12-2002

A Framework for Prognostics Reasoning

Thomas C. Clutz

A FRAMEWORK FOR PROGNOSTICS

REASONING

DISSERTATION

Thomas C. Clutz, Maj, USAF

AFIT/DS/ENS/03-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this dissertation are those of the author, and do not necessarily reflect the official policy or position of the Department of Defense, or the United States Government

AFIT/DS/ENS/03-01

A FRAMEWORK FOR PROGNOSTICS REASONING

DISSERTATION

Presented to the Faculty of the Graduate School of Engineering and Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy in Operations Research

Thomas C. Clutz

Major, USAF

December 2002

AFIT/DS/ENS/03-01

A FRAMEWORK FOR PROGNOSTICS REASONING

Thomas C. Clutz
Major, USAF

Approved:

Date:

_____ _____
Kenneth W. Bauer, Jr. (Chairman)

_____ _____
William P. Baker (Dean's Representative)

_____ _____
Mark E. Oxley (Member)

_____ _____
Raymond R. Hill (Member)

_____ _____
Jeffrey P. Kharoufeh (Member)

Accepted:

_____ _____
Robert A. Calico, Jr. Date
Dean, Graduate School of Engineering and Management

Acknowledgements

The author gratefully acknowledges the support of the many people who contributed to this work.

To the people at AFRL/HESR: First Paul Faas, then Barb Masqualier, for suggesting this topic and encouraging its development. To Vicki, for her indefatigable efforts in finding the most obscure research papers.

To the people at AFIT/ENS: To Dr Jeff Kharoufeh, for his enthusiasm and professional insights into the technical writing required for a dissertation. To Lt Col Ray Hill, for always being interested and willing to help. To Dr Mark Oxley, for spending many hours discussing and perfecting the details of these new concepts. And most of all to Dr Ken Bauer. He generously shared his many talents to ensure the success of this effort, from truly amazing demonstrations of innovation, to a remarkable proclivity for clearly expressing the most difficult concepts.

Table of Contents

Page

List of Figures

Page

Page

Page

Page

List of Tables

Page

ABSTRACT

The use of system data to make predictions about the future system state, commonly known as prognostics, is a rapidly developing field. Prognostics seeks to build on current diagnostic equipment capabilities for its predictive capability. Many military systems, including the Joint Strike Fighter (JSF), are planning to include on-board prognostics systems to enhance system supportability and affordability. Current research efforts supporting these developments tend to focus on developing a prognostic tool for one specific system component. This dissertation research presents a comprehensive literature review of these developing research efforts. It also develops presents a mathematical model for the optimum allocation of prognostics sensors and their associated classifiers on a given system and all of its components. The model assumptions about system criticality are consistent with current industrial philosophies. This research also develops methodologies for combining sensor classifiers to allow for the selection of the best sensor ensemble.

A FRAMEWORK FOR PROGNOSTICS REASONING

I. Introduction

Historically, military aircraft maintenance has been conducted using manual inspections of various aircraft components. These inspections occur either after a completed flight, or according to a particular maintenance schedule. This work is usually conducted without knowledge of existing aircraft faults. This traditional pattern of maintenance and inspection has become increasingly less efficient as aircraft systems have become more complex [9]. Various sources estimate that up to 50 percent of the components removed from the aircraft for fault repair actually retest as fully functional at the maintenance repair facility [17], [19].

As the above problem continues to absorb more manpower and resources, alternative approaches to aircraft maintenance are being considered. Rather than following the pattern of traditional inspections conducted in a periodic fashion without knowledge of existing faults, various organizations are attempting to improve the efficiency of this process. Typically, this is being done through the addition of sensors to the aircraft components, allowing for a direct measure of system functionality. In addition, these sensor data streams may also be able to provide information about the remaining life of the aircraft component. This sensor data would conceptually be fed into an intelligent system which would attempt to detect existing or impending component faults. Not only

would this increase the efficiency of the current process, it would also allow for on-board fault detection and subsequent flight plan modification. This dissertation addresses some of the different aspects associated with this effort to improve current aircraft maintenance practices.

1.1 Definitions: Diagnostics and Prognostics

The science of diagnostics is best described as the utilization of specialized machinery monitoring hardware and/or software for detecting and isolating faults in a given system, which may be either mechanical, electrical, or both. This system may include both hardware and software components. The Air Force Research Laboratory [21] defines diagnostics as the determination of a failure cause (fault detection and isolation) given all available information. Once a failure occurs, diagnostic information can be used to expedite the troubleshooting/repair process. The analysis may also be used for future diagnostics. Current machinery monitoring technology provides data used in expert analysis to extract usable information to isolate causes of any problem. This situation leads to today's *time-based* or *event-driven* maintenance approach (i.e., perform maintenance every 100 hours or when something breaks). Consequences of this approach may include performing unnecessary maintenance actions and causing other problems in the machine that did not exist prior to the maintenance action.

Prognostics is an emerging technology that seeks to build on current diagnostic equipment capabilities. Some current diagnostic systems can accurately detect and isolate faults in a particular system. The goal of a prognostics system is to use diagnostic

information to accurately predict a system's future health, as well as report the systems' current and predicted health, using automated procedures which do not require human intervention to provide the systems' health report. (For clarity, system health is defined as the instantaneous operational status of the equipment being monitored. It relates to the equipment's immediate readiness for deployment or its need for repair actions [21].) In effect, the prognostics system provides the expert interface, and reports on the systems' health. The Air Force Research Laboratory [21] defines prognostics as an assessment of likely future health (educated prediction) of a piece of equipment, based on current information (current health status, history, etc). Accurate analysis of prognostic information can prevent equipment failure and minimize the frequency of scheduled maintenance actions through performance monitoring, tests, and reasoning.

A prognostics system is often referred to as *condition-based* maintenance, since the prognostics system indicates required maintenance actions, either now or in the future. This condition-based method should replace time-based or event-driven maintenance methods, ideally resulting in less system downtime and only required maintenance actions.

The terms "Prognostics and Health Management" (PHM) system, and "Autonomic Logistics System" (ALS) are also found in the literature. The "PHM system" term usually refers only to the sensors, diagnostic algorithms, and prognostic algorithms required for predictive failure capability on a particular system. An ALS is defined as a system intended to communicate appropriate maintenance, supply, and other appropriate

actions to the proper agencies in a timely fashion, based on the information obtained from a prognostics system. However, the term "PHM system" may also refer to both of the previous two definitions: both the predictive failure capability and the ALS component. In this dissertation, it will be clear from context which meaning of "PHM system" is intended.

1.2 Problem Statement

As previously discussed, a PHM system is intended to predict when aircraft component failure will occur. The data from PHM system sensors are collected and fed through to an intelligent data model which has been trained to recognize and differentiate between healthy, degraded, and failure modes of different aircraft components. According to Scheuren [78], this analysis is currently conducted using regression models, allowing all relevant sensor data to be analyzed before a failure is reported. This section discusses the motivations for pursuing a prognostics program, primarily from an Air Force perspective.

The Air Force's aircraft diagnostic approach uses Built-in Test (BIT) units which are incorporated as part of the aircraft hardware and software to detect aircraft faults. However, these BIT units do not adequately identify all aircraft failures down to the single component level. The aircraft mechanic has access to other technical data in addition to the BIT unit data, such as: logic trees, fault charts, symptom/cause charts, and schematics/wiring diagrams. However, the maintainer is still often left with an inability to correctly diagnose the problem, and many times cannot replicate the problem the BIT unit reported. The reported fault may not even exist, which contributes to the inability of

the mechanic to replicate the problem. As stated on Joint Strike Fighter's (JSF) homepage [76], "Aircraft Maintenance and supportability based on Built in Test (BIT) Diagnostics is an antiquated strategy that has proven countless platforms to be unsuccessful in producing the desired results in aircraft reliability and availability."

There is significant motivation in the Air Force to streamline the aircraft maintenance process, from both a cost and operational readiness perspective. According to Stoll and Vincent [87], there is considerable room for improvement in the current Air Force maintenance system. Problems identified in their report include trial-and-error switching of electrical components to determine where the fault is, if one exists. The "Can not duplicate (CND)" and "Re-Test OK (RTOK)" diagnoses also occur regularly (50% of the time [17],[19]). This is thought to be due to stresses related to the operating conditions aboard the aircraft that intermittently interrupt the functioning of the part, causing it to be removed for maintenance. Usually, these stresses cannot be duplicated on the ground. The communication busses and permanent wiring on an aircraft are not tested at present. These components degrade over time, causing intermittent failures in flight and/or sluggish responses from aircraft systems which may be attributed to otherwise fully functional aircraft components. Lastly, since CND results indicate an inability to duplicate on the ground a fault detected during flight, many maintenance personal believe obtaining aircraft system diagnostic information at the time of the fault would improve their ability to identify the problem. This would allow the exclusion of maintenance on parts that did not function because of an aircraft system problem, rather than the part itself actually malfunctioning. Borden [18] expresses similar thoughts. Borky, *et al* [19]

also express this idea - the Air Force is committed to reducing aircraft life-cycle costs, and to achieving high sortie rates with a reduced force structure. To achieve this, the Air Force requires a built-in diagnostics system that can achieve a high rate of accurate fault detection. This capability is at the heart of a PHM effort. Blemel [16] indicates testing costs are skyrocketing, to the point where they are beginning to exceed half the cost of the aircraft they were built to test. Resources are being stretched to the point where it may no longer be feasible to produce adequate, functional test equipment and software. It will be far easier in the future to take advantage of the built-in processing power and software diagnostics aboard the system. MacDonald [52] sums it up by saying most aircraft are over-inspected at great cost to the Air Force.

A panel of defense experts reached similar conclusions in 1996 [71]. The Institute for Defense Analyses held a conference with 41 participants from the technology development, acquisition, and functional support areas of the Armed Services. The participants concluded that current performance of defense systems is not commensurate with what the current state-of-the-art suggests is attainable. Current performance limitations constitute critical problems resulting in increased life cycle costs (and consequently increased support and maintenance workloads), and decreased systems availability. Perhaps even more importantly, the panel stated that potential integrated diagnostic solutions are not limited by currently available technologies. Hence, the diagnostic problem is not a technological problem, but "…a political, cultural, and organizational problem" [71]. However, given the amount of research being done and the fact there are almost no fielded integrated diagnostics/prognostics systems, it seems

that there are still many technical hurdles remaining before implementation of these systems is possible.

The idea of using sensors to predict equipment failure has been around for some time. Most references indicate published research along this line began to appear in the early 1980's [57].

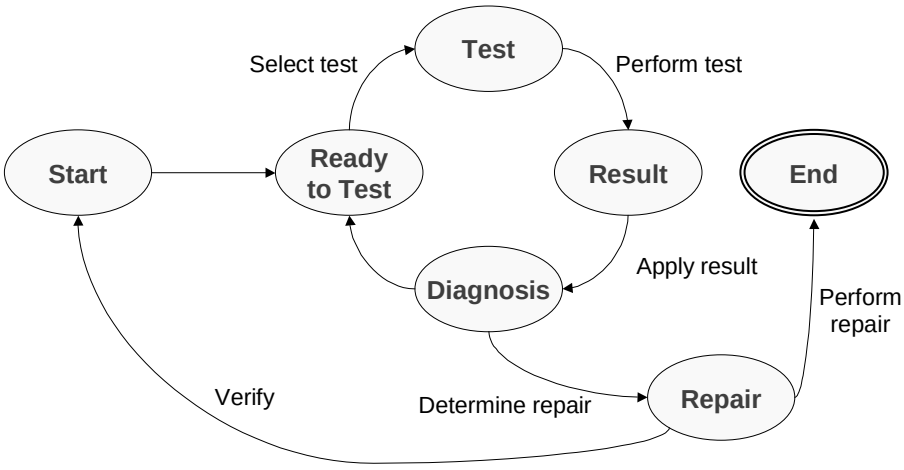

Figure 1-1. A generalized diagnostic process [81]

Figure 1-1 shows a generalized diagnostic process. The initial assumption is that the system undergoing diagnosis has a known fault. At first, the system is prepared for the diagnostic procedure ("Ready to Test"). The appropriate test procedure is chosen ("Test"), conducted ("Result"), and the test outcome is transformed, if required, into a diagnosis. Once the diagnosis is completed, the appropriate repair action is identified ("Determine Repair") and implemented ("Perform Repair"). If there are multiple system

faults, the system is again prepared for diagnosis and the above procedure is repeated until the system is fully functional again. All repairs are also verified: the diagnostic process is repeated to ensure that there are no faults in the system once all repairs have been performed.

In the early 1980's, there were two main groups each favoring a different approach to diagnostics. One group contended that a simple yet comprehensive collection of the observed "abnormal" behaviors of a test unit and the actual failure mechanism provided sufficient understanding of the situation to diagnose the fault. This refers to the testing of a component using a fault tree approach (see Figure 1-2). The test results obtained while following the fault tree's directions help narrow the possible failure mechanisms until the actual mechanism is identified. There is little concern with connecting the failure with the associated symptom since an established diagnostic approach exists. At times, this approach is known as a rule-based diagnostics system, since it was often implemented as an "if-then" set of rules.

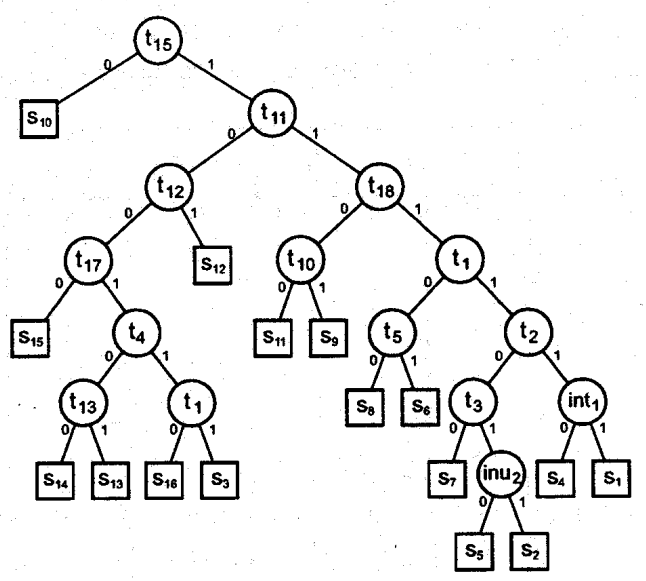

Figure 1-2. Diagnostic fault tree [14]

Figure 1-2 shows a fault tree which may be used to determine the state of a given system or component. Each node in a fault tree indicates the particular type of test, t_i, that should be conducted. The goal is to determine the current system state, shown in this diagram as an s_i index. The 1s and 0s indicate a pass or fail result, respectively, for a particular test. The technician systematically conducts tests to isolate the correct system state. This is similar to the current Air Force maintenance process. However, as discussed previously, CND and RTOK results undermine the fault isolation process.

The second group preferred a model-based prognostics system. This approach assumes an underlying knowledge of the system under consideration. The methodology includes "black box" approaches like neural nets, genetic algorithms, etc. where the user does not require exact knowledge of the workings of the model to obtain useful results. The

knowledge base then contributes to a fundamental understanding of the unit under test, although at times this knowledge may be quite superficial.

The model-based system is being introduced into the Air Force maintenance process. Various authors have published papers summarizing their efforts in this area. One such example described in [22] is the use of neural nets to develop troubleshooting procedures for an on-board avionics system. The particular system chosen for this study was the F-16 Fire Control Radar (FCR) data. Only units known to be faulty were chosen for this study. The purpose of this experiment was to use a neural net to distinguish between three types of faulty FCRs. In this experiment, a success was defined as (correctly) classifying a FCR unit as faulty. FCR units which were classified as faulty were called "normal". The other two ratings were "lemon" or "bad actor". A "lemon" rating meant the faulty FCR system was consistently (incorrectly) identified as a good system in different aircraft. A "bad actor" rating indicated the faulty FCR system was (incorrectly) identified as good only in certain aircraft, and this identification was not necessarily consistent within that aircraft. The neural net obtained around 80% accuracy, which was somewhat less than the authors hoped to achieve.

