the necessary PDME decisions and improves overall system accuracy.

An elementary level of machinery prognostics has always been provided by the DLI expert system which since its inception, has provided a numerical severity score along with the fault diagnosis. This numerical score is interpreted through empirical methods which map it into four gradient categories.

These categories are Slight, Moderate, Serious and Extreme and correspond to expected lengths of time to failure described loosely as: no foreseeable failure, failure in months, weeks, and days of operation. This approach to prognostics was developed and improved upon during the last 10 years and was refined further for the Data Concentrator and the PDME through the introduction of believability factors for each of the diagnoses. These believability factors are based on DLI's statistical database that demonstrates the individual accuracy of each diagnosis by tracking how often each was reversed or modified by a human analyst prior to report approval.

## 6.2 Wavelet Neural Networks

The Wavelet Neural Network (WNN) belongs to a new class of neural networks with such unique capabilities as multi-resolution and localization in addressing classification problems. For fault diagnosis, the WNN serves as a classifier so as to classify the occurring faults. CMB uses a fault diagnostic system based on the WNN.

Critical process variables are monitored via appropriate sensors. Features extracted from input data are organized into a feature vector, which is fed into the WNN. Then, the WNN carries out the fault diagnosis task. In most cases, the direct output of the WNN must be decoded in order to produce a feasible format for display or action. Results of the WNN can be used to perform fault diagnosis via classification using information sucha as the peak of the signal amplitude, standard deviation, cepstrum, DCT coefficients, wavelt maps, temperature, humidity, speed, and mass

## 6.3 State Based Feature Recognition

SBFR is a technique for the hierarchical recognition of temporally correlated features in multi-channel input. It consists of a set of several enhanced finite-state machines operating in parallel. Each state machine can transition based on sensor input, its own state, the state of another state machine, measured elapsed time, or any logical combination of these. This implies that systems based on SBFR can be built with a layered architecture, so that it is possible use them to draw complex conclusions, such as prognostic or diagnostic decisions.

Our implementation of the SBFR system requires very little memory (100 state machines operating in parallel and their interpreter can fit in less than 32K bytes) and can cycle with a period of less than 4 milliseconds. It is thus ideal for embedding into the DC. We have successfully applied SBFR-based diagnostic and prognostic modules to several problems and platforms, including:

- valve degradation and failure prediction in the Space Shuttle's Orbital Maneuvering System,
- imminent seize-up in Electro-Mechanical Actuators through electrical current signature analysis and other parameters, and
- failure prediction in several subsystems (including Control Moment Gyro bearing failure) in the Space Station Freedom's Attitude Determination and Control System.

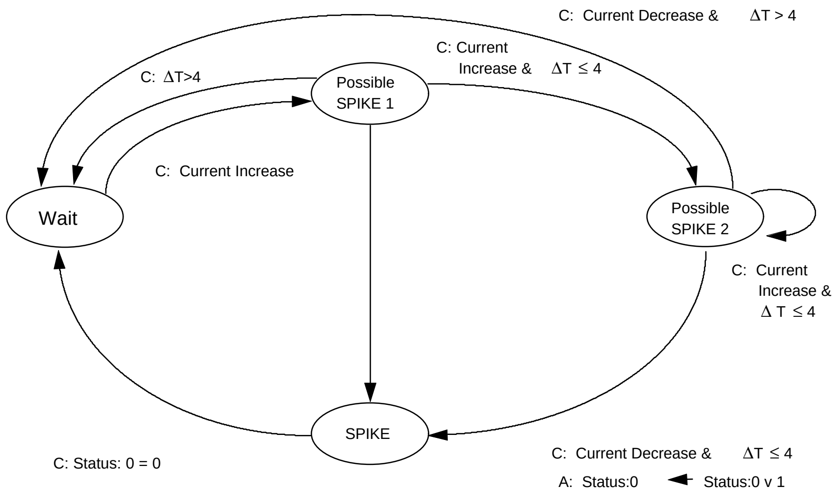
SBFR embedded in the DC will take as input the raw sensor data and the output of other algorithms (e.g., DLI's vibration analysis and FFTs) and perform trending analysis, feature extraction, and some diagnostics and prognostics on this data. Local knowledge gained by these techniques will be delivered to ICAS and the subsystems contained within ICAS for further analysis and crew alerting. Some level of local alerting (e.g., indicator lights) will also be made available. Under control of the System Executive running in the PDME under ICAS, new finite-state machines may be downloaded into the smart sensor. This will allow the behavior of the sensor to adapt to its data in appropriate ways. It will have, for instance, the capability to take a "closer look" at a problem that has been discovered.

