A Reasoning Architecture for Expert Troubleshooting of Complex Processes

Abdul Naveed[1], Jiaming Li[2], Bhaskar Saha[3], Abhinav Saxena[4], and George Vachtsevanos[5]

[1, 2, 5] *Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332,USA*
naveed@gatech.edu
jli339@gatech.edu
gjv@ece.gatech.edu

[3]*Palo Alto Research Center, Palo Alto, CA, 94304, USA*
bhaskar.saha@parc.com

[4]*SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
abhinav.saxena@nasa.gov

ABSTRACT

This paper introduces a novel reasoning methodology, in combination with appropriate models and measurements (data) to perform accurately and expeditiously expert troubleshooting for complex military and industrial processes. This automated troubleshooting tool is designed to support the maintainer/ repairman by identifying and locating faulty system components. The enabling technologies build upon a Model Based Reasoning paradigm and a Dynamic Case Based Reasoning method acting as the intelligent database. A case study employs a helicopter Intermediate Gearbox as the application domain to illustrate the efficacy of the approach.

1. INTRODUCTION

Complex military and industrial systems (machines, aircraft, etc.) experience fault/failure modes that must be diagnosed accurately and rapidly in order to sustain the operational availability of these systems as high as possible. Considerable downtime translates into loss of productivity and increased maintenance costs. In a manufacturing or industrial setting, problems reported by the machine's internal checks (fault indicator) or an operator alert the maintainer of possible problem areas that must be addressed. Unfortunately, in these situations, the alert or advisory does not offer sufficient information to the maintainer that may permit the accurate and rapid diagnosis of the problem.

An "expert" observes the faulty system, determines the root cause of the problem and composes a work order to schedule people, tools/equipment, or materials for repair and maintenance. If the problem is not diagnosed correctly and corrected, the troubleshooting and repair task is passed on to other technical personnel until a successful solution to the problem is reached. Early efforts to assist the maintainer with troubleshooting and early diagnosis have ranged from built-in test (BIT), and built-in test equipment (BITE) to interactive electronic technical manuals (IETM's). More recently, several organizations have employed rule-based expert systems to provide the maintainer with more systematic tools and methods for fault/failure diagnosis. Although significant improvements have been reported in the troubleshooting arena, the challenge of minimizing the diagnosis/repair cycle and returning critical equipment to service as soon as possible is still remaining and new methods/tools are sought to address it.

This paper addresses the development, testing and evaluation of an intelligent and systematic methodology to troubleshooting that aims to improve current practice and expedite the diagnosis/repair cycle. An expert decision support system is designed to assist the maintainer in navigating through complex system interconnections while reducing the variability of coupled components/subsystems into a well understood series of steps. The enabling technologies borrow from the domains of data mining, modeling and such reasoning paradigms as Model Based Reasoning and Case Based Reasoning. The objective is to integrate trouble-shooting practice with technologies from Condition Based Maintenance as many military and industrial installations are adopting these new and emerging technologies. The vision is the eventual development of a rigorous and automated diagnosis and troubleshooting architecture that will optimize maintenance, repair, and overhaul of complex assets.

2. THE TROUBLESHOOTING METHODOLOGY

Diagnostic reasoning algorithms use troubleshooting information modeled in a specific way to select the appropriate troubleshooting procedure, recommend tests and corrective actions as new evidence is acquired, and determine the root cause of the system failure. As shown in Figure 1, we suggest a novel combination of Model-Based Reasoning (MBR), a reasoning tool about fault modes, and Dynamic Case Based Reasoning (DCBR), a 'smart' knowledge-base. We describe each of these two tools and their contribution to expert troubleshooting in the following sections.