The panelists at the workshop specified in [60] identified three major classes of models: physical, phenomenological, and empirical models. The panel considered these classes of models important for manufacturing and machine monitoring. Physical models, or mathematical descriptions of a system derived from its physics, represented the first class of models they identified. The panel felt that the most useful physical models do not

capture every detail of the system, but capture the essential features with minimum complexity. Secondly, they summarized phenomenological models as those which identify certain key features of the data, such as spectral lines or modulations, which are used to characterize the system. These models demonstrate a much looser or even only a qualitative coupling between the actual physics of the process and the model features. Finally, they called "empirical," or data-driven models, those models that were based predominantly on features extracted from training data by mathematical or statistical methods without direct reference to the physical system. Some examples from this class of models include Markov models, varieties of neural networks, and simulation models.

The panelists also discussed the conflict between physically based models and empirical models. (Phenomenological models represent the middle ground between the two approaches.) The following direct quote mirrors directly the conflict between the two different modeling camps, as previously mentioned before from [60]:

> Perhaps the greatest differences of opinion among workshop participants centered on the topic of physical versus empirical modeling. Some participants felt that only models well grounded in physics could lead to significant progress. Proponents of empirical modeling argued that, while empirical modeling might not lead to the best possible solution, it can offer substantial improvements, it can be applied immediately in situations for which adequate physical models do not currently exist or are too expensive or complicated to obtain, and substantial success has been demonstrated in real applications. Perhaps grudgingly, almost all workshop participants ultimately agreed that both physical and empirical models have an important role to play, and that significant research is needed in both of these directions. [60, Section 3.4.5]

The participants in [60] did manage to agree that the two physical and empirical modeling approaches require different approaches to model validation. Empirical models require a training/validation set containing sufficient examples spanning the full range of

machines, faults, or situations. Of course, this makes it much more challenging to develop a robust empirical model, in terms of the volume of work required. Physical models usually have a much smaller, more restricted set of parameters, and the validity of the model is determined with a much smaller test set. Furthermore, the intrinsic confidence in a physical model is usually much higher since it is based on known principles of physics rather than "unknown" features which empirical models identify based on the data. Empirical models, in general, require much more rigorous, extensive, and expensive training and validation than physical models; however, there are situations in which the necessary quantity and quality of training and validation data is available or can be collected more easily than developing an adequate physical model.

The workshop participants then commented that methods used to analyze the data from mechanical system processes must be robust, i.e., methods which can tolerate significant deviations from assumed or nominal signal characteristics. In general, the signal and noise environment in these kinds of applications is highly complex, non-Gaussian, and exhibits large variability and/or non-stationarity. The operating conditions may vary dramatically between sensor locations. To ensure the user accepts these monitoring methods, low false alarm rates are an absolute necessity. This places an additional burden on the robustness of the methods.

The workshop participants identified reliable estimation of time-to-failure as one of the greatest challenges in manufacturing and machine monitoring, and one of weakest areas in existing methods. Most faults of interest are believed to begin with small precursor

events and to stem from a progressive (not necessarily linear) degradation of the tool or machine component. Thus, the tracking of this degradation along with ongoing prediction of the time-to-failure is of great importance. As the signal characteristics from many types of degradations are non-monotonic, continuous monitoring which tracks the history of the developing fault is often essential.

1.3. Motivation for a Prognostics System

The manufacturing infrastructure of most of the civilized world embodies the operation and maintenance of machine systems. Both the commercial and government sectors have a vested interest in technical advancements which may enhance the productivity, efficiency, or quality of these machine systems' operations. Such efforts can potentially provide enormous cost savings and enhance industrial competitiveness. A primary example is the repair and maintenance of these systems, which represents an annual cost of many billions of dollars to U.S. consumers, industry, and government [60]. Although monitoring is not cost-effective for inexpensive and non-critical machines such as lawnmowers or fans, accurate system component condition assessment has the potential to save large amounts of money while dramatically increasing safety and reliability of important, complex systems.

Examples where system assessments are appropriate include power generation turbines and critical equipment in nuclear reactors or on large oil rigs, where unscheduled failure can result in lost revenue approaching a million dollars per day. Failure during

operations of aircraft engines or power train components in helicopters can often result in loss of life as well as the equipment [60].

The combination of rapid advances in signal processing techniques with cost-effective digital technologies for their implementation may alleviate the system monitoring challenges that currently exist. These advances include both improvements on existing methods such as spectral analysis and cyclostationary signal analysis, and emerging techniques. Among these new technologies are advances in wavelet and time-frequency signal analysis. These techniques can be used to characterize both transient phenomena and persistent harmonic structure. Consequently, they appear well-matched to the signals associated with rotating machinery. Other recent developments, such as higher-order spectral theory, could also possibly contribute in these applications. Also, higher-level techniques such as neural networks and statistical pattern recognition and classification provide means for combining lower-level processing into detection and categorization of faults. In fact, preliminary research by several groups in applying the techniques mentioned above to a variety of related problems has demonstrated improvements over traditional approaches [60]. These methods, with appropriately directed research, may offer solutions for the critical technology needs in manufacturing and machine monitoring and assessment.

Methods for machine monitoring and assessment which provide warning in time to cease operations or schedule maintenance can provide immense value in these applications, such as aircraft engines, aircraft electrical systems, and automobile assembly lines. In a

number of cases, some prognostic monitoring is routinely used or at least eagerly sought. An excellent example is found in some military applications. Since the cost (and security risk) of unscheduled failure in some military applications is enormous, preventative maintenance is routinely practiced. Future weapons systems, such as the Joint Strike Fighter (JSF), will have these kinds of prognostic condition assessment methods designed and built as an integral part of the system.

Prognostic condition assessment allows performance of maintenance during regularly scheduled service rather than on an emergency basis after failure, thereby greatly reducing the total cost of the maintenance operation. Other sources of unnecessary cost include replacing critical components based on mean time to failure data versus actual component operational status. Additionally, fault indicators can be unreliable, meaning many good components are removed for maintenance or repair as a result of an incorrect fault indication, thus wasting resources on non-existent problems. This action violates the "if it ain't broke, don't fix it" philosophy. However, the practicality of this philosophy is predicated on reliable system condition assessment. To accomplish the converse of the above principle (“fix things only if they’re broken”) requires early detection of precursors to equipment failure. Finally, routine maintenance itself may cause failures. Some sources state that routine maintenance is actually the dominant cause of failure [60].

A recent DoD study noted that “There does not appear to be a consistent approach in either commercial or defense systems for functional and physical partitioning of the hardware and software used to perform integrated diagnostics functions.” [72] This study

defines integrated diagnostics as"…part of the systems engineering (or reengineering) process in which diagnostic functions are partitioned to components, both on and off the product, to optimize economic and functional performance throughout a products life cycle. Optimal performance is achieved by ensuring effective communication of information relevant to the test and diagnostic process occurs between diagnostic functions and components and across each life cycle phase." Success in these efforts is essential for a successful prognostics system.

This study encompassed fourteen civilian and military programs in an attempt to determine what current industrial and military practices were in the field of prognostics. Besides the preceding conclusions, the study determined that a consistent approach to diagnostics is feasible. In general, the study's approach consists of four steps. The first is to develop a consistent, information-based technical architecture for integrated diagnostics. The second is to identify key/critical interfaces and elements of this architecture. The third step is to develop a rough information model for integrated diagnostics. And the fourth step is to prepare a roadmap to advance an open system approach to integrated diagnostics.

The DoD study also identified key requirements for success in the development of prognostic programs. Among these items were: reducing diagnostic ambiguities and inaccuracies, correlating diagnostics with operational performance, the development of measurable and relevant metrics, and the development/maintenance of industry standards facilitated by a domain specific organization. A significant requirement for the last item

is the development of standardized data encapsulation and adherence to a consistent architecture for integrating diagnostic elements.

The Air Force intention is to use prognostic systems to completely eliminate traditional aircraft inspection and repair patterns. Currently, an aircraft goes on a mission and returns. The aircraft mechanic then uses Built In Test (BIT) results from Line Replaceable Units (LRUs) (available only after the aircraft lands) and pilot input (when available) to check the aircraft for malfunctions. The malfunctioning units are identified, removed, and sent to the maintenance depot for further diagnosis and repair. As previously indicated, a BIT result does not always indicate the exact system fault, nor can the mechanic always identify the problem, if one even exists. The first goal of the proposed prognostics system is to fix this diagnostics problem; the new system is intended to be able to find and isolate aircraft faults with complete confidence. Once this is complete, the prognostics system can report the specific aircraft faults to the maintenance and planning/operations activities. (While the goal for a prognostics system is to predict the occurrence of these faults, the first capability required for a prognostics system is the ability to identify an aircraft fault with high confidence.) Reported aircraft faults allow the mechanic to estimate the required workload and preposition/order the necessary maintenance equipment or replacement parts. This capability is usually referred to as health management. Any fault and time-required-to-fix information can be sent to the planning/operations activity to allow them to update the functional capability of that aircraft and overall mission readiness.

Quoting from the JSF homepage [76], “Prognostics and Health Management (PHM) is a technology maturation project focused on using advanced sensors integrated through algorithms and intelligent models such as neural nets to monitor, predict, and manage aircraft health. The goal of PHM is to enable what the JSF program calls Autonomic Logistics: a maintenance and supply system wherein information on aircraft faults detected while the aircraft is airborne is automatically downlinked to trigger the logistics system to meet the returning aircraft with appropriate parts, maintenance personnel, and maintenance equipment. This will allow the Right maintenance action, at the Right time, for the Right reason.”

A National Science Foundation (NSF) Workshop on Signal Processing for Manufacturing and Machine Monitoring workshop brought together 37 academic researchers and industrial leaders and users of prognostics together to identify the pertinent signal processing technologies and the most important industrial needs. Their findings were disseminated to the entire community [60].

Most of the applications discussed in the NSF workshop involved either rotating or reciprocating machinery. It thus appears quite possible that a promising prognostic method could potentially solve a wide variety of machine monitoring problems. However, the workshop participants cautioned that requirements, signals, and data rates can be very different for similar kinds of machinery (rotating and reciprocating machinery), as well as different types of machinery. Consequently, different prognostic methods may be required based on the individual case. [60]

The industry participants in the NSF workshop [60] made it clear that the value of monitoring lies primarily in fault prediction. As might be expected, after-the-fact detection of serious failures is generally of little use, and does not require specialized sensors to determine that something has gone seriously wrong. As an example, consider the failure of an F-16 jet engine. Since the F-16 is a single-engine aircraft, engine failure will almost always lead to pilot ejection and consequent loss of the aircraft. It is clear in the case of engine failure that there was a catastrophic failure—what may be unclear is the cause of this failure. Specialized sensors may have been able to detect an impending failure condition, and that detection may have been able to save the aircraft.

The primary value of monitoring comes in predicting failure in time to prevent it, and in reliably estimating the remaining time before the component fails. (See the taxonomy of a PHM system in the immediately following section for a complete discussion of PHM system capabilities.) The NSF conference participants provided the following example from the automotive industry:

> …in the automotive manufacturing industry, it is a common practice to change all of the tool faces in all of the machines at the end of a shift. The only monitoring question of real interest in this context is whether a tool will fail before the end of the shift and thus cause an extremely expensive unscheduled shut-down; the exact amount of wear on a drill bit is of little interest unless it presages a catastrophic failure. Research efforts should thus be more focused on prognostics and on early detection of fault precursors. [60, Section 3.4.1]

Researchers at the Boeing Company have also devoted considerable thought to the integration of on-board monitoring methods in mechanical systems, specifically military

aircraft. They term their concept Integrated Vehicle Health Management (IVHM) [9]. Their concepts include on-board monitoring elements and ground-based logistic support functions, which function similarly to the DoD's concepts of a PHM and ALS, respectively. The title of their paper includes the term "Tri-Reasoner," and this term refers to the incorporation in their system of three independent views of the vehicle's health. These three views are: the anomaly detection and reasoning system, the prognostic reasoning system, and the diagnostic reasoning system. Outputs from all three systems are combined in a concept termed the "integrated model" and the "reasoner integration manager". This paper provides a valuable overview of the issues which must be addressed for any prognostics system.

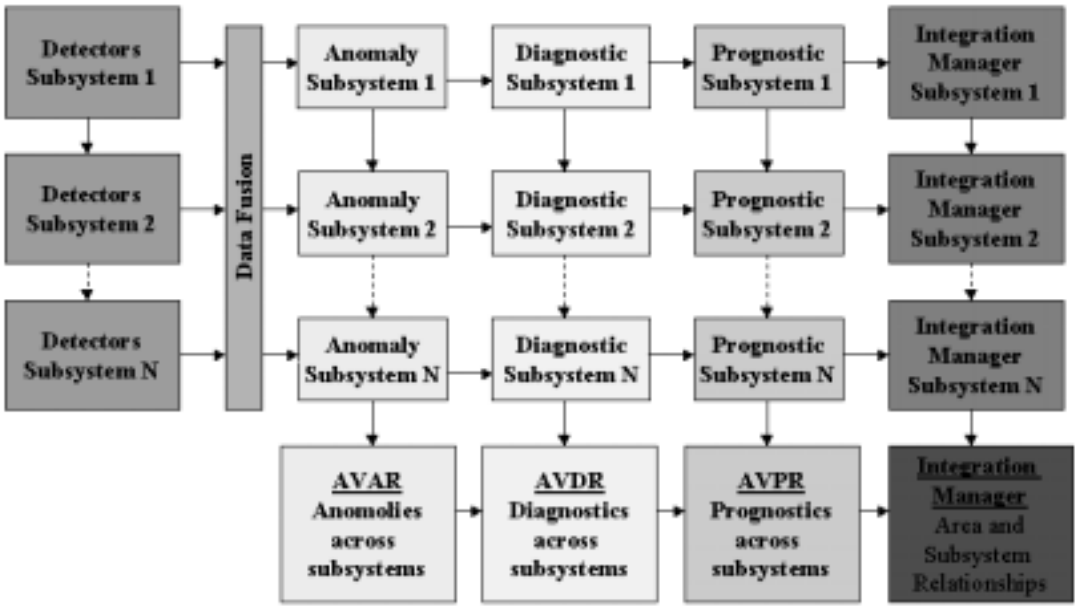

Figure 1-3. The Tri-Reasoner IVHM system [9]

Figure 1-3 shows the data collection scheme and reasoners for different aircraft subsystems. Each subsystem has a dedicated set of detectors and the three independent reasoners. Detector information is combined in a data fusion engine and passed to the three reasoning subsystems. The subsystem integration manager takes the results from the three reasoners and sends the appropriate information to the central integration manager.

Each reasoner has a specific function as well. The anomaly detection algorithms typically use the raw detector data. These detection algorithms condition the data as part of their processing. The associated Anomaly Reasoner (AR) assesses this conditioned information within the integrated model. The AR's task is to evaluate both the raw data and extracted features for correlation and measures of evidence for fault conditions. The main tools the AR uses are generic signal processing and statistical techniques. The correlation and "ripple" effect of anomalies across subsystems is then examined within the Air Vehicle Anomaly Reasoner (AVAR). The AVAR's goal is to correlate anomalies that occur across subsystems and to separate the "upstream" causes from "downstream" effects.

The individual diagnostic algorithms and the associated Diagnostics Reasoner (DR) further examines the root cause of an anomaly detected by the AR. The DR is intended to incorporate a-priori engineering knowledge and models of a component or subsystem (i.e. model-based diagnostics).

The Prognostic Reasoners (PR) and their associated individual prognostic algorithms are focused on predicting the time to system failure, or the failure of a component or components within a subsystem. The intent is for these predictions to be given as distributions about a Mean Time To Failure (MTTF), thus resulting in different acceptable risk limits based on the consequences of the particular failure mode. A PR relies inherently on the individual prognostic algorithm results and an integrated model.