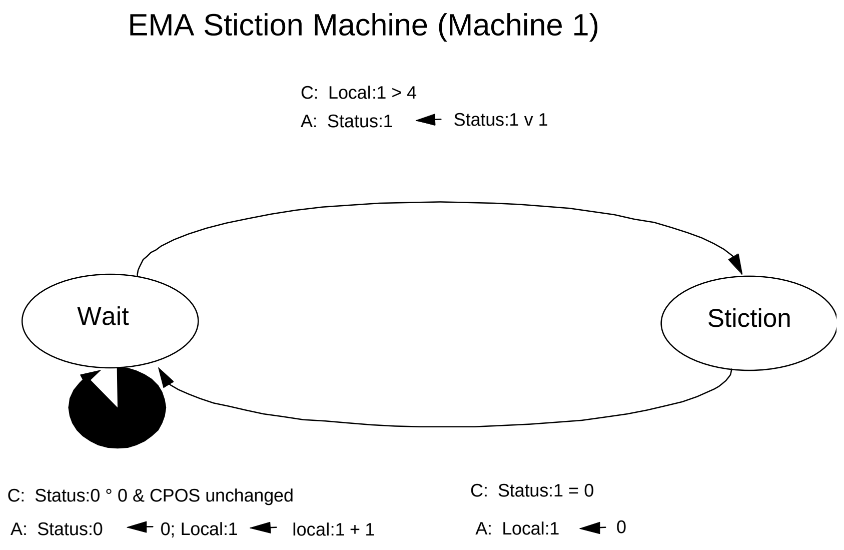
To give a sense for how SBFR works, here is an example. The two state machine system shown in Figure 3 was used to predict a seize-up failure mode in an electro-mechanical actuator (EMA). EMAs are essentially large solenoids meant to replace hydraulic actuators for the steering of rocket engines. Prediction of this fault was done by recognizing stiction in the mechanism. Stiction is recognized by the two state machines in the following way: Machine 0 recognizes spikes in the drive motor current. Machine 1 counts the spikes that are *not* associated with a commanded position change (CPOS). When the count is greater than 4, a stiction condition is flagged, and higher level software (e.g., the PDME) can conclude that a seize-up failure is imminent.

We now consider these machines in more detail. Starting with Machine 1, notice that it has two states labeled Wait and Stiction. Each state has transitions into and out of it with labels C and A, standing for condition and action, respectively. The condition on one of the transitions out of the Wait state is "Status:0 - 0 & CPOS unchanged".

This means that the transition will be taken if both of the following are true: First, the status of Machine 0 (the other one) must be nonzero. This means that a spike has occurred. Second, no command has been issued to change the position of the EMA. If both of these are true, the action associated with this condition is executed. In this transition, the action is 1) to set the status register of Machine 0 back to 0 so that it can continue looking for spikes in parallel with the actions of any other state machines and 2) to increment local variable 1 (Local:1), which represents the number of spikes noticed by Machine 1.

Current SPIKE Machine (Machine 0)


C: Current Decrease & ΔT > 4
C: Current Increase & ΔT ≤ 4
C: ΔT>4
Possible SPIKE 1
C: Current Increase
Wait
Possible SPIKE 2
C: Current Increase & Δ T ≤ 4
SPIKE
C: Status: 0 = 0
C: Current Decrease & ΔT ≤ 4
A: Status:0 ← Status:0 v 1



EMA Stiction Machine (Machine 1)
C: Local:1 > 4
A: Status:1 ◄ Status:1 v 1
Wait
Stiction
C: Status:0 ° 0 & CPOS unchanged
A: Status:0 ◄ 0; Local:1 ◄ local:1 + 1
C: Status:1 = 0
A: Local:1 ◄ 0

Figure 3. Two state machines

Notice two things: first, a state machine can transition states based on the state of another machine. Second, each machine can have any number of local variables, one of which, the "status", is distinguished by being readable and writeable by any of the state machines.