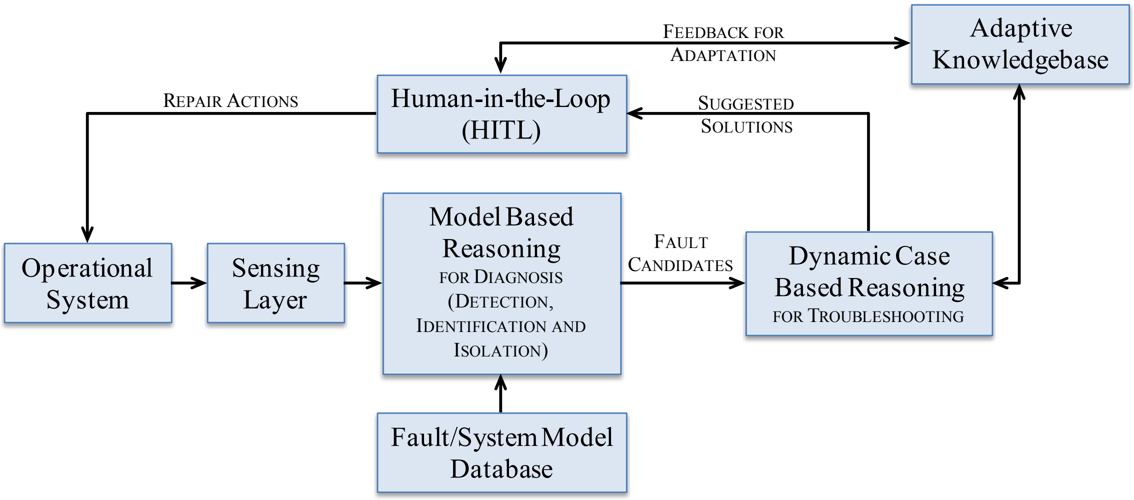
Adaptive Knowledgebase
FEEDBACK FOR ADAPTATION
Human-in-the-Loop (HITL)
REPAIR ACTIONS
SUGGESTED SOLUTIONS
Operational System
Sensing Layer
Model Based Reasoning FOR DIAGNOSIS (DETECTION, IDENTIFICATION AND ISOLATION)
FAULT CANDIDATES
Dynamic Case Based Reasoning FOR TROUBLESHOOTING
Fault/System Model Database

Figure 1. Schematic showing troubleshooting methodology based on reasoners.

3. DIAGNOSTIC REASONING

Critical components that are prone to failure are usually constituent elements of a larger subsystem or system. For example, a gearbox, found on most land, sea and aerial vehicles, is an assembly of various components like shafts, bearings, gears, etc. Such subsystems are instrumented with accelerometers mounted on their frame, which respond to vibration signals that might originate from anyone of the gearbox components. It is crucial, therefore, to devise methodologies that are capable of determining which component is defective and, additionally, how specific components, faults/failures may propagate from faulty to healthy components (the domino effect!) Model-based reasoning (MBR) belongs to this methodological category. It utilizes all the system sensors to enhance and confirm fault isolation. It is an intuitive multi-level modeling approach with inherent cross checking for false alarm mitigation and multi-level correlation for accurate failure isolation (Davis, 1984; De Kleer & Williams, 1987).

3.1 Model-based Reasoning-The Knowledge Database

The starting point of this diagnostic reasoning methodology is the creation of the *a priori* knowledge database. In the decomposition of a system into its components, often satisfactory results can be achieved by looking only as deep as the macro level where we have commercially available replaceable components whose behaviors have been studied in detail. Once the component units have been identified, their behavior is studied and stored in the database in the form of functional descriptions aided by a FMECA study. The functional description of the system is described in terms of these behaviors. Associated with these functions the components also exhibit specific fault modes. These anomalies, represented in the form of features extracted from signals gathered from the system, are called *fault features* or *condition indicators* (CI's). The features are also stored in the database along with the functional description for each component. The basic unit of reasoning here is the fault mode j of component i, represented logically as:

$$F_{ij} = \bigwedge_{k} f_{ijk} \qquad (1)$$

where, f denotes feature threshold being exceeded, k is the symptom index and $\wedge$ denotes logical AND.

3.1.1 System Model Abstraction

Given a system, we start out by analyzing its structural links. The different component parts are identified and their specific structural organization is stored in the form of a *structural model*. Information about the location of each component and its relative proximity to other neighboring components is crucial in predicting how a local failure in one component may propagate though the entire system. In logical terms, this information is represented as:

$$\textit{Set of components}, V = \{v_1, v_2, \ldots, v_n\}$$

$$\textit{Interconnection}, E = \{<v_a, v_b>, \ldots, <v_i, v_j>, \ldots\}$$

$$\textit{Structural Model}, M_S \equiv V \cup E. \quad (2)$$

where, ∪ denotes the union of two sets.

The *functional model* of the system is constructed by traversing the partially connected graph represented by the structural model and substituting the corresponding function for each component from the database. Any anomaly in the system response is then reasoned about and expressed in terms of faulty operational mode(s) of one or more components. This idea is formalized as:

$$Functional\ Model, M_F \equiv \bigwedge_i \bigwedge_j \neg F_{ij} \quad (3)$$

i.e., the nominal system behavior is described as the absence of all known fault modes.