The overarching reasoner, known as the Reasoner Integration Manager's (RIM) function tracks and evaluates the progression of anomalies, diagnoses and prognoses across all subsystems. Through direct algorithm interaction with the Integrated Model and corroborating/conflicting evidence associated with the individual reasoner reports, the RIM prioritizes the most probable fault or failure modes at the air vehicle level. The RIM then isolates the most probable failure modes. The RIM then creates reports for the operators, maintenance personnel and engineering support staff.

Since not all aberrant behavior patterns in a new aircraft system can be predicted before system completion, the IVHM will need to be flexible in its capability to diagnose system problems. Similarly, the techniques and technologies used for observing the aircraft's behavior, and for reasoning about these observations, will continuously improve during an aircraft's operational life. To ensure these capabilities for new diagnoses and new methodologies can be included in the current on-board system, the IVHM architecture incorporates embedded learning components. Additionally, the underlying diagnostic procedures and reasoners will be coded in a modular format to allow for easy exchange

of software modules as new diagnostic procedures are developed and new programmatic tools come into existence. The overall scheme is shown in Figure 1-4 below:

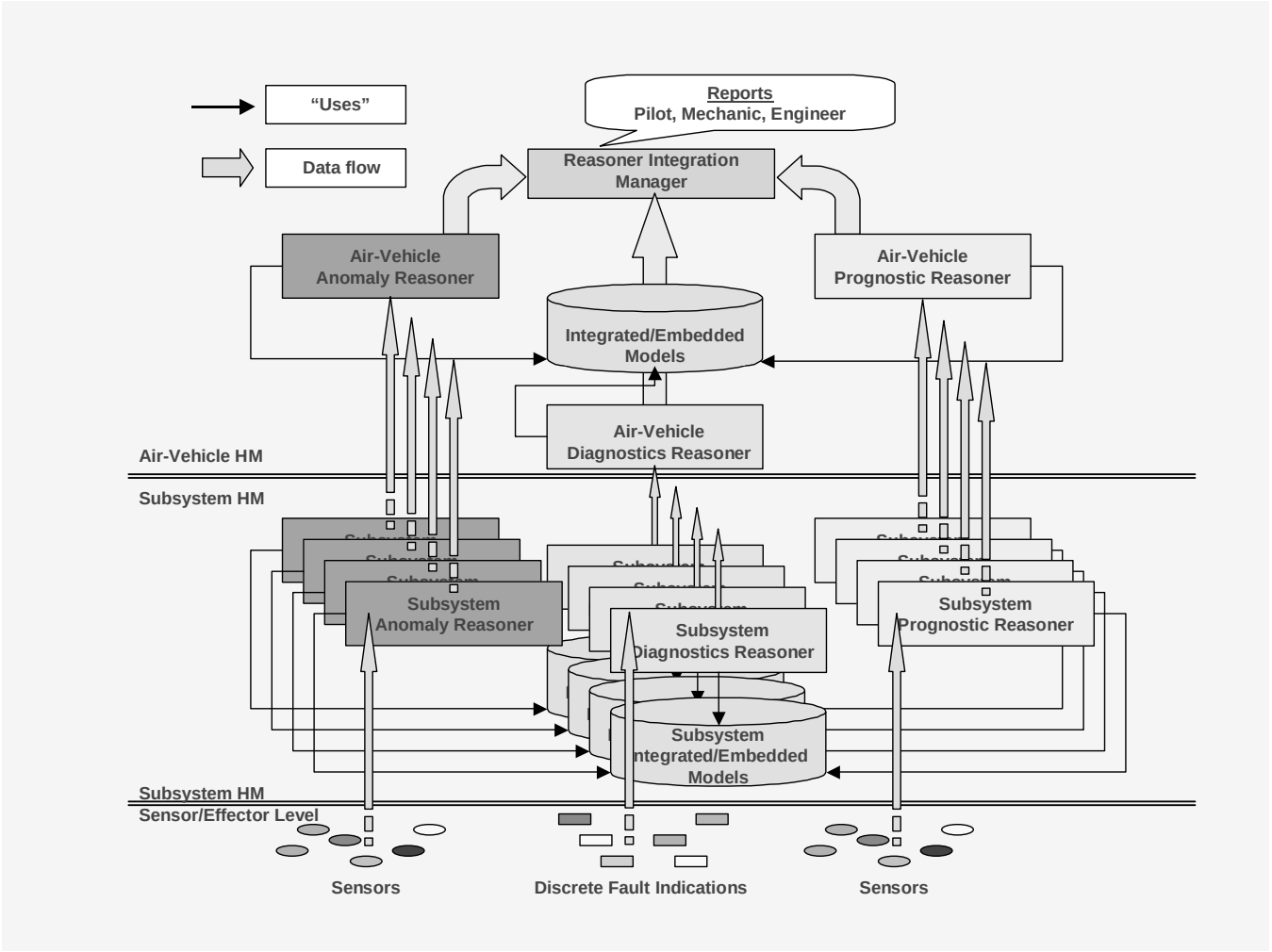

Figure 1-4. The Tri-Reasoner Integrated Vehicle Health Management system [9]

Figure 1-4 shows the overall IVHM tri-reasoner architecture. As previously explained, there are three independent views of the vehicle's health and a reasoner integration manager (RIM) (the box at the top center of Figure 1-4). Each aircraft subsystem has a

dedicated reasoner suite composed of the three models and the associated sensors or fault indicators. This information flows to the RIM for overall diagnostic/prognostic assessment and reporting to appropriate entities. The tri-reasoner algorithms are generic and decoupled from any domain knowledge to enable the use of algorithms that have withstood a wide variety of applications thus increasing the confidence in their reliability.

1.4 Research Goals

There are three main research goals for this dissertation. The first is to summarize the major areas of research currently being performed in the field of prognostics. The second goal is to create a mathematical architecture for the implementation of a prognostics system. This architecture includes a sensor selection algorithm and methodologies for combining sensor information. The third goal is to demonstrate the utility of this algorithm by solving some notional examples.

1.5 Dissertation Organization

This dissertation is organized into five chapters. This chapter has provided a general overview of the prognostics problem. The second chapter provides a literature review of prognostic method papers. Chapter three provides an overview of some mathematical techniques which are commonly used in the analysis of prognostic data. The fourth chapter presents a notional methodology for defining and solving a prognostics problem. Chapter five illustrates this methodology using a sample problem. Chapter six summarizes the contributions of this work and provides recommendations for further research.

II. Literature Review

2.1 Background

The purpose of this chapter is to summarize the history of aircraft diagnostics/prognostics development, provide a PHM system taxonomy, and summarize the major areas of research being performed today. The first section summarizes the historical development of diagnostic/prognostic efforts, and describes a notional PHM system. The second section describes some technologies that may be used in a prognostics application. The third section describes the main diagnostic approaches used for aircraft maintenance. The fourth section describes the main modeling approaches used for diagnostic/prognostic applications. The last section summarizes the information provided in this chapter.

2.1.1 History

The material for the history section is primarily drawn from Atlas, *et al* [9].

Early generation aircraft relied on manual detection and isolation of problems on the ground. These aircraft were composed of systems that were analog and independent of one another. Only a schematic, voltmeter, and reports from the pilot were required to diagnose problems.

As these aircraft systems became more complicated, Built In Test Equipment (BITE) was introduced in the aircraft to warn the pilots of critical failures in important components.

However, the aircraft mechanic did not use BITE. The mechanic still relied on the schematics, voltmeter, and pilot reports.

In time, aircraft design engineers realized that the output of the fault detection monitors could be made available to support mechanic troubleshooting (in the form of analog BITE reports). With these monitors, the concept of "fault balls" was born, and was incorporated on some aircraft systems as early as the 1940s. Fault balls are indications, normally on the front of a Line Replaceable Unit (LRU), that a fault has been detected - they were originally mechanical, but later were replaced with small Light Emitting Diodes (LED's). In many cases, the LRU front panel contained a test switch to command the LRU to test itself in a manner similar to how ground support equipment could test the LRU. This capability also became known as Built In Test Equipment (BITE). This capability began to decrease the need for some of the ground support equipment previously used to test airplane equipment. Depending on the system, the fault balls could effectively point the mechanic in the right direction, but schematics and voltmeters were still needed for most conditions. The BITE results of this era was often confusing, unreliable, and difficult to use. Mechanics often distrusted it. Despite problems, many systems on airplanes such as the Boeing 707, 727, early 737/747, McDonnell Douglas DC-8, DC-9, and DC-10's employed this type of maintenance design.

In the 1970s, some of the increasingly complex aircraft systems began to use computers to perform their fault diagnostic calculations. This was called digital BITE. With these computers came the ability to display fault detection and isolation information in digital

form, normally via numeric codes, on the front panel of the LRU. The digital logic could produce codes that could better isolate the cause of the fault. The digital display offered the capability to display many different codes to identify each type of fault that was detected. These codes often pointed to some description in a manual that could be used to isolate and correct the fault. Many systems on the Boeing 757/767, Airbus A300/310, McDonnell Douglas DC-10, and Lockheed L-1011 still employ this approach.

As the number of systems grew, use of separate front panel displays to maintain the systems became less effective, particularly since each LRU often used a different technique to display its fault data. In addition, some of the systems had become increasingly integrated with each other, due to the introduction of digital data buses, such as the ARINC 429. Autopilot systems were among the first to use digital data buses and depend on sensor data provided by other systems. Consequently, these autopilot systems have been a leading cause of requiring more sophisticated maintenance systems. The more sophisticated monitoring was necessary to meet the integrity and certification requirements of its automatic landing function. For example, the 767 Maintenance Control and Display Panel integrated the maintenance functions of many related systems. In 1986, the ARINC 604 digital data bus defined a Central Fault Display System (CFDS) to incorporate the maintenance indications for potentially all of the systems on the airplane into one display. This approach enabled more consistent access to maintenance data across systems, a more comprehensive display function than each of the systems could provide individually, and saved the cost of implementing front panel displays on many of the associated system LRUs. In this approach, the CFDS is used to select the

aircraft system for which the aircraft mechanic desires maintenance data, and then the CFDS routes the maintenance data from that aircraft system to the display. This approach was employed on some of the systems on later Boeing 737s, and most systems on the Airbus A320/330/340, and McDonnell Douglas MD11.

As systems became more complex and integrated, a single airplane fault could cause fault indications for many systems, even when displayed using the CFDS. The mechanic had little help in determining which fault indication identified the source fault, and which were merely effects of the source fault. To solve this problem and related issues, the ARINC 624 was developed in the early 1990's. This system provides a more integrated maintenance system that can consolidate the fault indications from multiple systems, and provide additional functionality to support maintenance. Minimal ground support equipment is needed to test airplane systems, as most of this capability is included in the ARINC 624. For example, most factory functional tests of airplane systems on the Boeing 747-400 and 777 airplanes consist of little more than execution of selected tests, monitoring fault displays, and monitoring certain bus data using the ARINC 624.

The main goal in fault isolation on the airplane has always been to identify the LRU causing a fault. This allows the aircraft mechanic to confidently remove the failed component and correct the fault condition. Although in many cases this is possible, there are many other cases where diagnosis and repair is not possible without the addition of sensors and/or wiring. The addition of sensors and/or wiring increases the number of components that can fail, and thus sometimes can worsen the maintenance effort, since

the aircraft mechanic must now distinguish between failed aircraft systems and failed aircraft sensors and/or wiring. In addition, these diagnostic sensors and/or wires add cost and weight to the airplane.

As a result, current fault isolation techniques for aircraft cannot produce the perfect answer (the single faulty LRU) in all cases. This is a practical matter, since the wholesale integration of aircraft systems is really the reason why perfect diagnosis in modern aircraft is impossible, given current techniques. However, today, it can point the mechanic to a small group of LRUs in almost all cases. Since the technical limit of diagnostic systems has been reached, aircraft engineers are looking into prognostic systems for assistance with diagnostic issues. The accurate prediction of when faults on an aircraft can be expected to occur is the next big step.

2.1.2 Fault Taxonomy

Any given system has a multitude of unique characteristics due to myriad sources of variability. These sources include manufacturing (both across and within manufacturers), reaction to ambient environmental conditions, system part replacement and repair, etc. In addition, variability appears in the performance of the system's components (e.g. mechanical, electrical, and hydraulic). A system's age also modifies these unique characteristics. In the presence of this variability, on-board aircraft health management systems must be able to accurately distinguish between "normal" operation and the presence of a fault.

This section presents a taxonomy of system behaviors between which a prognostics system must be able to distinguish. These behaviors are defined as: nominal, incipient fault, intermittent fault, active fault, system fault, sensor fault, and novel fault. The term "anomalous event" is used to collectively include the six kinds of faults. An anomalous event indicates a system that either does not have all available functionality or is not operating within its intended design constraints.

The nominal behavior of a system is that behavior that exists when all intended functionality is available and is operating within the constraints of the intended design at a given point in time. The system can be functioning as intended at two different points in time, even though the characteristics of individual system components and sensor operating characteristics may have changed. As an example, the Concorde fuselage expands about 12 inches in length during a flight across the Atlantic. However, the aircraft does not lose functionality as a consequence of this expansion.

An incipient failure exists on a system or component that is still operational, but is trending towards a failure condition. An example would be a hydraulics systems that is losing pressure. The hydraulic system may still be fully functional, but is trending towards a state of non-functionality.

An intermittent fault occurs infrequently, yet repeatedly. The system with an intermittent fault has full functionality when the fault is not present. An example of this kind of fault is a loose electrical connection that causes sporadic short circuits in the affected system.

An active fault is system behavior outside the range of intended functionality. An example is exceeding the revolutions per minute limit of a passenger car engine—this is an operation of the engine above its intended functionality. Active faults do not necessarily indicate a loss in system functionality, though a system may quickly transition from an active fault to another kind of fault.

A system fault is when a system component or subcomponent is no longer functional. Examples include an engine that no longer rotates or a hydraulics system that has lost sufficient fluid/pressure to properly operate system components.

A sensor fault occurs when a sensor within a system component or subcomponent 1) reports a fault condition when none exists, or 2) does not report a fault condition when one does exist. Of these two conditions, the second may be more detectable on an attended system since an operator will likely notice a loss of functionality despite the lack of a fault report. The first condition, also called a false alarm, is likely to be the most troublesome since measures may be taken to correct the non-existent fault which disable other correctly functioning systems. For example, a false alarm of an aircraft engine fire may lead the pilot to eject from the aircraft, resulting in destruction of the entire aircraft, and possible injury or death to the pilot.

A novel fault is an unknown anomalous condition. This type of failure event does not result in nominal system behavior, nor can it classified in any of the known fault conditions. It is something completely new in the system's behavior. This type of fault

may adversely affect the performance of the system, or it may not. It is the only kind of fault which may not be of concern to an operator. An example is the development of a rattle in an aircraft throttle lever. If it does not affect the pilot's control of the engine speed, it would be classified as a novel fault, and is not likely to concern the pilot.

2.1.3 PHM System Taxonomy

The main goal of prognostics, and a PHM system, is to accurately predict future failure of system components in order to replace these components before they actually fail, avoiding shutdown and potential damage to the system. The ultimate benefit is enhanced performance at lower cost, since components are not needlessly replaced before their life cycle ends, and components do not fail while still integrated in the system. Components left to fail while still in the system can shut the system down and potentially lead to damage to other, otherwise healthy, system components.

A PHM system accomplishes accurate detection through real-time on-board diagnostics and the performance of prognostic functions (forecasting the useful remaining life of component parts) with reasonable lead times, eliminating traditional inspection and repair patterns. Rather than fixing a component after it has failed, it can be replaced when prognostics indicate that probable time to failure (or probability of component failure) is within some critical threshold.