Another transition out of the Wait state occurs if local variable 1 is greater than 4 (i.e., more than four spikes have been noticed). If this transition is taken, the status register of Machine 1 (the current machine) is set to 1. (Actually, only the lowest bit is set to one, since we would like to save the option of using other bits for some other purpose.) Machine 1 then enters the Stiction state. At this point some other agent — another state machine or some other software — can recognize that stiction has occurred and take appropriate action (e.g., notify the operator). That agent has the responsibility to then reset Machine 1's status register to 0 allowing the machine itself to set the count back to 0 and start over.

The Spike recognition machine (Machine 0) is somewhat more complex: it has four states and seven transitions. Although it is not necessary to cover this machine in detail, there are a few points to make. The most important is that this spike recognition machine is relatively noise free.

There are time constraints ($\Delta$T in the figure) with which changes in the current must be consistent before a spike can be recognized. Intermediate states (Possible Spike 1 and Possible Spike 2), from which the machine can return to the Wait state, exist so that the machine can be sure about whether the fluctuation in the current should be labeled a spike or not. Notice also that some of the transitions can be taken without any explicit actions.

The Spike state has one transition out of it. This occurs when Machine 0's status register (the current machine) is reset to 0. Recall that Machine 1 did this after it had recorded the existence of the spike.

The sizes of the current spike machine (Machine 0) and the stiction machine (Machine 1) are respectively 229 and 93 bytes. The interpreter that executes the SBFR system in the DCs is about 2000 bytes long.

# 7 Communication Protocols

## 7.1 Failure Prediction Reporting Protocol

One of the goals of the MPROS system is to encourage the incorporation of many diverse expert systems supplying diagnostic and prognostic conclusions based upon similar, overlapping or entirely disjoint sensor readings. At the same time, we recognized that these diverse results must be unified into a meaningful report to the system's users. To this end, a standard protocol has been defined for reporting failure predictions to the PDME for fusion and display.

## 7.2 Diagnostic Data

1 KnowledgeSourceID: The unique MPROS object ID for the instance of the knowledge source.

2 SensedObjectID: The unique MPROS object ID for the sensed object to which this report applies.

3 MachineConditionID: The unique MPROS object ID for the diagnosed machine condition.

4 Severity: Numeric value in range 0.0 to 1.0 indicating relative severity of machine condition to operation. Maximal severity is 1.0.

5 Belief: Numeric value in range 0.0 to 1.0 indicating belief that this diagnosis is true. Maximal belief is 1.0.

6 Explanation: An optional text string (possibly very long) providing human-readable description of the diagnosis. Where information beyond defined the Machine Condition description is unnecessary or unavailable, this is allowed to be blank.

7 Recommendations: An optional text string (possibly very long) providing human-readable description of the recommended actions to take. Where information beyond defined Machine Condition description is unnecessary or unavailable, this is allowed to be blank.

8 Timestamp: The timestamp for when this report should be considered "effective."

9 Additional Information: An optional text string (possibly very long) providing human-readable additional information.

### 7.3 Prognostics Vector

Zero to *n* ordered pairs of the form "(probability, time)." Each pair indicates the probability that the given machine condition will lead to failure of the machine within 'time' seconds from now.

# 8 Hardware

### 8.1 Prognostics/Diagnostics/Monitoring Engine

Unlike the Data Concentrator, the PDME is entirely software. It runs on any sufficiently powerful NT machine. Currently, the PDME runs in the office of one of our developers communicating over HTC’s data network with the Data Concentrator in the basement.