3.1.2 The Fault-Symptom Matrix

The construction of the *fault-symptom matrix* is the main reasoning step regarding overall system behavior. Each unit in the functional model is associated with a number of fault modes, with each fault mode corresponding to one or more condition indicators. A system fault is defined as:

$$F \equiv \neg M_F = \neg \bigwedge_i \bigwedge_j \neg F_{ij} = \bigvee_i \bigvee_j F_{ij} \quad (4)$$

i.e., at least one fault mode has been excited. Here ¬ denotes logical NOT and ∨ denotes logical OR.

A matrix tabulating the various fault modes and their symptoms is generated by traversing all units of the functional model and extracting their features from the database. If sufficient data from seeded fault testing is available from a FMECA study, then the fault-symptom matrix can be enhanced with criticality metrics like severity and frequency of occurrence.

As an example, Table 1 gives the fault-symptom matrix of a generic case where we consider 3 symptoms spread over 3 faults. The fault modes can be written as:

$$F1 = S1 \wedge S2 = (S1 \wedge S2 \wedge S3) \vee (S1 \wedge S2 \wedge \neg S3)$$

$$F2 = S1 = (S1 \wedge S2 \wedge S3) \vee (S1 \wedge S2 \wedge \neg S3) \vee$$
$$(S1 \wedge \neg S2 \wedge \neg S3) \vee (S1 \wedge \neg S2 \wedge S3)$$

$$F3 = S3 = (S1 \wedge S2 \wedge S3) \vee (S1 \wedge \neg S2 \wedge S3) \vee$$
$$(\neg S1 \wedge \neg S2 \wedge S3) \vee (\neg S1 \wedge S2 \wedge S3)$$

(5)

	Symptom 1 (S1)	Symptom 2 (S2)	Symptom 3 (S3)
Fault Mode 1 (F1)	X	X	
Fault Mode 2 (F2)	X		
Fault Mode 3 (F3)			X

Table 1: A generic fault-symptom matrix (X denotes a valid fault-symptom relation).

Using baseline data for the system under study, we calibrate this matrix for acceptable levels of the fault mode symptoms. The choice of thresholds on these symptoms is arbitrary. In the real world, maintenance personnel pick these thresholds from operational experience. In the absence of such expert knowledge we assume that the data is *normally distributed* (equation 5), and we construct this distribution based on the mean μ and standard deviation σ of the given baseline data. We use the upper 3-sigma limit as our threshold.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)} \quad (6)$$

A multi-branched *diagnostic tree* is then constructed from the measurements of the overall system behavior. The nodes of the tree represent monitored system variables, whereas the leaves denote the components that are fault candidates. For the example above, the diagnostic decision tree is shown in Figure 2, where HIGH branches correspond to the X's in Table 1.

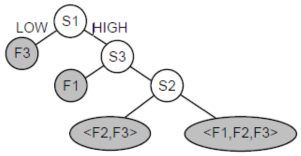

Figure 2. Diagnostic decision tree for the fault-symptom matrix in Table 1.

The leaf nodes below <Fi, …,Fn> in Figure 2 denote that any possible combination of the included faults may occur. Thus, <F1,F2,F3> includes the fault modes F1, F2, F3, F1∧F2, F1∧F3, F2∧F3, and F1∧F2∧F3.

The overall MBR-derived diagnostic procedure can thus be represented by the schematic shown in Figure 3. More details of this architecture have been published by Saha & Vachtsevanos (2006).

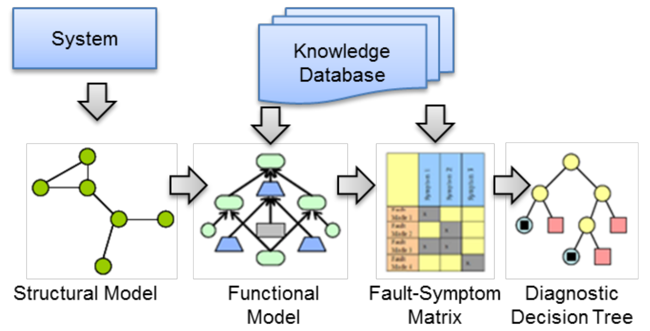

Figure 3. Schematic of MBR diagnostic reasoning architecture.

3.2 MBR Enhancements – Confidence Metrics

In this section we briefly outline an MBR enhancement that incorporates confidence metrics in the reasoning procedure. The architecture is elaborated by substituting the crisp Boolean logic implicitly inherent in the reasoning process, with *probabilistic* values that express the *confidence* of a given symptom. The confidence metric expresses the deviation of an observed symptom measurement from an expected baseline value.