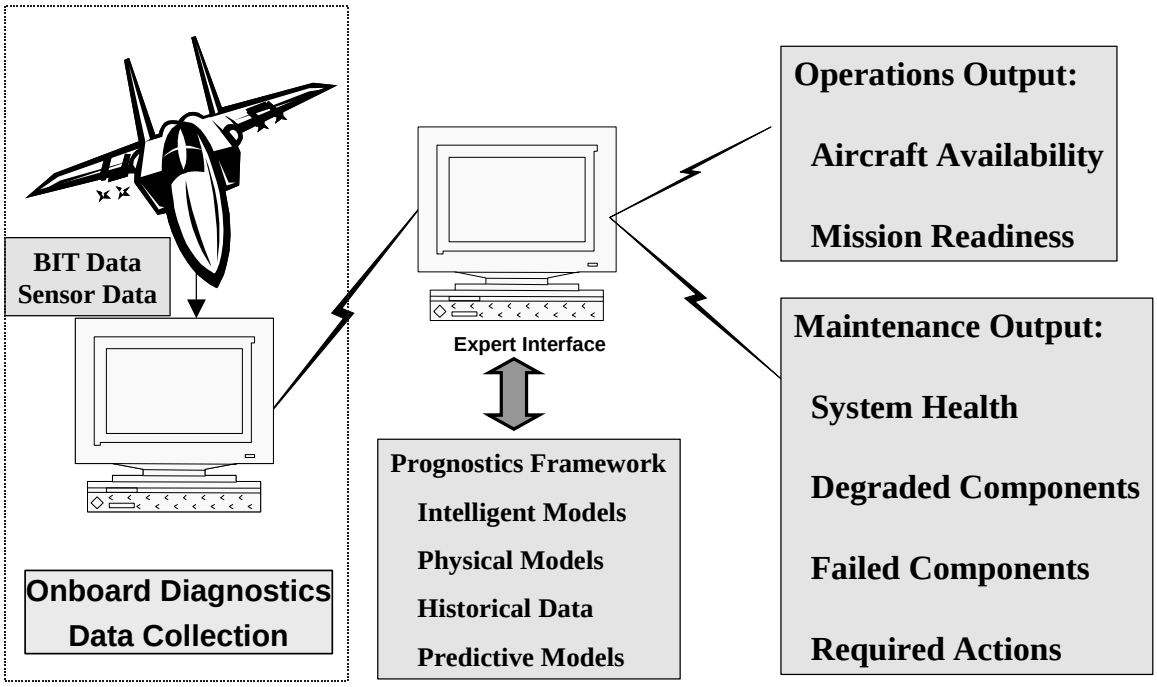

Figure 2-1. A notional prognostics system

Figure 2-1 shows the flow of data through a notional prognostics system. Aircraft sensor data is routed to a diagnostics data collection terminal. This data is sent to an expert interface which employs a prognostics framework to analyze the diagnostics sensor data. The expert interface then provides a report on the health of the aircraft. This report includes a list of components with estimated time to fail, a list of components that have failed, required maintenance parts and actions, and an assessment of aircraft readiness/time before becoming fully operational again.

A prognostics system needs a fully functional diagnostics system. The diagnostics system must accurately report appropriate data from system components up to an

appropriate level (based on the system). This may be done in either of two ways: passive or active monitoring.

A passive monitoring system observes the current behavior of the system components. For example, this can be a sensor (ensemble) that monitors the current coming from a motor, or a sensor (ensemble) monitoring airflow from an engine. The majority of sensors used in aircraft today are passive. As an example, an on-board BIT unit is a passive monitoring system, since it observes and records component performance.

An active monitoring system interacts in some way with a system component of interest (even while the system is in operation). It may send a known signal of some kind into the system component of interest. It may also collect a sample from the system component, such as engine fluid. As an example, an external sensor (ensemble) is attached to the component, and this sensor (ensemble) sends a signal through the component at a level that minimally affects the component's operation. The component's reaction to this signal is captured through either the same or a different sensor (ensemble). A BIT capability to conduct a component self-test is an active monitoring capability. This captured data is then sent to an expert interface for analysis.

A prognostics system also requires an expert interface with appropriately high levels of sensitivity and specificity. In this context, sensitivity means the prognostics system correctly identifies when a fault or degradation is present. Specificity means the prognostics system correctly identifies when a fault or degradation is not present. The

incoming diagnostic data must be correctly classified as indicative of either correct system function, system degradation, or a system fault. This expert interface may be just for a single system component, meaning there very likely are many of these interfaces within one system. The expert interface may also be an overarching system which combines the results of all the diagnostics inputs from all the system components. The design will depend upon the mechanical system.

A prognostics system should also provide system component health predictions based on the incoming diagnostic data. There are many different kinds of predictions that a prognostics system may produce. These predictions include assessments of future component/system events and probabilities associated with both current and future component/system events. The following paragraphs discuss the main predictive outputs of a PHM system.

The expert interface of a prognostics system should provide a level of confidence associated with its assessment of fault/non-fault for a particular system component. Another key prediction capability is the time remaining until component/system failure. The prognostics system may also be able to characterize this measurement using two confidence level measurements and a system-level measurement. The first confidence level measurement is associated with the predicted time remaining until component failure. In turn, this leads to a system-level measurement of the probability that this component actually fails before the "predicted time remaining" elapses. These predictive measurements allow for the replacement of components before they actually fail,

preventing catastrophic consequences in systems where component failure can lead to the failure of many previously healthy components. Related operational measures are degraded system status information and a future time frame health status for critical systems, such as aircraft.

The above prediction capabilities may then be extended to the prediction of a degraded component/system condition. The definition of "degraded" is unique to the component or system under consideration. The expert interface should have a third classification status of degraded, in addition to indicating fault and non-fault status. Again, the prognostics system may then use confidence level measurements similar to those previously described. A "degraded" predictive measurement allows for a more precise (perhaps) replacement of parts that are about to fail – it may allow for increasing the functional lifetime of the part before it is removed to prevent system failure.

Another measure is the probability of failure of a component/system within the next cycle of operation of the mechanical component/system. As an example, the goal could be to determine the probability of failure of an aircraft engine during its next overseas flight, or during its two-week hiatus in a location with very limited access to maintenance parts. Again, a level of confidence in the immediately preceding probability definition is a desirable measurement.

Any predictive information can be obtained from a prognostics system and used for automatic maintenance planning, parts orders, mission planning, etc. This automated

logistics concept is called the Autonomic Logistics System (ALS). The goal of PHM and ALS is to provide a complete overall system health monitoring capability, and consequent maintenance and planning management capabilities. Eventually, sufficiently redundant mechanical systems may be designed that can reconfigure themselves based on predicted failures. However, much basic research remains to be done before a complete overall system health monitoring capability becomes a reality.

2.1.4 Description of an ALS System

A main component of a PHM system is an ALS. An ALS is intended to be a real-time, intelligent global logistics network dedicated to the support of the Joint Strike Fighter (JSF). An ALS is intended to identify and communicate appropriate maintenance, supply, engineering, safety, and training actions to support and enhance mission execution. Figure 2-2 shows a notional ALS concept.

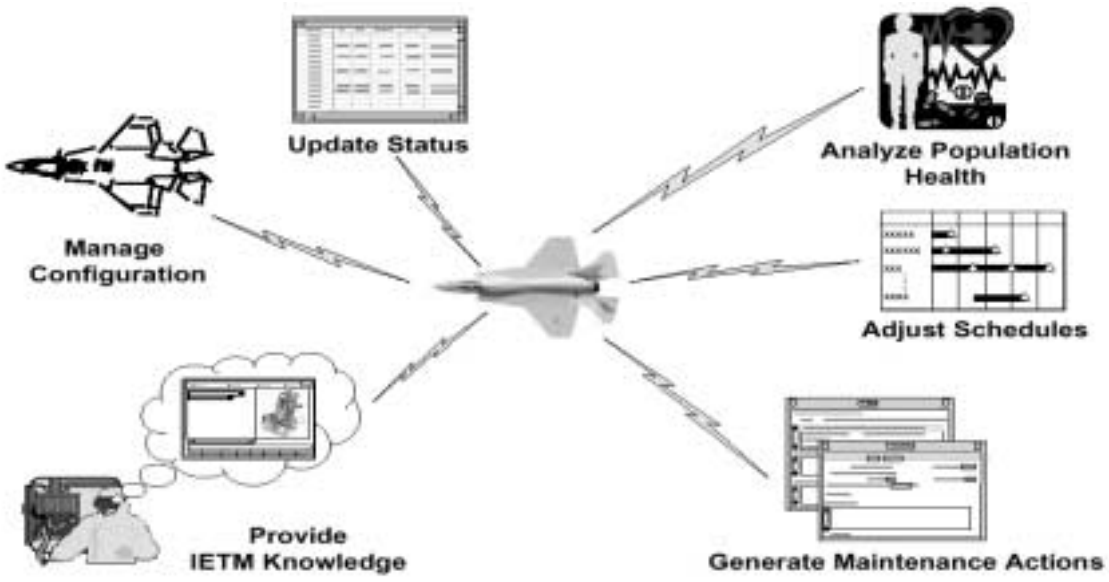

Figure 2-2. Autonomic Logistics System (ALS) model [77]

Figure 2-2 shows how aircraft sensor data, once processed by a PHM system onboard an aircraft, provides a list of degraded and failed components to appropriate maintenance and planning activities. This information provides an up-to-date picture of the aircraft health, required maintenance actions and parts, and updated mission planning schedules. The aircraft mechanic's Interactive Electronic Technical Manual (IETM) is also updated with information about the current state of the aircraft's systems.

The autonomic term in the ALS acronym refers to an intended automatic trigger of appropriate actions within the system (subject to human controller inputs), based on current mission status and requirements. The autonomic support concept is similar to the human autonomic nervous system that directs the human body to perform heartbeat, breathing, and other functions with minimal human intervention. The logistics parallel is a system that is stimulated, prior to an aircraft's return, to ready appropriate tools and spare parts.

The ALS, using PHM input, automatically determines that certain parts are reaching the end of their service life and ought to be replaced, and reports this information before the aircraft lands. This is in contrast to the traditional method of diagnosing aircraft component failures upon the return of the aircraft, and then readying the appropriate tools and spare parts. Also, in present systems, maintainers rely on often ambiguous problem descriptions from the pilots. The autonomic system, in contrast, relies on an integrated report from aircraft diagnostics that minimizes incorrect maintenance actions and consequently reduces maintenance support requirements. The ultimate intent of the ALS

and PHM working in concert is to reduce maintenance manpower, logistics machinery, and increase sortie rates. Most of the discussion that follows focuses on the PHM aspects of the system, rather than the autonomic support concepts, as PHM capabilities are necessary to realize ALS goals.

Su [88] divides how people have thought about prognostics into two different concepts: component/part and system. The concept used to model prognostics has influenced the way the prognostics problem is addressed. According to Su, prognostics have traditionally been regarded as a component/part problem. This led to the adoption of failure statistics and analysis methods to identify and replace failed components. Some examples of the sensors employed include time/stress measurement devices, vibration monitoring, and system sensors (oil, water, etc.). Some examples of the analysis methods investigated include neural networks, genetic algorithms, and trend analysis. These techniques are usually system specific—they are applied piecemeal to the particular problem under consideration and combined in a unique fashion to provide results which are meaningful only for that particular piece of equipment. However, when viewed as a system problem, the prognostic approach necessarily becomes much more involved. Systems such as satellites operate in environments with little or no human interaction. Ideally, there would exist a common set of sensors and techniques which could be applied to all of these types of systems. Su calls this concept an overall Prognostics Framework, a generic, tailorable software tool that uses model-based reasoning to integrate embedded test and sensor data into diagnostic and prognostic systems. The ultimate goal is to produce a generic tool capable of being applied to all different kinds of

warfighting systems. This would lead to the integration of all warfighting systems into a single architecture for the future battlefield.

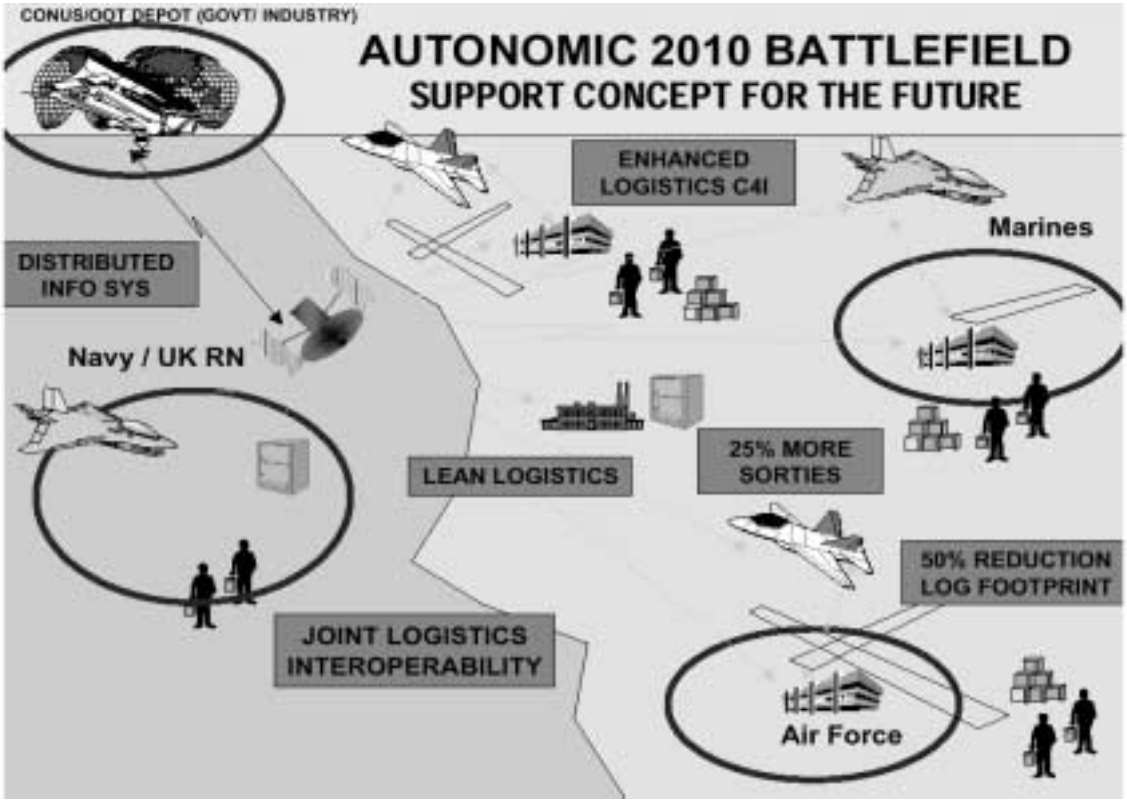

Figure 2-3. Future military systems support concept [77]

Figure 2-3 illustrates the single architecture concept. PHM and ALS are extended from just aircraft systems to all systems used in a warfighting scenario. All the PHM and resulting logistics information from the involved warfighting platforms is collected via a distributed information system, and delivered to an enhanced logistics system. This enhanced logistics system handles all the required logistics actions, allowing for joint logistics interoperability and the notional improvements in logistics performance shown in Figure 2-3.

2.1.5 Technical Feasibility

The main goal of a PHM system is to understand and predict when components (and possibly consequent systems) will fail. To accomplish this, a PHM system will likely use artificial intelligence or other methods to predict failure of system components. Traditional sensor-based diagnostics recognize the functional and failure modes of the aircraft and its components. A PHM system extends this approach, using models to predict the onset of failure modes.