## Data Concentrator (DC)

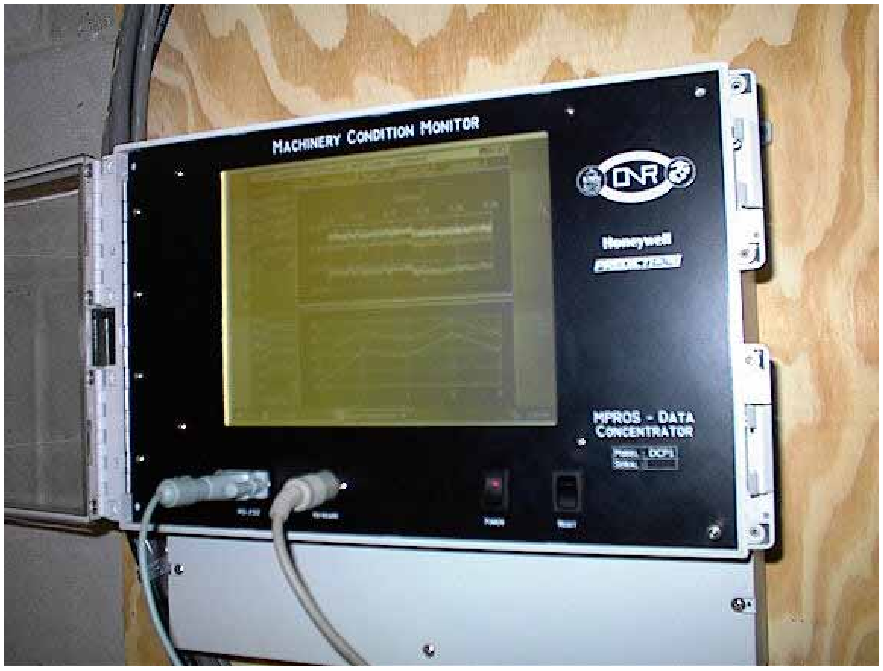


Figure 4. Data Concentrator Installed at HTC

The DC hardware (figure 4 shows the HTC-installed DC) consists of a PC104 single board Pentium PC (about 6”x6”) with flat screen LCD display monitor, a PCMCIA host board, a 4 channel PCMCIA DSP card, two multiplexer (MUX) cards, and a terminal bus for sensor cable connections. The operating system is Win95 and there are connections for keyboard and mouse. Data is stored via DRAM.

The DC is enclosed in a NEMA enclosure with a transparent front door and fans for cooling. Overall dimensions are 10”x12”x4”. The system was built entirely with commercial off the shelf components with the exception of the MUX cards which are a DLI hardware subcomponent and the PCMCIA card which was modified from a commercial 2 channel unit to meet the needs of the project.

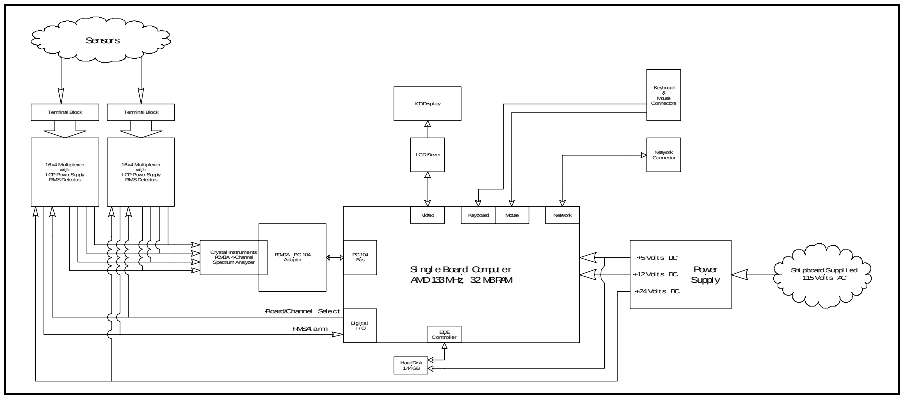

Sensors
Terminal Block
Terminal Block
16x4 Multiplexer with ICP Power Supply RMS Detectors
16x4 Multiplexer with ICP Power Supply RMS Detectors
Crystal Instruments PCMCIA 4-Channel Spectrum Analyzer
PCMCIA - PC-104 Adapter
PC-104 Bus
LCD Display
LCD Driver
Video
Keyboard
Mouse
Network
Keyboard & Mouse Connectors
Network Connector
Single Board Computer AMD 133 MHz, 32 MB RAM
Board/Channel Select
RMS Alarm
Digital I/O
EIDE Controller
Hard Disk 1.44 GB
+5 Volts DC
+12 Volts DC
+24 Volts DC
Power Supply
Shipboard Supplied 115 Volts AC

Figure 5. Data Concentrator Hardware

The 4 channel PCMCIA card samples DC and AC dynamic signals. Highest sampling rate exceeds 40,000 Hz and the length of sampled signals is limited only by the PC's storage capacity.