The reasoning method is suggested above is a crisp Boolean logic. The occurrence of a behavioral anomaly appears as the deviation of the feature (or CI) value from a fixed threshold. When this deviation becomes distinct, the feature *f* is set from *0* to *1*. Unfortunately, such a crisp distinction does not reveal the intermediate levels of confidence that are associated with the observation of feature values. In many cases the certainty of a behavioral anomaly based on the observation of a feature value is best described by a value within the *(0,1)* interval.

In order to insert the concept of confidence in the MBR architecture we propose that a Particle Filter (PF) module be used to generate the confidence that a feature exhibits anomalous behavior. Denote by ϕ_i the feature measurement that corresponds to the symptom s_i. The output of the PF algorithm is a *sequence of pdf's*, denoted by $P_{\phi_i}(k)$. At each instant k, the pdf $\mathrm{P}_{\phi_i}(\mathrm{k})$ represents the probability distribution of the feature ϕ_i. This is a function of the time instant k. Historical data are used to determine the baseline pdf denoted by P_B, which of course remains constant with time.

The confidence that the symptom s_i exhibits anomalous behavior based on the measurements of the feature ϕ_i is defined by the overlap area of the two pdf's, $P_{\phi_i}(k)$ and P_B. This overlap area $P_{\phi_i}(k) \cap P_B$ of the two pdf's is the confidence that the symptom s_i indicates *healthy* behavior based on feature ϕ_i, given by $C_{healhy}(\phi_i; k) = P_{\phi_i}(k) \cap P_B$.

Therefore, the confidence that the symptom exhibits *anomalous* behavior is $C_{anomalous}(\phi_i, k) = 1 - C_{healthy}(\phi_i, k) = 1 - P_{\phi_i}(k) \cap P_B$

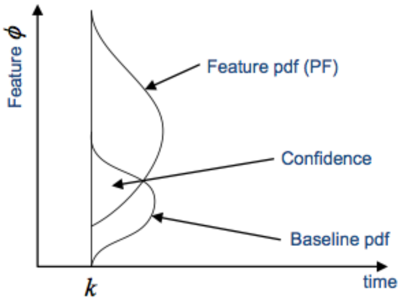

Figure 4. The confidence estimation using PF.

Figure 4 illustrates how the PF module is used to determine the confidence that a feature exhibits anomalous behavior.

For example, Figure 5 shows the time-series progression of Feature 1 denoting the energy in the gear natural frequency in dB, in a gearbox diagnostics scenario (explained in detail in section 4). The baseline pdf indicates the 3-sigma fault threshold at -30dB. From the graph it is clear from the PF tracking output (shown in red circles) that the anomalous behavior confidence $C_{anomalous}(\phi_i; k)$ increases with k for this feature.

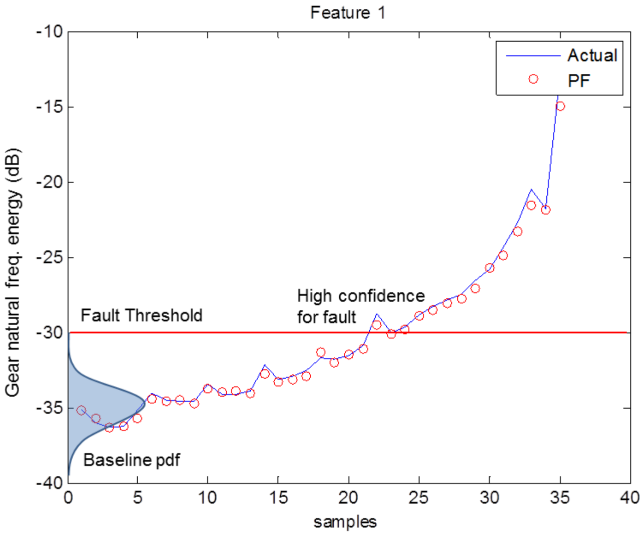

Figure 5: Feature 1 actual and estimated signal from the PF algorithm.