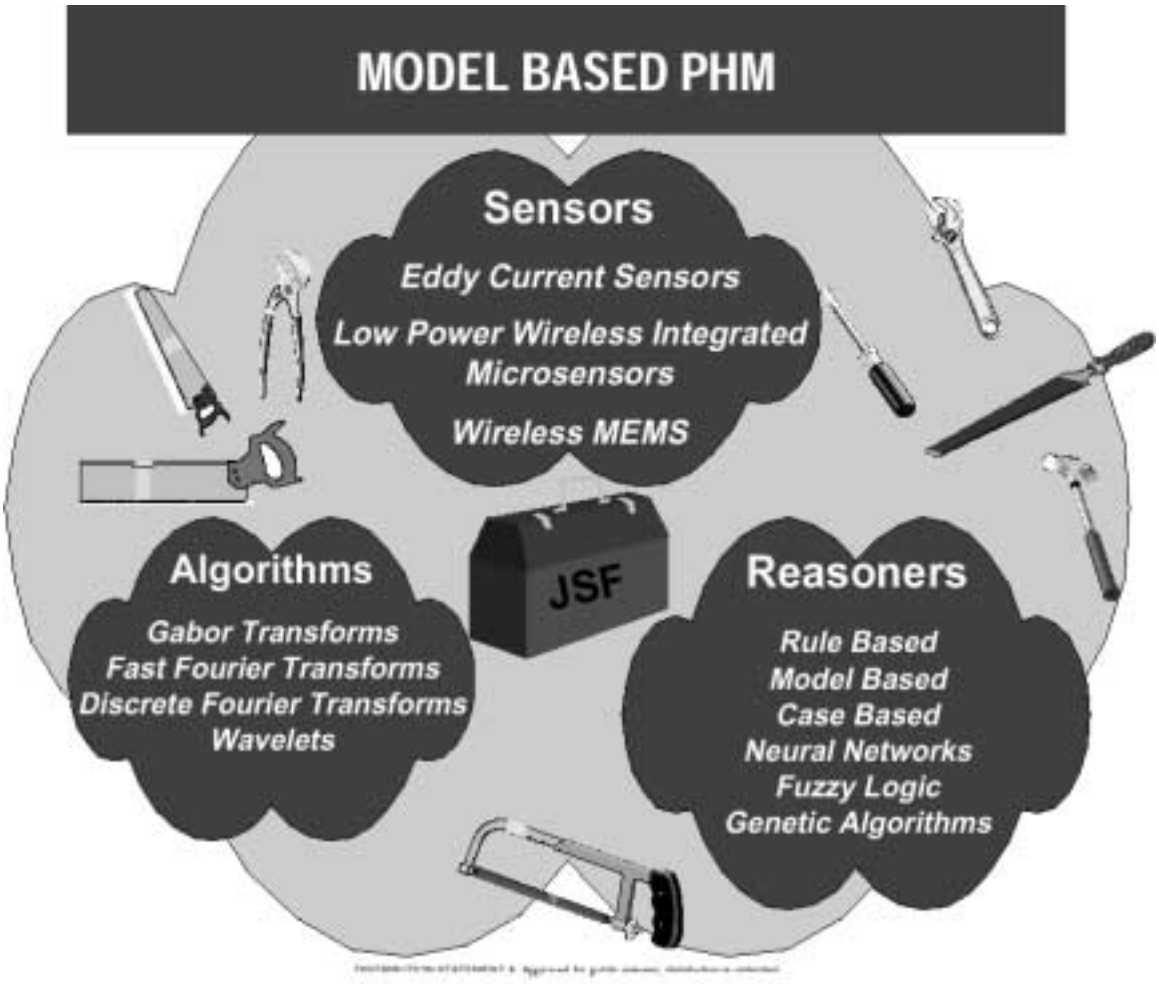

Figure 2-4. Model of PHM system [77]

Figure 2-4 shows the collection of raw data from the sensors, the transformation of this data into a meaningful output via algorithms, and the extraction of key features from the output via some reasoners. The focus is on using the mathematical models of artificial intelligence, such as neural nets and fuzzy logic, to extract key features of the operation of the aircraft system. Individual sensor data is used in these computations, but all features are fused before a PHM system reports a failure. This fusion action is intended to minimize the number of incorrect diagnoses the system produces, reducing unnecessary maintenance actions and costs. Research with intelligent diagnostic systems has shown that accurate measurements of appropriate variables can be used to reliably predict future failure [11], [12].

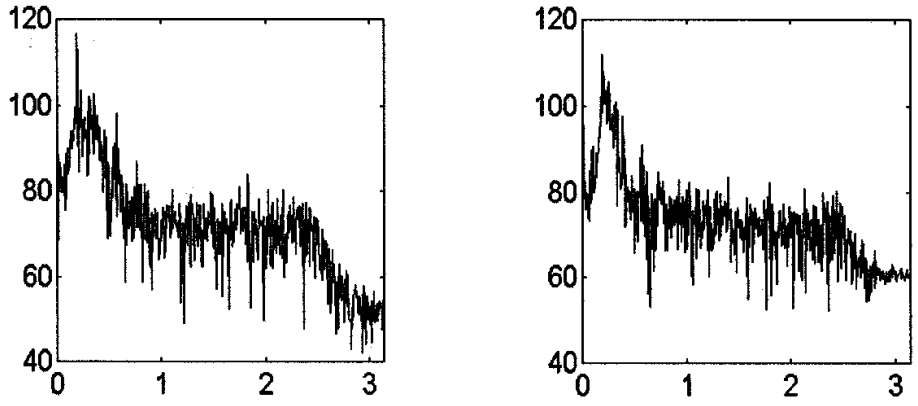

Figure 2-5. Spectral lines from a faulty item (left) and a correctly functioning item (right) (Magnitude in dB vs. frequency; wavelet decomposition can detect the difference) [49]

Figure 2-5 shows an example of raw data taken from accelerometers attached to the aft transmission of a helicopter. According to the authors, exhaustive tests indicate there are

no obvious features in this raw data which can be used to classify it as a fault or no-fault class. So, the authors used a wavelet decomposition on the data in an attempt to extract useful features. They discovered the features useful for classification are non-stationary, confirming the wavelet decomposition as a very suitable choice. (The features were certain frequency bands.) For fault classification, the wavelet coefficients are computed as a function of time. A simple average and standard deviation are computed for each data channel in a given time window, and the results are compared to a set of nominal values for fault classification.

Other research programs seem to indicate that a PHM system is, in fact, an attainable goal. The UK Ministry of Defense used a neural net model to accurately predict structural life used on the basis of recorded flight data [10]. Also, DARPA participated in a research project which showed an engine control sensor suite could be operated with 4 sensors instead of 7 [32].

2.2 Technologies/Applications

According to the NSF workshop [60] participants, the most prominent method (by far) for manufacturing and machine monitoring is spectral, or "FFT" (fast Fourier transform) analysis. Cepstral variants are often employed to increase robustness or to reduce the variability of the FFT estimates.

A cepstrum is the Fourier transform of the log magnitude spectrum:

$$\mathrm{FFt}(\ln(\,|\,\mathrm{FFt}(\mathrm{window}\,.\,\mathrm{signal})\,|\,)$$

and was coined in a 1963 paper by Bogert, *et al* [17]. (A “window signal” is the signal that appears on a given graph—it occurs in the “window” that the graph shows.) They observed that the logarithm of the power spectrum of a signal containing an echo has an additive periodic component due to the echo, and thus the inverse Fourier transform of the logarithm of the power spectrum should exhibit a peak at the echo delay. They called this function the cepstrum, interchanging letters in the word spectrum because "in general, we find ourselves operating on the frequency side in ways customary on the time side and vice versa. (sic)" This term has come to be accepted terminology for this inverse Fourier transform of the logarithm of the power spectrum of a signal [66].

The unusual terminology surrounding the computation of the cepstrum was introduced in the original article by Bogert *et al* [17], in which various terms from signal processing (spectrum, frequency, analysis, phase) were rearranged into anagrams (cepstrum, quefrency, alanysis, saphe). The authors did this to highlight this unusual treatment of frequency domain data. The frequency data was treated as if it were time domain data in the transformation of it to a data set which had units of seconds across its x-axis values (the quefrencies), but which indicated variations in the frequency spectrum.

The cepstrum is commonly used in voice recognition applications and rotating/reciprocating machinery analysis. As an example of the former, the consonants of speech are usually transient and of short-burst character. However, vowel sounds (and tones sung by a singer) are formed by repetitive emission of pulses into the vocal tract [62]. This leads to the use of the cepstrum to analyze these pulses. Similarly, rotating

machinery exhibits a repetitive emission of pulses, and this suggests the same analysis technique. As a result, the concept of the cepstrum has become a fundamental part of the theory of systems for processing signals that have been combined by convolution [62].

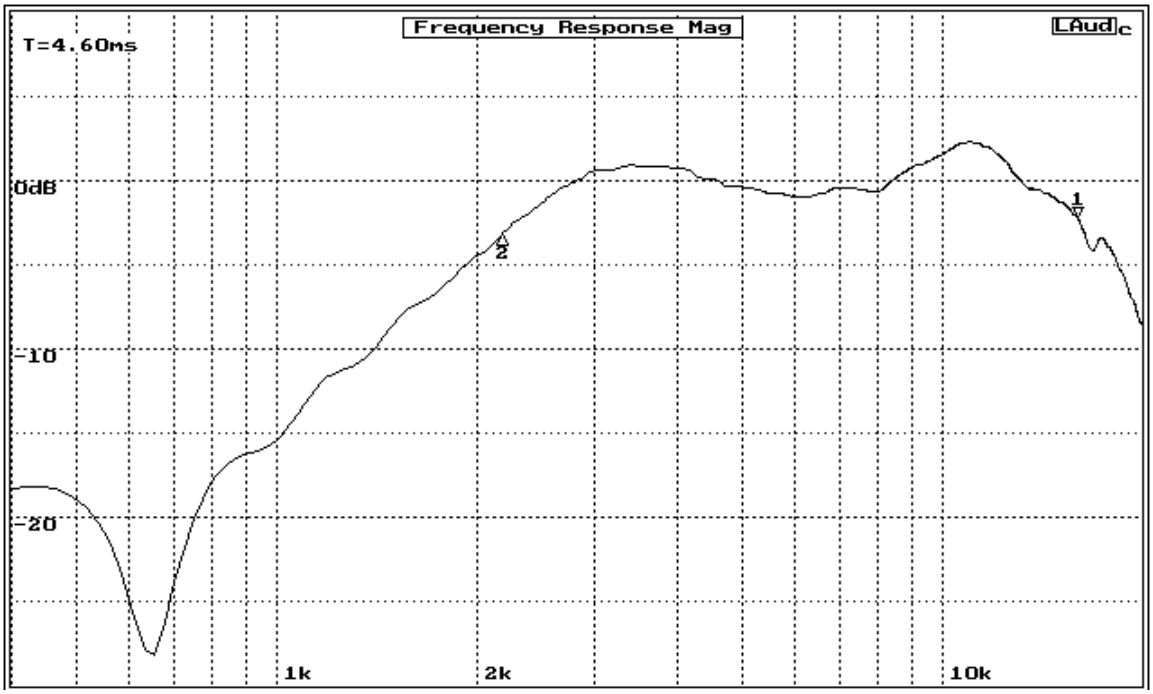

Figure 2-6. Frequency response of a round horn without reflection [7]

Figure 2-6 shows a graph of horn signal strength (dB) vs. frequency. The center to upper right hand corner portion of the graph is relatively smooth, lacking a definite periodic component.

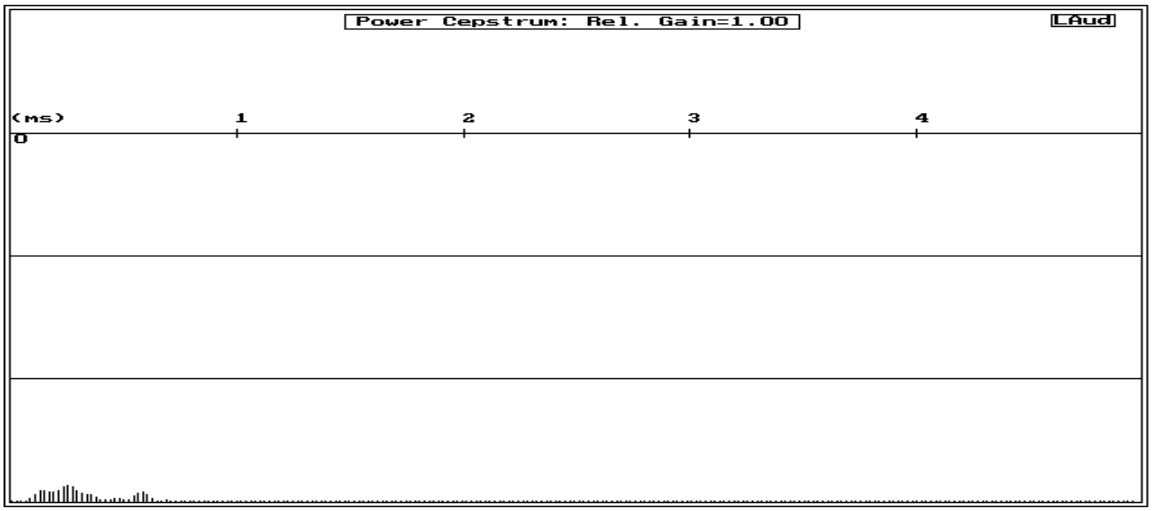

Figure 2-7. Power cepstrum plot of the data from Figure 2-6 [7]

Figure 2-7 shows the cepstrum transformation of data from figure 2-6. Since there is no periodic component associated with this signal, the cepstrum shows very little power.

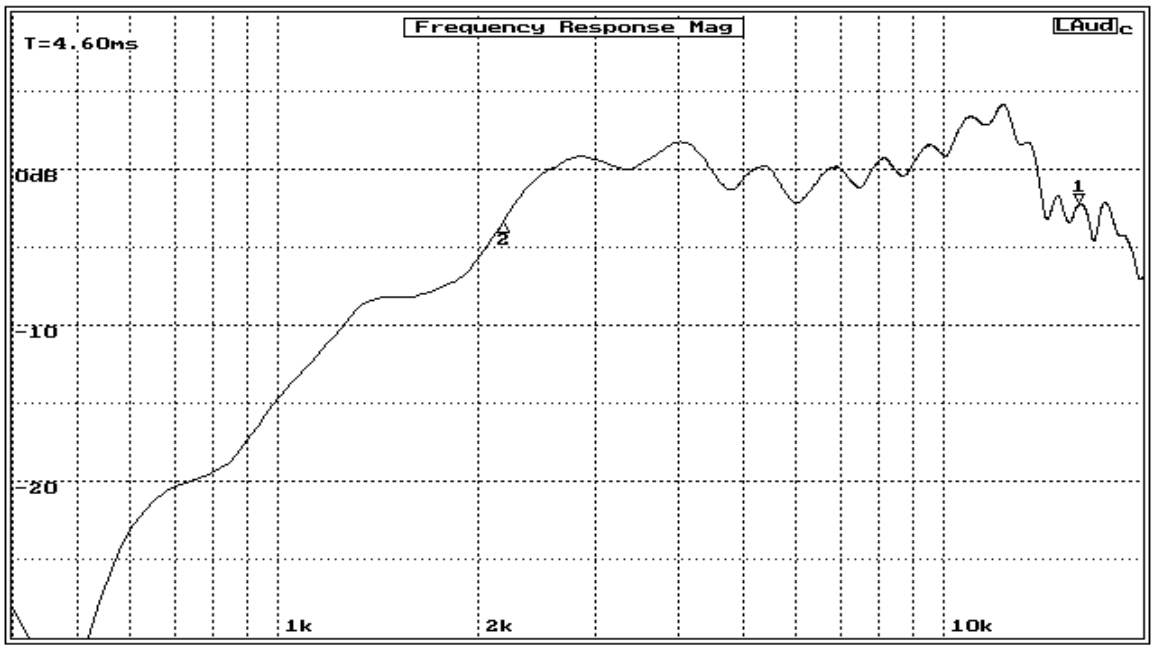

Figure 2-8. Frequency response of the same horn when reflection is included (notice the ripples in the curve) [7]

Figure 2-8 shows a graph of frequency response of the same horn, but with a reflected component. The reflection of the signal can be seen in the upper center to right hand corner of the graph (the oscillations). It is this feature that the cepstrum excels in detecting.