The MUX cards provide power to standard accelerometers and are controlled using the PC's IO port. Each of the 2 MUX cards can switch between 4 sets of 4 channels each yielding up to 32 channels of data. Of those 32 channels, 24 can power standard accelerometers. All channels can be configured to sample DC voltage signals. Additionally, all channels are equipped with an RMS detector which can be configure to provide a digital signal when the RMS of the incoming signal exceeds a programmed value. This allows for real-time and constant alarming for all sensors.

## 9 Validation

One question we are often asked is "How are you going to prove that your system does what you say it does?" This question, as it turns out, is a quite difficult one. The problem is that we are developing a system that we claim will predict failures in devices, and that in real life, these devices fail relatively rarely.

In fact, for any one failure mode, it is entirely possible that the failure mode (although valid) may never have occurred on any piece of equipment on any ship in the fleet! We have a number of answers to the question:

We are still going to look for the failure modes. We have a number of installed data collectors both on land and on ships. In addition, DLI is collecting time domain data for a number of parameters whenever their vibration-based expert system predicts a failure on shipboard chillers. This will give us data that we can use to test our system.

As Honeywell upgrades its air conditioning systems to be compliant with new non-polluting refrigerant regulations, older chillers become obsolete. We have managed to acquire one of these chillers that HTC replaced. It has been shipped to York, and we are now constructing a test plan to collect data from this chiller through carefully orchestrated destructive testing.

Seeded faults are worth doing. Our partners in the Mechanical Engineering Department of Georgia Tech are seeding faults in bearings and collecting the data. These tests have the drawback that they might not exhibit the same precursors as real-world failures, especially in the case of accelerated tests.

Honeywell, York, DLI, NRL, and WM Engineering have archives of maintenance data that we will take full advantage of in constructing our prognostic and diagnostic models.

Similarly, these partners have human expertise that we are able to tap in building our models.

Although persuasive, these answers are far from conclusive. The authors would welcome any input on how to validate a failure prediction system.

## 10 Current and Future Related Activities

We have just completed phase 1 of our program. The next phase calls for an installation on a Navy ship. Current plans are to install it on the Mercy, a hospital ship stationed in San Diego. This ship was chosen for a number of reasons:

1 It will be in a climate that is likely to require cooling during the winter shipboard test phase of our centrifugal chiller prognostics and diagnostics system.

2 It is likely to stay stationary and not put out to sea (as, for instance, a carrier would).

3 It contains pieces of equipment that are the target of our prognostics and diagnostics efforts.

4 We have a good working relationship with the crew.

Other activities outside our current contract are:

The destructive centrifugal chiller test. Honeywell has donated a surplus centrifugal chiller for use by the prognostics/diagnostics community. We are in the process of assembling a test plan to take full advantage of this opportunity, both to validate our existing system and to discover new indicators of incipient failure that can be incorporated into future versions of our and others' systems.

1 RSVP (Reduced Shipboard manning through Virtual Presence).

2 Honeywell is teamed with Pennsylvania State University for this CBM follow-on.

3 ACI Integration. Honeywell has volunteered to lead this effort to combine the Advanced Capability Initiatives.

These include CBM, Corrosion, Oil Debris Monitoring, and Human Computer Interaction. Other potential CBM related activities. There are a number of these, both inside and outside our members' companies.

## 10.1 Future Directions For Knowledge Fusion

Several high-level control extensions are under consideration for future extensions. First, multi-level data is represented the object-oriented ship model. We are not currently exploiting this fully. For example, we could reason about the health of a system based on the health of a constituent part.

Currently, only the parts are tracked. Second, spatial reasoning using the object-oriented ship model could lead us to fuse information about spatially related components. Examples of spatial relations are proximity (for example, a device is vibrating because a component next to it is broken and vibrating wildly) and flow.

Flows are relationships that represent either fluid flow through the system (one component passing fouled fluids on to other components downstream), electrical flow or mechanical flow of physical energy. Third, temporal reasoning components could be implemented to scrutinize failure histories and provide better projections of future faults as they develop.

Two other aspects of knowledge fusion future directions will be the refinement of specific knowledge fusion components for diagnostics and prognostics. For example, Bayes' Nets seem to be a promising approach to diagnostic knowledge fusion when causal relations and a priori relationships can be teased out of historical data for this system.

Prognostic knowledge fusion could be improved with the addition of techniques from the analysis of hazard and survival data. These approaches scrutinize history data to refine the estimates of life-cycle performance for failures. These refined inputs to the prognostic analysis would yield better projections of future failures.