3.3 Dynamic Case-based Reasoning (DCBR)

Having determined the fault with sufficient confidence, DCBR constitutes the main system level reasoning for troubleshooting options and incorporates essential elements of a learning strategy. Case-Based Reasoning (CBR) was founded on the belief that human memory is episodic in nature, which comprises human knowledge accumulated from past experience. Faced with a new problem, a human often relates the problem to one or more memory episodes and composes a solution from these episodes. CBR is a computer program to simulate this human recognition process and has been applied to a variety of process operation support systems. The CBR application domain usually involves problem solving, i.e. identify similar cases for better understanding, assessing and/or comparing with the current situation. The CBR architecture entails five basic steps:

- ***indexing*** – given a new situation, generate appropriate semantic indices that will allow its classification and categorization;
- ***retrieval*** – given a new, indexed problem, retrieve the best past cases from memory;
- ***adaptation*** – modify the old solution to conform to the new situation, resulting in a proposed solution;
- ***testing*** – determine whether the proposed solution is successful; and
- ***learning*** – if the solution fails, explain the failure and learn how to avoid repeating it; if the solution succeeds, incorporate it into the case memory as a successful solution.

The proposed software framework imposes requirements that classical (static) CBR methods are not capable of addressing due to the dynamic nature of the systems of interest and the temporal dependence of problem solutions typically found in troubleshooting. To circumvent such issues a new reasoning paradigm called Dynamic Case Based Reasoning (DCBR) was introduced in (Saxena, 2007; Saxena, Wu, & Vachtsevanos, 2005). DCBR enhances the advantages of conventional CBR systems by interpreting from sensor data not only static features but also dynamic and composite ones. Instead of one indexing path, the DCBR applies two—the abnormal symptom (AS) path, i.e. problem situation detected, and the problem description (PD) one. Furthermore, it entails functions to support case similarity evaluation and situational prediction through temporal reasoning and time tagged indexes. The remembrance calculation module updates the remembrance factors of existing cases. Figure 6 shows the major modules of the DCBR architecture.

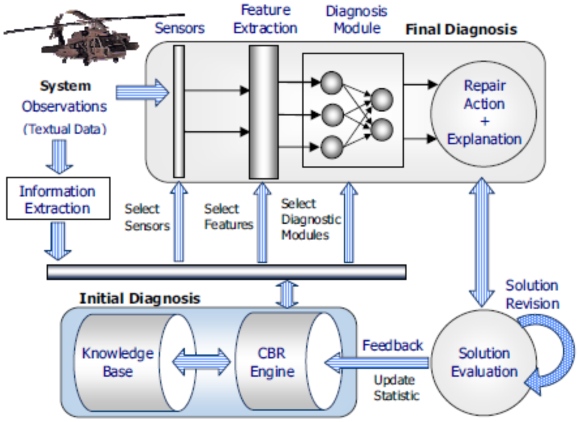

Figure 6. DCBR system for integrated diagnosis of industrial systems (Saxena et al., 2005).

Here, DCBR module depends on MBR to provide probable fault candidates. Once a set of fault candidates are available, DCBR carries out a targeted search within the knowledgebase to identify if any additional measurements are required to confirm the and isolate the fault modes. The next step is selection of the cases from the case base on which reasoning will be based. In this step the notion of similarity between the cases in memory and the new episodic evidence is applied and the performance relies on the quality of similarity calculation algorithms. Depending on the information type contained in the case base the similarity function can take various forms. Several similarity scoring functions have been suggested in literature. As an example we present one such function here. Let Ent_e be a new case presented to the system. A “similarity by proximity” notion may be calculated by the following scoring function (Bichindaritz, 1995):

$$sim(Ent_e, Ent_j) = \frac{\sum_{k=1}^{n} \alpha \times sim(El_{i,k}, El_{l,k}) + \sum_{k=1}^{n} n_{k_i,pred} \times n_{i,pert} \times sim(El_{i,k}, El_{l,k})}{\alpha \times n + \sum_{k=1}^{n} n_{k_i,pred} \times n_{i,pert}} \tag{7}$$

Where Ent_j are cases previously presented, El_i is a feature or an (attribute, value) pair, $n_{i,pert}$ is a pertinence weighted variable associated with the description element El_i, $n_{i,pred}$ is a prediction weighted variable associated with each case in memory. The similarity measure can be defined as:

$$sim(X,Y) = \sum_i \frac{1 - |x_i - y_i|}{1 + \alpha |x_i - y_i|} \tag{8}$$

3.3.1 Learning

Learning from the performance feedback is the key differentiator here with other knowledge based systems. Specific learning methods are applied according to the troubleshooting data environment and reasoner type. In our formulation there are two key learning mechanisms.