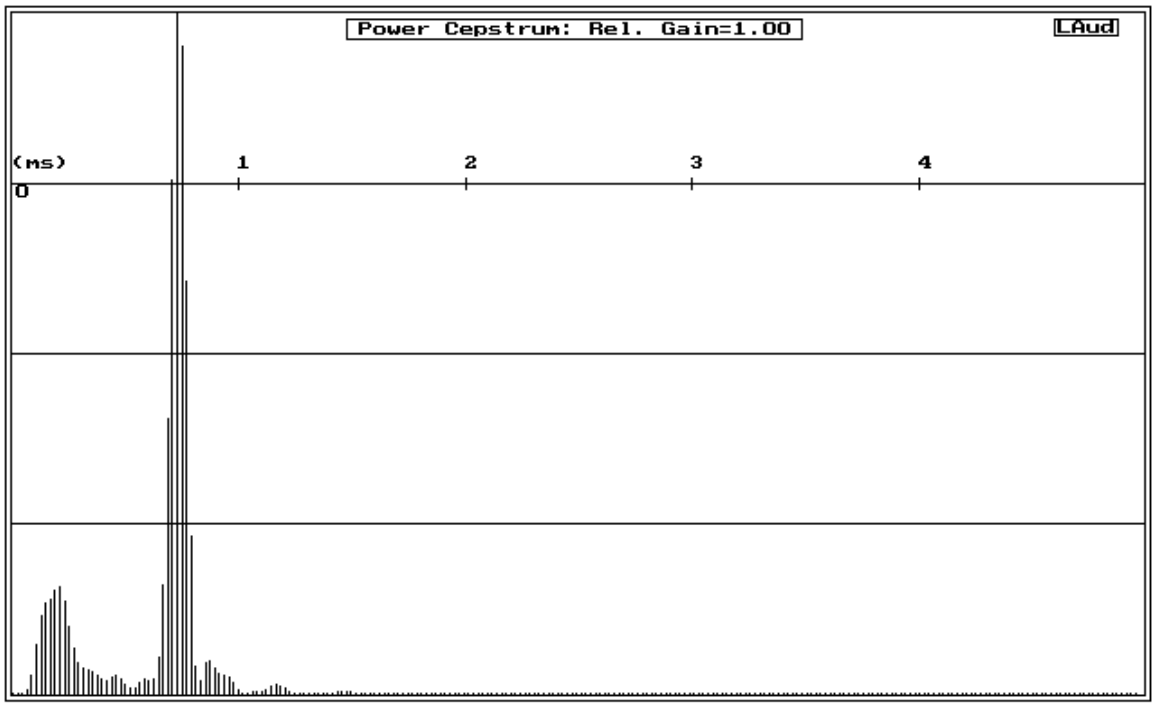

Figure 2-9. Power cepstrum plot of the data from Figure 2-8 [7]

Figure 2-9 shows the cepstrum transformation of the data from Figure 2-8. With the addition of the repetitive signal component, the cepstrum shows a dramatic increase in power. This type of unambiguous signal processing is particularly useful for diagnostic/prognostic applications, provided the presence/absence of repetitive emissions is the sole determinant of proper functioning.

For machinery analysis, usually a number of spectral lines associated with harmonics of the various rotating frequencies of the machinery are identified and their levels are compared to pre-selected thresholds. Spectral analysis has the advantages of a natural and direct association with the characteristics of rotating machinery, relatively simple interpretation, a certain robustness to noise, propagation path, and other sources of distortion, backed by a large body of theory and experience. "Trending," in which the evolution of parameters over time is tracked, is also commonly used; for example, the rate of increase of the magnitude of a spectral line may be estimated or even used to predict the time to failure.

The NSF workshop participants noted that many types of sensors which measure a great variety of physical phenomena are used for both manufacturing and machine monitoring. Mechanical characteristics such as vibration, torque, displacement, shaft velocity, strain and pressure are measured by many different types of sensors, ranging from accelerometers to strain gauges to non-contact displacement pickups using eddy currents. Electrical characteristics such as motor current, capacitance, and RF emissions are often used. Acoustic emissions (AE) play an increasingly important role in manufacturing applications and are under investigation for certain machine monitoring tasks. Visual, infrared, ultrasonic, and X-ray inspection for non-destructive evaluation (NDE) play major roles in certain applications. In spite of this vast array of sensor technologies, there appears to be a constant need for new, more, and better sensors. Many types of sensors have significant limitations, such as restricted bandwidth, nonlinear behavior, or a

susceptibility to saturation. The Air Force goal is to minimize the number of sensors since these are going in smaller JSF aircraft.

Montauk [26] contends that sensors were integrated into electronics systems to accomplish four tasks. The first two tasks related to engine operation and wear. The first task determines when an aircraft's engine performance has deteriorated to the point where the fuel burn changes to something other than its optimal level. At that point, the fuel burn is readjusted to a proper level, instead of letting the condition degrade until the engines needed an overhaul. The second task determines engine damage, hopefully before it impacts the operational schedule or significant consequential damage occurs. The third task assesses how realistic the operational procedures are in order to improve operational safety and enhance profitability. The fourth task is most relevant to this study, as it concerns locating and rectifying faults in complex avionics systems.

This final task evolved into two different types of systems. The first concerned itself with determining how long an aircraft can operate in a particular condition, and the second provided data on which components need replacement. Chu [22] refers to the first system as an Aircraft Integrated Data System (AIDS). The primary goal of AIDS is preventative maintenance, and as such is not usually used to troubleshoot an aircraft, although it may help an experienced user in pinpointing some problems. He refers to the second system as a Central Maintenance System (CMS), and this system is the one intended to allow a mechanic to easily identify faulty avionics units. The exact methods by which either system makes it diagnoses are not mentioned in the article. The CMS

would also trigger the Autonomic Logistics System to provide appropriate spare parts. This trigger and subsequent parts delivery would allow the aircraft repairs to begin as soon as it lands, which improves operational efficiency, and for commercial airlines, profitability.

Moving on to more specific applications, Su [88] proposes an overarching software solution to the prognostics problem. The software would be capable of handling data inputs from any sensor on any system. These inputs would be tied in with a logistics infrastructure to provide the "Autonomic Logistics System" capability. A primary requirement would be the collection and analysis of system data in real-time or near real-time. Faults would be identified using a "Diagnostician" consisting of algorithms that, among other things, would correlate all possible faults to all possible system components. The prognostic part of the software uses predictive techniques which include item specific mechanisms such as neural networks. It also includes linear signal degradation measures, historical conclusions and statistics, and engineering correlations. These correlations are presumed to be the correspondence between sensor indications and resulting system faults. Su does not provide any estimate of when this proposed software solution would be functional.

A number of authors address the issue of the human/machine interface. Dussalt, *et al* [29] focus on the development of management tools to support diagnostic decision making. The current Air Force Integrated Diagnostics policy requires that all faults, either known or expected, be detectable and unambiguously isolated within a system.

This policy does not specify the amount of automation required to be present for system diagnostics. Consequently, a diagnostic system may consist of automatic and manual testing procedures. The paper describes an approach the Air Force is taking to consider what the most appropriate mix of diagnostic measures may be. Similar concerns are expressed by Dean [27].

Thesen and Beringer [91] take a slightly different approach. They use a hierarchical model which represents the user and system as two independent control systems. Communication between these two "independent systems" takes place when each operates with appropriate expectations about the control strategy used by the other. The human must be in-the-loop with the diagnostic system to ensure the automatic recommendations the system makes are correctly understood, and that type I and II errors are not made with regard to the system recommendations (type I - ignoring correct automatic decisions; type II - acting on decisions that are incorrect).

Eilbert and Christensen [30] note that search procedures designed to detect system faults may discern apparent patterns when none, in fact, actually exist. The following figure provides an example of their viewpoint.

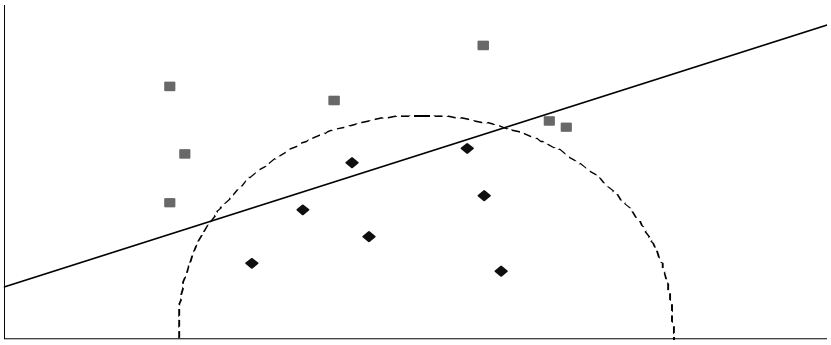

Figure 2-10. Hypothetical bivariate data set [30]

Figure 2-10 shows a data set with complete discrimination ability between both data classes using either a parabolic or circular separator (dashed lines). The optimal linear discriminator misclassifies three events. Because of the small sample size, it is not clear that a quadratic discriminant is preferable, or indeed correct. The implication is that using a search procedure to determine the cause of a particular sensor's report may continue the string of problems (RTOK and CND) already present in the current diagnostic system. This effect may be mitigated to some extent if the system can provide a level of confidence associated with its diagnosis.

2.3 Diagnostic applications

A diagnostic approach using decision trees is presented in [14]. Determining the sequence of steps required to reach a diagnostic conclusion (using a decision tree) has been shown to be NP-hard [41]. Biasizzo, *et al* [14] represent the fault-free operation of a system and the presence of a system fault as two distinct system states. The diagnostic procedure is intended to discover the actual system state using tests which provide

information about system components. The sequence in which the tests are conducted and how information from previous tests is incorporated into the test sequencing procedure is the subject of this paper. Determining this sequence using the minimum number of steps (minimum cost) is known as the test sequencing problem.

Much research has been devoted to this problem. The conventional approach has been to use symmetrical tests. A symmetrical test has only two possible outcomes. Biasizzo, *et al* [14], use asymmetrical tests. An asymmetrical test has more than two possible outcomes.

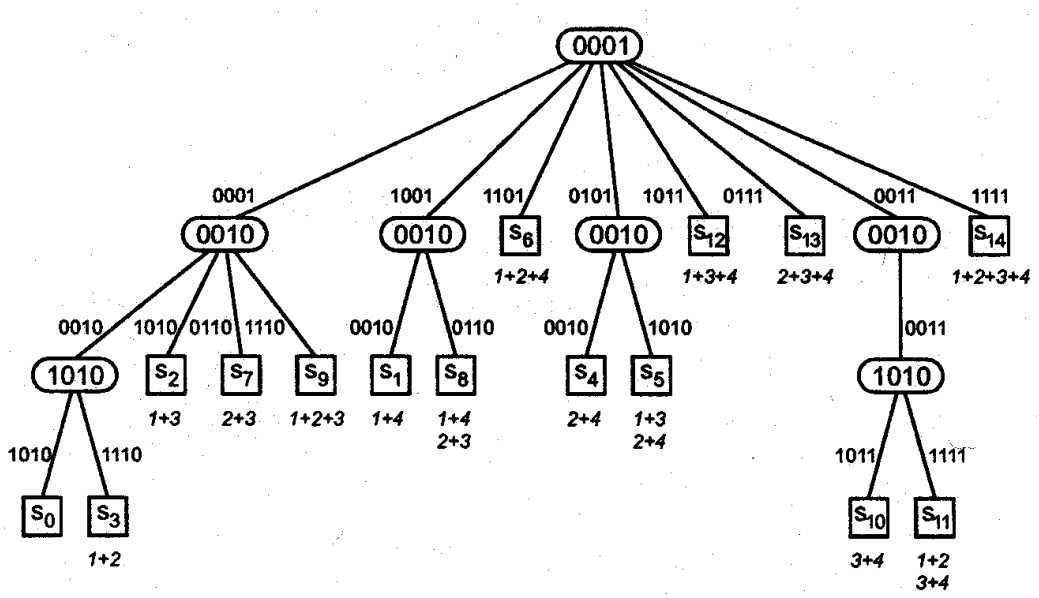

Figure 2-11. An example of an asymmetrical test pattern to determine system state for four components [14]

Figure 2-11 shows a typical asymmetric fault tree used determine which of four system components are faulty. The tree shows the optimal diagnostic test pattern when four tests

are available to test the functionality of the four components. The four components are represented by the four digits in the ovals in the diagram. The first digit corresponds to the first component, etc. The s with subscript indicates which system state the test indicates.

Table 2-1. Test schematic [14]

Test	Component			
	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	0	1	0	1

Table 2-1 shows the test schematic for Figure 2.11. The first test, t_1, determines the status of components 1 and 3, t_2 determines the status of components 2 and 4, t_3 determines the status of component 3, and t_4 determines the status of component 4. Although not explained in the article, it seems tests 1 and 2 cannot determine which of the components they are testing are okay. If the result is faulty, both components are faulty, otherwise one of the two components is okay. It is also not explicitly stated whether 1 represents a fault or normal behavior, but using 1 to represent a fault is implied. It also seems that component 4 is assumed to be faulty given the starting state of the system. Based on these assumptions, state s12 can be determined just by running test 1. Since component 4 is known to be faulty, running test 1 would show components 1 and 3 are faulty, and hence state s12 where components 1, 3, and 4 are faulty is reached.

The ultimate goal is the generation of an optimal diagnostic tree (the order in which the test are conducted based on previous test results). Biasizzo, *et al* [14] employ a Sequential Diagnosis Tool using graph search algorithms on existing decision trees for particular systems. They use a heuristic evaluation function to guide the graph search. The heuristic is an estimate of the remaining cost in the diagnosis procedure from a particular node. They contend the "proof of the admissibility" of this technique is given in [64].

The conventional test sequencing problem is defined as follows [64]:

1. The set of system states $S = \{s_0, s_1,...,s_m\}$ where s_0 denotes the fault-free state of the system and s_i, $(1 \leq i \leq m)$ denotes one of m potential faulty states of the system. In practice, the latter refers to a faulty functional part of the system or to a faulty system function.

2. The set of probabilities $P = \{p(s_0), p(s_1),...,p(s_m)\}$, where $p(s_i)$ is the *a-priori* probability of the system being in the state s_i before the diagnostic procedure is started (i.e., the probability of a fault occurrence described by the system state).

3. The set of available tests $T = \{t_0, t_1,...,t_m\}$ and the associated test costs $c = \{c_0, c_1,..,c_m\}$ which can be measured in terms of time, manpower requirements, or other economic factors.

4. The binary test matrix **D** composed of binary column vectors, $\mathbf{D} = [d_j]$, $1 \le j \le n$, where $\mathbf{d}_j = [d_{ij}]$, dij $\in \{0, 1\}$, $1 \le i \le m$, represents diagnostic capabilities of test t_j. $d_{ij} = 1$ denotes that test t_j fails if the system is in state s_i, and $d_{ij} = 0$ otherwise.

This diagnostic procedure is a sequence of tests to isolate any system state, presented as a decision tree. The problem is to find a diagnostic procedure for a given system at minimal cost. Since a diagnostic procedure is easily described by AND trees, the authors use AND/OR graph search algorithms to determine the best diagnostic procedure.

The preceding definition can be modified to generalize to asymmetrical and multi-valued tests by using the following step.