3.3.2 Human in the Loop Learning (HITLL)

As originally proposed in (Saxena, 2007) we adopt a Human-in-the-Loop Learning (HITLL) framework where not only the experts are kept in the loop for continuous performance monitoring, but also the system learning activity is accelerated. The human operator who actually performs the maintenance and repair operations (MRO) has the best knowledge about the effectiveness and the adjustments that were needed. He closes the loop between sensing and decision support and provides an assessment of results as well as a confirmation of the success or failure of control actions. The operator's expertise will assist to define and verify the completeness and correctness of the selected features for a particular problem set. It is then the task of system developers to devise appropriate mechanisms in the software to invite such feedback and incorporate the information by adjusting the weights in the knowledgebase.

3.3.3 Incremental Learning in DCBR

Incremental learning occurs whenever a new case is processed and its results are identified. Thus, the memory keeps track of each of its experiences, whether success or failure, in a declarative way; it is then ready to take advantage of future experiences. It must be pointed out that there is no guarantee that the matching process will lead to the most similar case (Kolodner, 1993). Incremental learning is pursued using Q-Learning, a popular reinforcement learning scheme for agents learning to behave in a game-like environment. Q-Learning is highly adaptive for on-line learning since it can easily incorporate new data as part of its stored database. An attractive feature in a game-like situation is that the player is learning to choose the best action for each particular game setting. In this framework, the expected reward or "cost-to-go" is stated as:

Immediate reward

Expected reward: "cost-to-go" funciton — Learning rate — Discount factor

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \qquad (9)$$

Current action — Next action

Current state — Next state

4. A Case Study: The IGB

We now discuss how we apply the proposed troubleshooting approach to a helicopter intermediate gearbox (IGB). A schematic view of the H-60 IGB is shown in Figure 7 followed by the functional diagram in Figure 8.

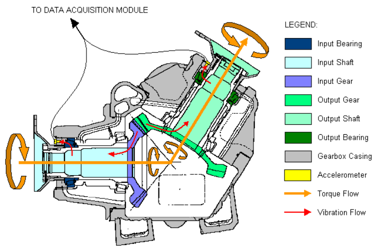

Figure 7. The IGB structure

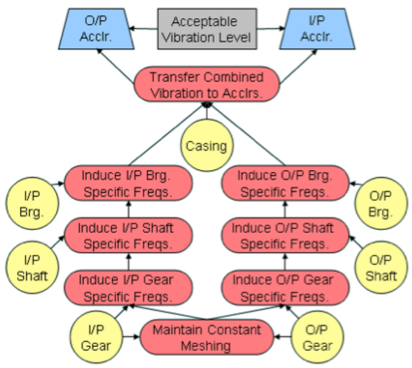
O/P Acclr.
Acceptable Vibration Level
I/P Acclr.
Transfer Combined Vibration to Acclrs.
Casing
I/P Brg.
Induce I/P Brg. Specific Freqs.
Induce O/P Brg. Specific Freqs.
O/P Brg.
I/P Shaft
Induce I/P Shaft Specific Freqs.
Induce O/P Shaft Specific Freqs.
O/P Shaft
Induce I/P Gear Specific Freqs.
Induce O/P Gear Specific Freqs.
I/P Gear
Maintain Constant Meshing
O/P Gear

Figure 8. Functional diagram of the intermediate gearbox.

The three basic types of fundamental mechanical units of the IGB are gears, shafts, and bearings. A collection of features that are characteristic of anomalies in the IGBT functions provides a reliable platform for diagnostic reasoning.

The fault-symptom matrix (Table 2) shows that other than the gear natural frequency the remaining symptoms are different for the input and the output sides. To exploit this feature two diagnostic decision trees are constructed for each of the input and output accelerometer data. Taking advantage of the fact that most faults have only a single symptom we devise a simpler linear decision tree. This tree is traversed in a *depth-first* manner, visiting all the branches, starting from the root. This tree traversal is performed every iteration. Each iteration corresponds to a new dataset recorded from the sensors. When any one of these nodes returns a fault, then the same traversal process is applied to the right sub-tree of that node. It is to be noted that the search space is not narrowed down after initial diagnosis since a fault initiation in one component does not rule out the possibility of other components going bad.

Symptoms / Faults	Sub-synchronous	1 x shaft speed	2x shaft speed	(n/2)x shaft speed	Higher harmonics	Gear Natural freq.	g.m.f. sideband spacing
Input Gear Crack						820-825	65-70
Output Gear Crack						820-825	82-87
Input Bearing Race Defect				200-350			
Output Bearing Race Defect				250-435			
Excessive Clearance I/P Brg	20-50						
Excessive Clearance O/P Brg	25-60						
Input Shaft Imbalance		50-100					
Output Shaft Imbalance		60-125					
Input Shaft Misalignment			100-150				
Output Shaft Misalignment			125-185				
Input Shaft Looseness					350-450		
Output Shaft Looseness					435-560		

Table 2. Fault-Symptom Matrix for the IGB (the symptoms are represented by associated frequency bands in Hz).