4. The set of all possible outcomes L of the tests $t \in T$: $R = \{r_0, r_1,...,r_L\}$

The test matrix **D** composed of matrices, $\mathbf{D} = [\mathbf{D}^{(k)}]$, $0 \le k \le L$, where $\mathbf{D}^{(k)}$ is the test matrix associated with the response r_k. Each $\mathbf{D}^{(k)}$ is composed of column vectors:

$$\mathbf{D}^{(k)} = [\mathbf{d}_j^{(k)}],\ 1 \le j \le n \qquad (2\text{-}1)$$

The vector of diagnostic inference for the test t_j with outcome r_k is

$$\mathbf{d}_j^{(k)} = [d_{ij}^{(k)}],\ 0 \le d_{ij}^{(k)} \le 1,\ 0 \le i \le m \qquad (2\text{-}2)$$

where $d_{ij}^{(k)}$ is the conditional probability that the outcome of test t_j is r_k if the system is in state s_i.

Biasizzo, *et al* [14] demonstrate their technique using examples from other published papers. In general, systems with strongly interconnected functional blocks and few internal test points are more difficult to diagnose.

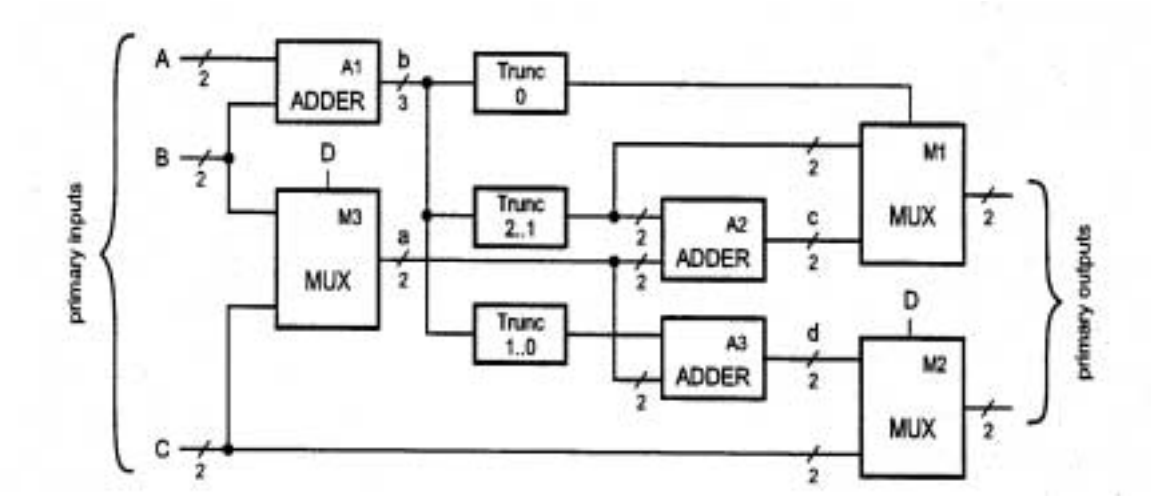

Figure 2-12. Electrical schematic [14]

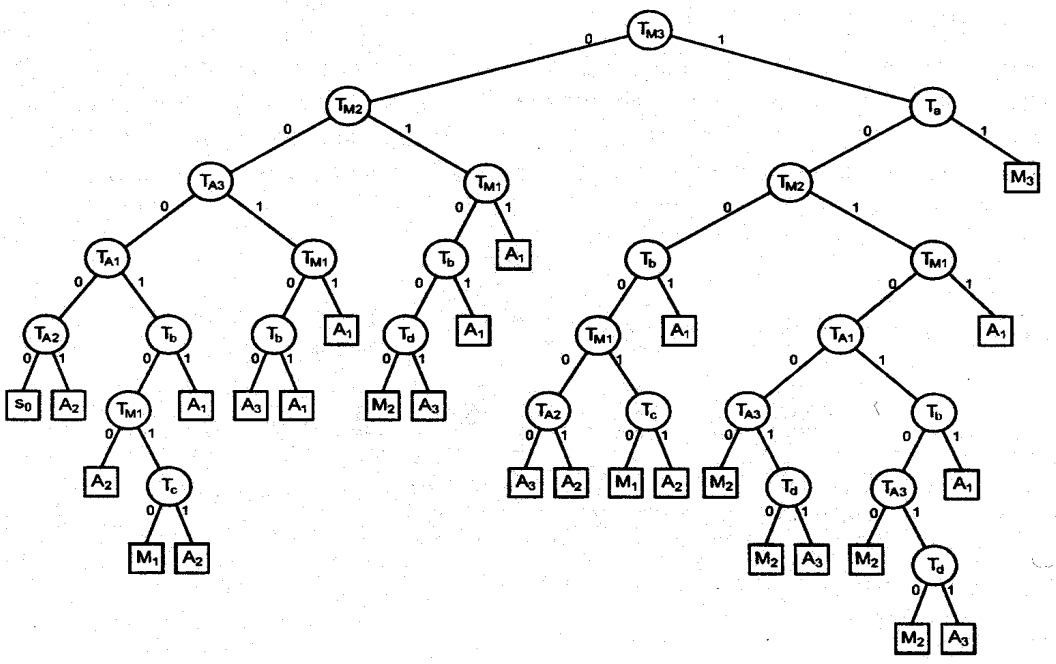

Figure 2-13. The resulting test tree based on the schematic of Figure 2-12 [14]

The electrical schematic in Figure 2-12 and resulting decision tree in Figure 2-13 show how their method works. In the electrical schematic in Figure 2-12, the M1, M2 etc. and the A1, A2 etc. labels indicate a test point. These test points are shown in Figure 2-13 as tests; for example, the node at the top of the tree tests point M3. These test points are transcribed into the optimal symmetric decision tree which would then be used to check the electrical component (shown in the schematic) for faults (non-uniform costs have previously been assigned to each test). This tree is optimal because it incurs the least average cost for a fault diagnosis among all possible trees for this problem.

A similar approach to [14] is found in Bearse [11]. Bearse describes a Diagnostic Inference Model which generates a new fault tree based on original information, allowing for asymmetric outcomes. Other similar approaches include Sheppard [80] and Dill [28]. Sheppard [80] uses case-based reasoning (a historical database) to generate information flow models. Case-based reasoning assumes that similar mechanical system faults produce similar symptoms. A case-based reasoning system starts with a case history, consisting of a number of historical cases. The symptoms and correct diagnosis/repair action are known for each historical case. When a fault occurs in the mechanical system, the symptoms are compared to the recorded historical symptoms. The “nearest neighbor” to the new case is identified, and the diagnosis/repair action used in the historical case is applied to the new case. The resulting system combines the case data with model based systems. Efficient, accurate diagnostic processes are developed from those models.

Dill [28] applies pass/fail limits to discriminate between operable and faulty systems. At times, it can be difficult to know whether the results of a particular test should be classified as a pass or failure. Ideally, pass/fail limits should be set in regions away from expected values observed in functional components and failed components (which presumes a significant gap between the two).

Ben-Basset, *et al* [13] point out issues with just using fault trees. Fault trees tend to cover only the most typical problems for a given system. However, covering these typical problems usually requires a very large fault tree. If a new problem occurs, or the repair recommendation is incorrect, there is no further help available from the tree. If the system which the fault tree covers is updated, even in a minor way, wholesale changes are required to every fault tree to keep things current. Better solutions to diagnostic problems are obtained if different methods (fault trees, physical models, case based reasoning, etc.) are used in concert to provide a diagnosis.

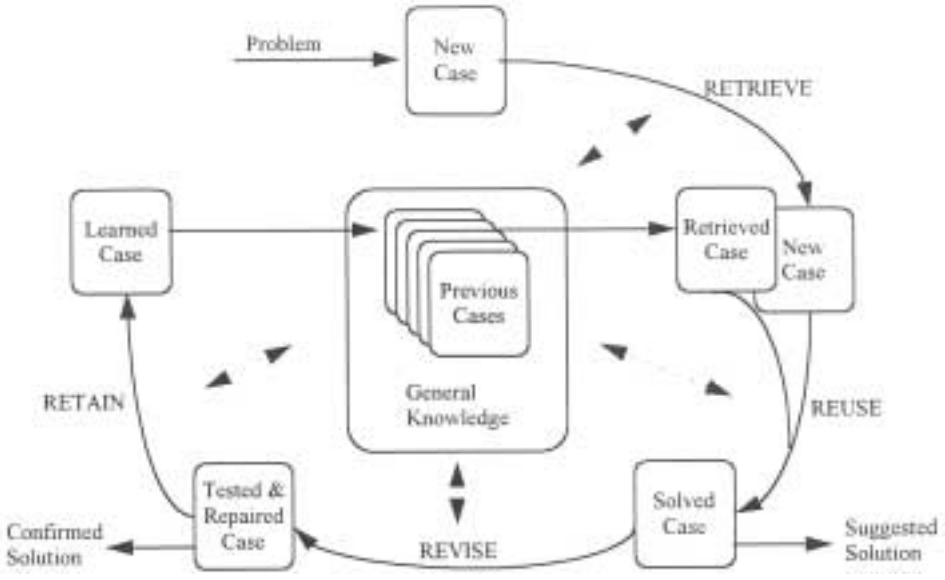

Figure 2-14. A case-based reasoning model [1]

Figure 2-14 illustrates a typical case-based model. A case-based reasoning system assumes that similar symptoms consistently result from identified problems. The general knowledge block contains the case history, consisting of a number of historical cases. The symptoms and correct solution are known for each historical case. When the next problem occurs, the symptoms are compared to symptoms recorded in the historical cases. The "nearest neighbor" to the new case is identified, and the solution used in the historical case is applied to the new case. If changes to the solution are required for this new case, the old solution is revised, and then this new learned case is added to the general knowledge repository.

Authors have also addressed the subject of combining model-based systems with decision tree structures. Ben-Basset, *et al* [12] present a way in which both types of systems are combined to provide a diagnostic expert system. They contend it is more cost-effective, in most real-life applications, to apply case-based reasoning after the system already has some basic initial knowledge of the system domain and the units requiring testing. Their system combines both kinds of reasoning in a module which integrates system knowledge from 4 different sources.

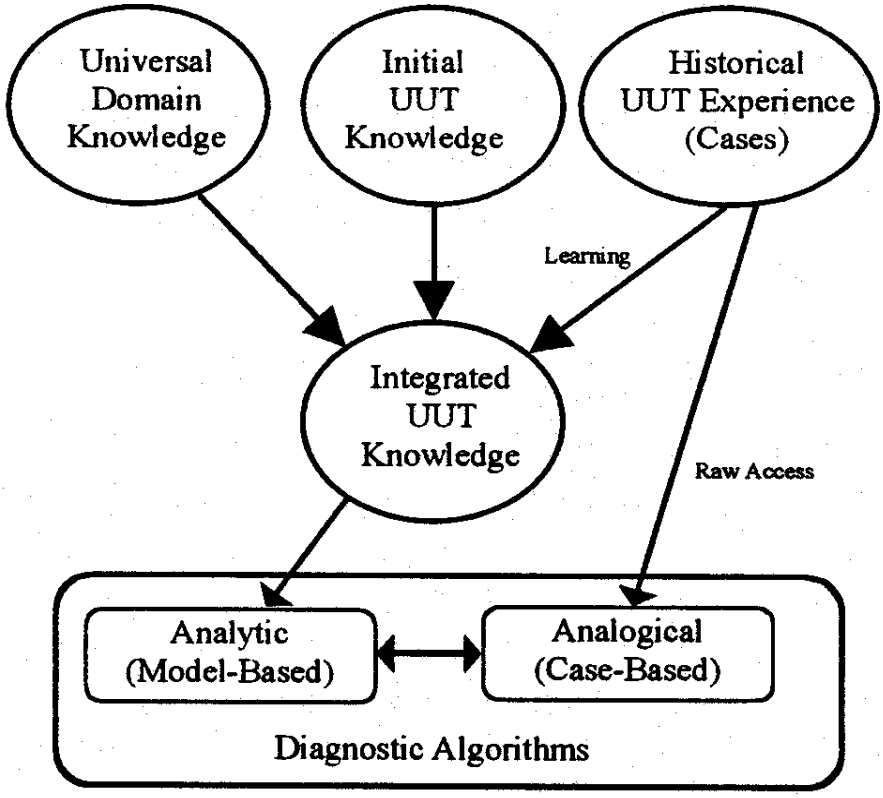

Figure 2-15. Knowledge integration scheme for case-based reasoning [12]

(UUT – Unit Under Test)

Figure 2-15 shows how data flows in the expert system [12]. Universal domain knowledge refers to universal knowledge on diagnostics considerations and processes. Initial unit under test (UUT) knowledge refers to the specifics about the UUT in terms of its structure, function, and relationship between symptoms and faults. Historical UUT experience represents past experiences with this UUT. This information is integrated to form the basis for determining the status of the UUT. The diagnostic algorithms include model-based reasoning, which matches symptoms with probable faults. The model-based reasoning portion is used most often. The case-based reasoning portion compares this

symptom set with previous symptom sets to determine a likely candidate solution. This method is used almost exclusively for new cases, or for cases for which the model-based portion has low confidence in its diagnosis (these two sets of cases should overlap considerably, if not completely).

The main argument against the exclusive use of model-based reasoning is that there are times when there is insufficient knowledge or time to build a model to support efficient and accurate diagnosis. However, a partial model of the system/unit under test is always available. If there is an insufficient number of cases to allow for efficient diagnosis, additional cases can be produced either by simulation or actual experience, and the consequent performance of the model will improve with time. Ultimately, the inference engine of the model-based reasoning function will make most of the diagnostic decisions, and the case-based function will only be used in very unusual cases. This will allow for high levels of accuracy in quick diagnoses. Combining the two disciplines into one model yields the following benefits according to the graphs in Figures 2-16 through 2-18.

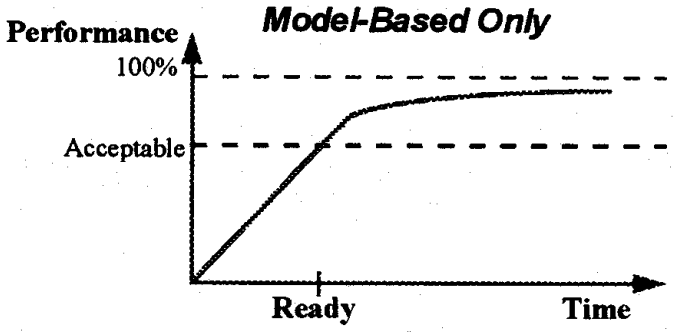

Figure 2-16. Model-based reasoning [12]

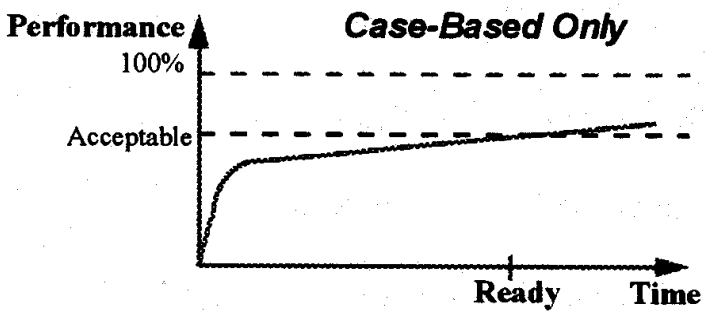

Figure 2-17. Case-Based reasoning [12]

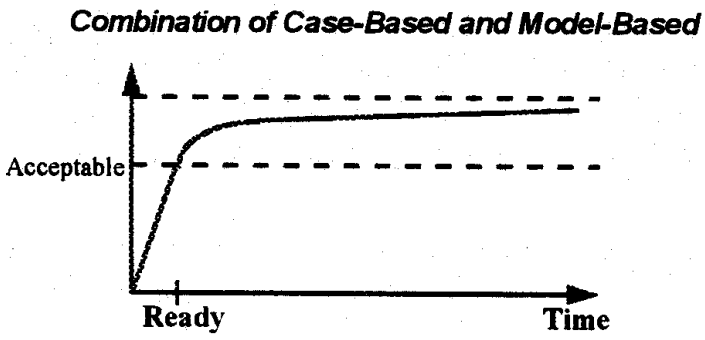

Figure 2-18. Combined reasoning [12]

Of particular interest are Figures 2-16 and 2-17. Case-based reasoning never reaches the level of performance attained by model-based reasoning, while it takes model-based reasoning a longer period of time to reach an acceptable performance level. Unfortunately, the article does not describe the situations the authors analyzed which led to these conclusions. The only statement is these graphs are the product of the authors' analysis of "many real-life situations". The graphs appear to be completely notional.