The IGB has multiple fault modes depending upon which components have failed or are failing. For example, a bearing may develop a defect in the inner/outer race, a shaft or a gear pinion might develop a crack, or even a gear tooth may get worn out or chipped. All of these fault modes give rise to characteristic vibration signatures. By looking at the corresponding frequency bands in the accelerometer readings we can classify the fault modes present. Some common IGB fault classification heuristics are given below (Vance, 1988):

- Rotor Imbalance – 1 x shaft speed
- Shaft Misalignment – 2 x shaft speed, high axial vibration
- Mechanical Looseness – higher harmonics of shaft speed
- Excessive Bearing Clearance – sub-synchronous whirl instability
- Bearing Race Defect – (*n*/2) x shaft speed, *n* is the number of balls
- Gear Bevel Defect – gear natural freq., sidebands spaced at the running speed of the bad gear.

The fault-symptom matrix can be readily constructed and is shown in Table 2. Figure 9 depicts the computer flowchart.

The choice of thresholds for these symptoms is arbitrary. In this application, for all the symptoms, except for the side band spacing, the energy thresholds are set at the upper 3 sigma limit as determined from the baseline data. For the sideband spacing feature the symptom can take the value of either the input shaft or the output shaft speed. The system parameters required, as input to the MBR program for full IGB diagnostics, are given as 68.6 Hz input shaft speed, 25:31 gear ratio, and gear natural frequency around 822 Hz for both input and output gears.

Since the fault symptoms described are primarily energy metrics, the energy content of each data sample received is used to determine whether the system is in nominal mode or not. Once a fault is diagnosed at the system level we evaluate the diagnostic decision tree to isolate and identify the fault.

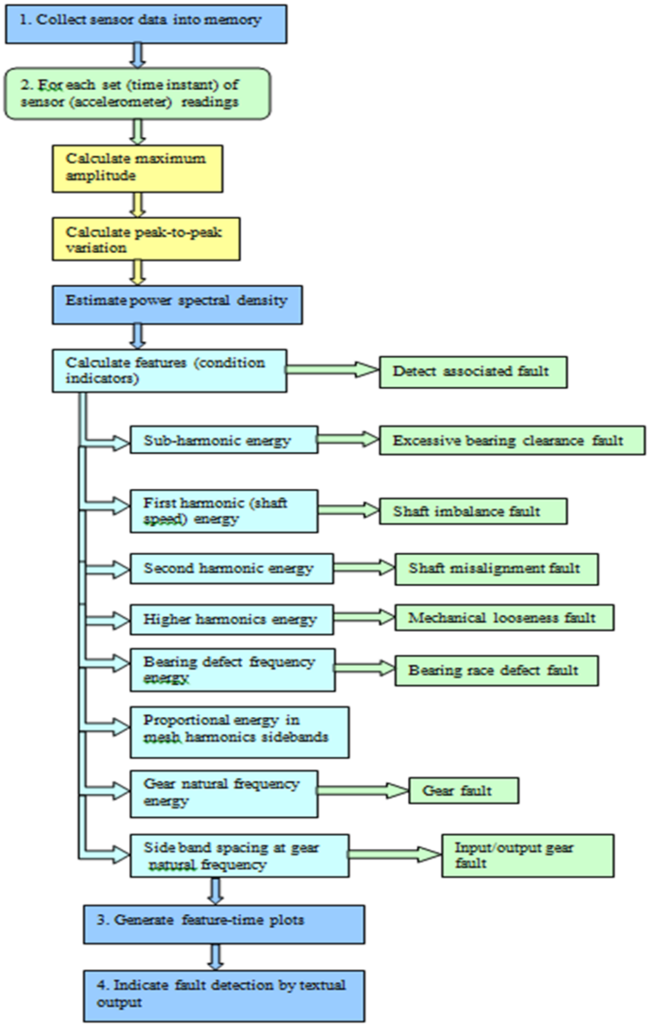
1. Collect sensor data into memory
2. For each set (time instant) of sensor (accelerometer) readings
Calculate maximum amplitude
Calculate peak-to-peak variation
Estimate power spectral density
Calculate features (condition indicators)
Detect associated fault
Sub-harmonic energy
Excessive bearing clearance fault
First harmonic (shaft speed) energy
Shaft imbalance fault
Second harmonic energy
Shaft misalignment fault
Higher harmonics energy
Mechanical looseness fault
Bearing defect frequency energy
Bearing race defect fault
Proportional energy in mesh harmonics sidebands
Gear natural frequency energy
Gear fault
Side band spacing at gear natural frequency
Input/output gear fault
3. Generate feature-time plots
4. Indicate fault detection by textual output

Figure 9. MBR Computer Flowchart for the Helicopter IGB Application.

5. RESULTS

For fatigue crack analysis an IGB pinion gear made of 9310 steel was used in a cyclic crack growth test. It was seeded with faults to initiate cracks. These faults consisted of notches made by an electric discharge machine (EDM), and were located at, and parallel to, the root of one of the gear teeth as shown in Figure 10. The crack growth test consisted of rotation in a spin pit, at a constant high speed with a varying load cycle, to simulate flight conditions. Data collection was done using 2 stud mounted accelerometers at the input and the output of the IGB.

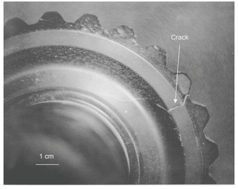

Figure 10. IGB crack.

The data consists of 36 sets of accelerometer readings from both the input and the output ends of the gearbox. A number of features were extracted by frequency domain demodulation (the Fourier Transform of the vibration signals). The observed behavior from the data was compared with the predicted behavior from the functional model of the system. Any discrepancy in the two patterns was used to detect a fault condition, while the heuristics presented above were used to classify the fault.

A healthy gearbox exhibits certain characteristic frequencies. The primary component of the spectrum is the gear mesh frequency, which is 1714.5 Hz in our case. The harmonics of the mesh frequency are also present but their energy content varies with load conditions. When a crack initiates in the gear, the power spectrum starts to show a clear peak at the natural frequency of the gear. There are also characteristic sidebands spaced at the running speed of the bad gear. Thus, the major energy component of the signal shifts from the mesh frequency to the gear natural frequency and its sidebands.

For the frequency domain analysis of the given data we primarily look at the power spectral density. First, we eliminate the mesh frequency and its prominent harmonics. Then we measure the normalized energy at the gear natural frequency (822 Hz). This is a prime indicator of a growing fault condition. The spacing between the sidebands is then analyzed to see if it matches either the input or the output shaft speed. This provides a means of classifying which pinion gear is faulty. The photograph of the input pinion crack, shown in Figure 10, provides validation for our approach.

6. Conclusions

Current troubleshooting tools rely on fault tree analysis, extensive electronic manuals or expert system methods to assist the maintainer in identifying faulty system components and take corrective action. There is a need to expand the technology base with tools and methods that will assure the accurate and expedient identification of fault modes while providing means to learn from similar cases. This paper introduced a framework for an intelligent approach to the troubleshooting problem. More work is needed to improve the mathematical rigor of the algorithm and ascertain through case studies that appropriate performance and effectiveness metrics are met.

References

Bichindaritz, I. (1995). Incremental concept learning and case-based reasoning: For a co-operative approach. Paper presented at the Progress in Case-Based Reasoning.

Davis, R. (1984). "Diagnostic Reasoning Based on Structure and Behavior", *Artificial Intelligence*, Vol. 24, 1984, pp 347-410.

De Kleer, J. & Williams, B.C. (1987). "Diagnosing Multiple Faults", *Artificial Intelligence*, Vol 32, 1987, pp 97-130.

Kolodner, J. L. (1993). Case-based Reasoning. San Mateo, CA: Morgan Kaufmann Publishers.

Saha, B. & Vachtsevanos, G. (2006). "A Model-Based Reasoning Approach to System Fault Diagnosis, WSEAS Transactions on Systems, Issue 8, Vol. 5, pp. 1997 – 2004, August 2006.

Saxena, A. (2007). Knowledge-Based Architecture for Integrated Condition Based Maintenance of Engineering System. PhD Dissertation, Georgia Institute of Technology, Atlanta.

Saxena, A., Wu, B., & Vachtsevanos, G. (2005). Integrated Diagnosis and Prognosis Architecture for Fleet Vehicles Using Dynamic Case Based Reasoning. Paper presented at the IEEE Autotestcon '05 Conference, Orlando, FL.

Vance, J.M. (1988). *Rotordynamics of Turbomachinery*, John Wiley & Sons, Inc., 1988.