2.4 Modeling Applications

The most common method for detecting aircraft faults seems to be the application of neural nets. Chu [22] describes the use of this method in conjunction with a statistical classifier (this example was briefly described in section 2.2). Chu's study determines the feasibility of using neural networks to develop troubleshooting procedures for an on-board avionics system, the F-16 Fire Control Radar (FCR) data. The purpose of Chu's experiment was to use a neural net to distinguish between three types of faulty FCRs. The neural network had three layers (input, hidden, and output) and was constructed using radial basis functions with a constant standard deviation, which determined the width of the Gaussian functions used in construction of the neural net.

There are two major classes of neural network models. The first uses nodes (units) which compute a non-linear function (usually sigmoid) of the product of an input vector and a weight vector. The other class of neural networks uses the distance between the input vector and another generalized vector (usually the average of the input vectors) for the computation at the node (unit). Radial basis functions (RBF) are used as activation functions in this second class of neural networks.

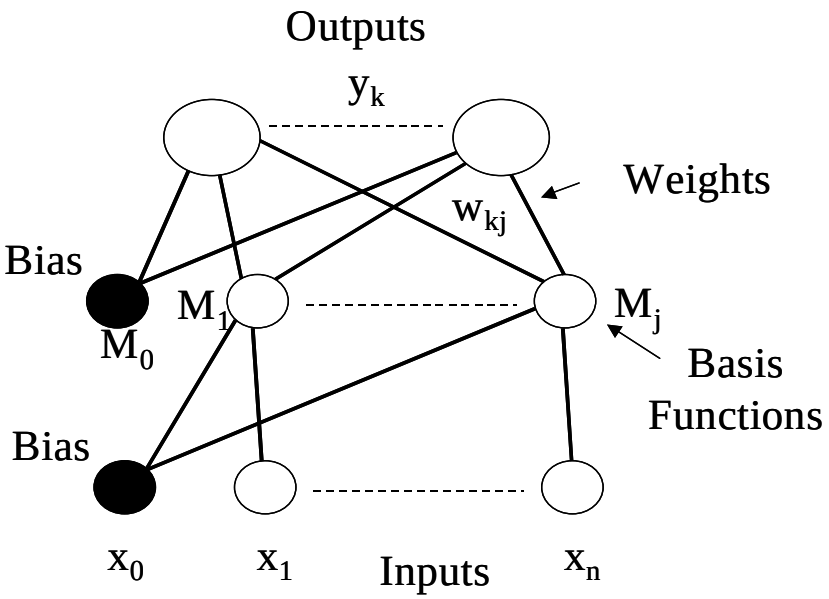

Figure 2-19. Typical RBF network

Figure 2-19 shows a typical RBF network. Each input vector has N inputs, indexed 1 to n, which are combined with M basis functions, indexed 1 to j. There are K output nodes, indexed from 1 to k. More details about RBFs are provided in Chapter 3.

The RBF structure was chosen because a complex classification problem in a high dimensional space, such as this one, is more likely to be linearly separable than one in a lower dimensional space [24]. As previously stated in Chu's paper, an output from the neural net classified the faulty avionics system (which all of these were) as either a "lemon", "bad actor", or "normal". The neural net had 137 neurons in the input layer, 465 neurons in the hidden layer, and 7 neurons in the output layer. The 137 inputs correspond to which of 137 different possible faults a particular radar set exhibited (by implication, the 137 different kinds of faults was not an exhaustive listing). Each radar set consists of 7 Line Replaceable Units (LRUs), and the output vector represented which

1 of the 7 LRUs was faulty. The neural net was trained using fault data from actual systems, and using the "leave one out" approach. This approach trains the neural net using all but one of the input exemplars (466 1x137 input vectors in this study), which is then used to test the accuracy of the neural network. The process is repeated for all the inputs, at a constant standard deviation value. The value for the standard deviation was then varied to determine the optimal standard deviation value (the value which resulted in the most correct classifications). A cost function was also developed to penalize the misclassification of each unit. The optimal value resulted in a correct classification of the faulty LRU 80% of the time. Chu hoped this value could be improved to 90% if more data was available. A similar study was conducted in 1988 [56] which showed that using neural nets to classify faults was feasible.

Keller, *et al* [44] used neural network and fuzzy logic technologies to create models of F/A-18 subsystem/component health. These tools were developed as part of an internal research effort at Boeing to develop an Advanced Onboard Diagnostic System (AODS) along with supporting technologies to reduce CND results which the authors claim were the most frequently occurring result for many subsystems. AODS was envisaged as a collection of software modules which implements subsystem/component health diagnostics, and an integrating system level element which combines the results of the health diagnostics.

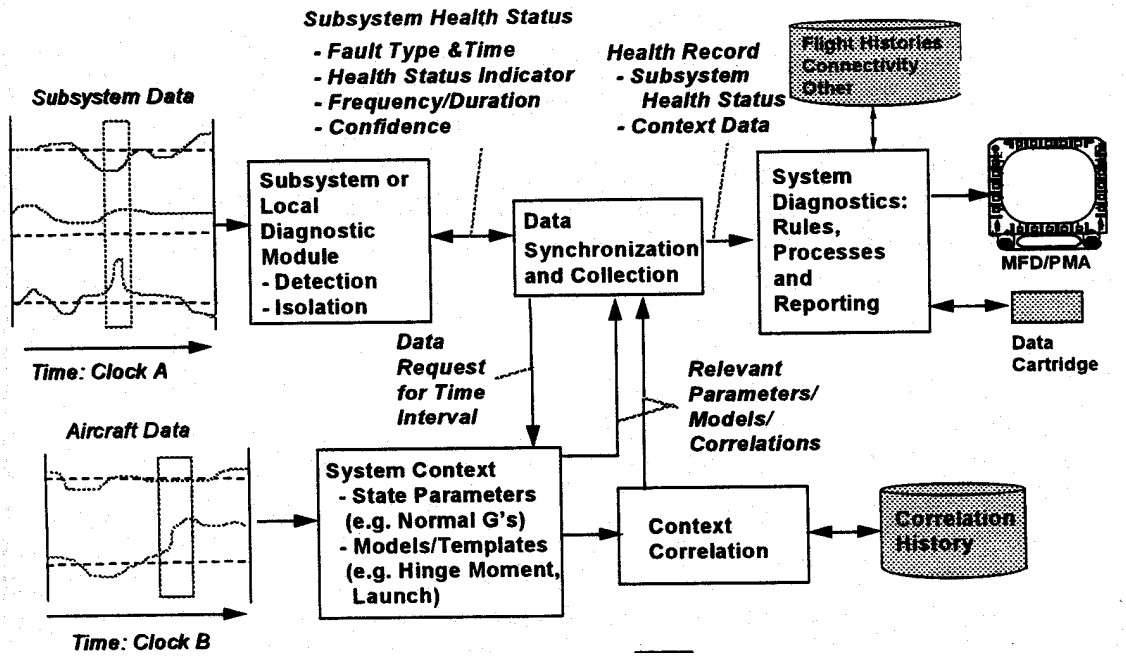

Figure 2-20. AODS top level data flow [44]

Figure 2-20 shows the data flow through the AODS system. The subsystem modules (of which there are many) process real-time subsystem parameters and provide a continuous assessment of system health. The subsystem module reports health status in the form of an incident type, time of the incident, the health status indicator for that type, the frequency/duration of the incident, and a level of confidence. Additional aircraft data which may support later ground processing by the system module or ground testing is also included. The synchronization module captures appropriate information about the status of different system components along with the strength of correlation to the health/fault incident. The system diagnostic assessor then processes the resulting health status record. This assessor is a rule-based system that processes the health status reports. It also maintains a record of previous health status messages. This record of health status

messages is the basis for maintenance recommendations, which are generated either in real-time or offline.

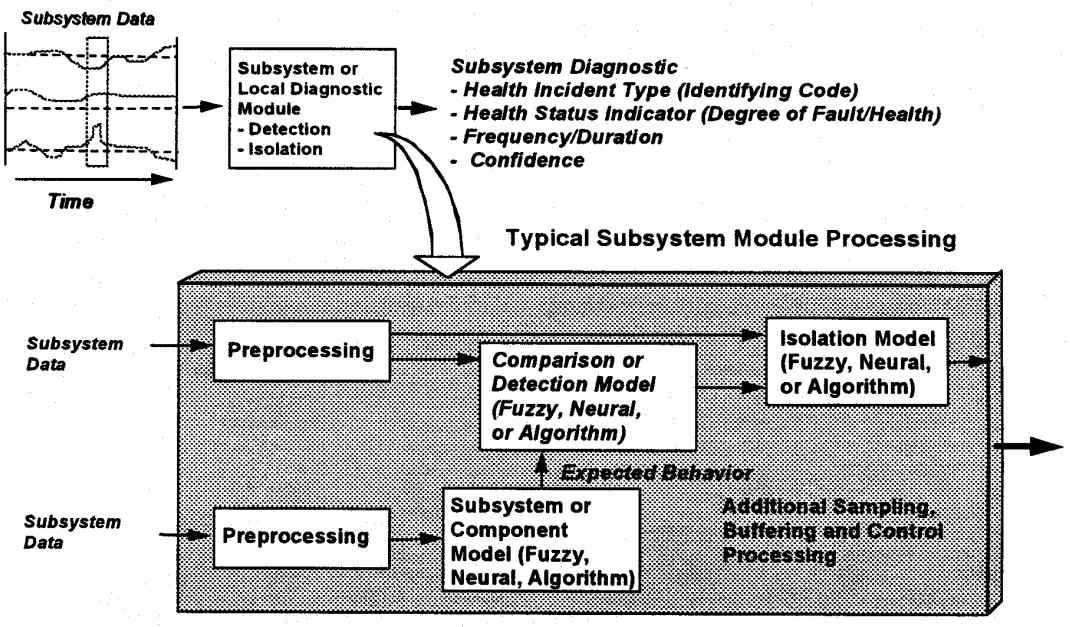

Figure 2-21. Generic subsystem diagnostic module [44]

Figure 2-21 shows a notional subsystem diagnostic module. A neural net or fuzzy model is used to generate an estimate of expected subsystem behavior, and this estimate is compared to actual subsystem outputs. Additional models are used to determine the degree of health of a particular aspect of a subsystem.

The Boeing researchers used both neural nets and fuzzy logic models in the development of this integrating system level element. The neural nets were trained using test cases while the fuzzy logic portion was developed manually (fuzzy logic model development using test cases is still in progress according to the paper).

Test results indicated neural networks provided greater resolution than the fuzzy comparison and detection models, but did not adequately incorporate adjustments based on expert human knowledge, which affected the accuracy of the results. Consequently, the neural networks were used for functional modeling and to map fault patterns to a system health indication. Fuzzy logic models were used in determining event correlation and to develop system health monitoring models which could be adjusted based on expert judgment and intervention. The authors claim this system is a viable architecture; however, no actual test results were provided to support this claim.

Widyantoro, *et al* [96] present an approach using RBF neural networks to detect the presence of air leaks in an engine. Air leaks in a turbine engine occur when a hole appears in a recuperator passage. This is a place where compressed fresh air is pre-heated by exhaust gases before entering the combustion chamber. Potentially, these leaks can result in a long starting procedure, low power, and other problems [93]. The authors [96] began by matching the effects of the problem with the appropriate values from the detection sensors. Three types of engines were selected for diagnosis: engines with no air leaks (normal), engines with small air leaks, and engines with large air leaks. There were 32 sensor readings available from the diagnostic instrument for each engine. The most effective discriminator signals were identified across the 3 engine types. Signals with patterns that were very similar between the engine types, or that were very irregular between the engine types, were not used. Only four signal patterns made the final cut, as shown below in Figure 2-22.

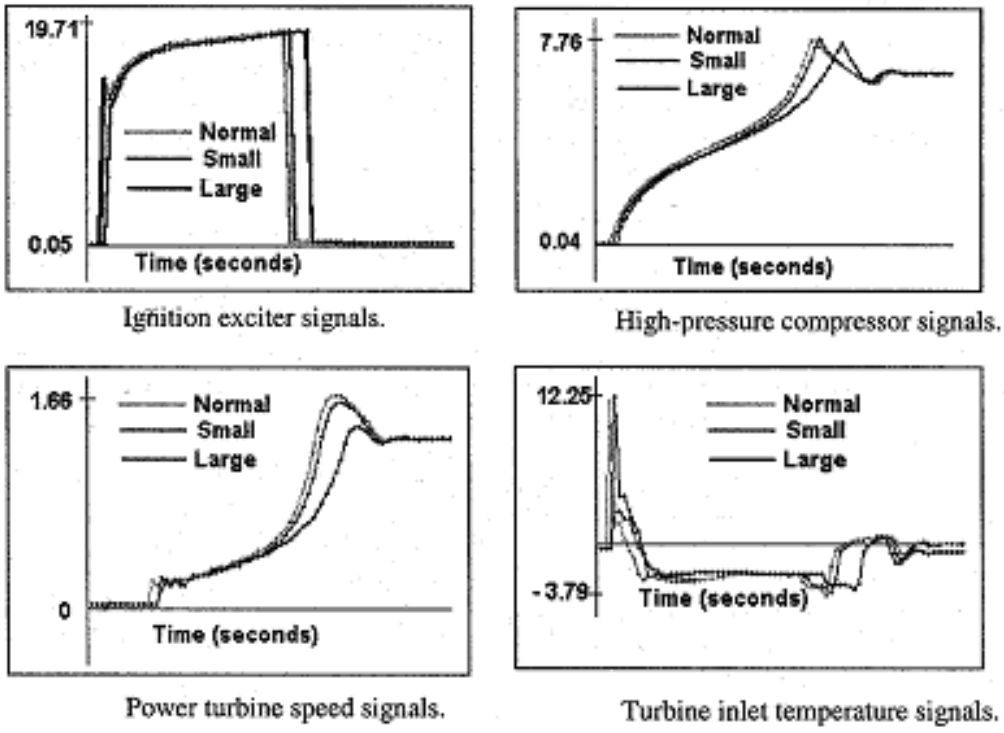

Figure 2-22. Graphs of the 4 signals vs time for the 3 engine conditions [96]

Figure 2-22 shows each of the 4 signals considered the best discriminators between healthy and faulty engines. The graph plots show signal strength versus time. The following paragraphs describe each signal type in detail.

The ignition exciter signals indicate that power has been applied to the ignition exciter to ignite the gas-fuel mixture in a combustion chamber. The power is turned off when the mixture is successfully ignited. A faulty engine (always) takes a longer time to start up than a healthy engine.

The second indicator signal is the speed of the high-pressure compressor of the engine. Among other things, this signal is used for fuel scheduling, and is continuously

monitored during startup. The graph shows that the presence of an engine crack reduces the acceleration of the compressor, and consequently it takes a longer time for the compressor to reach the operational point.

The power turbine speed signal is used to infer (indirectly) the presence of engine thermodynamic inefficiency. In a normal engine, the energy from combusted gases quickly increases the power turbine speed. This acceleration is reduced when an engine crack exists.

The fourth signal is the inlet temperature of the power turbine. According to the authors, it is commonly known that an increased inlet temperature is an indicator of an unhealthy engine, but the reason for this relationship is unclear. The graph shows a delay in the rise in signal strength for an engine with a large air leak, and then a somewhat stronger temperature signal at the end of the time the signals were recorded.

The neural network was trained using a template (generic representative) of each kind of signal for each kind of engine (12 templates in all). The following diagrams show the neural network structure:

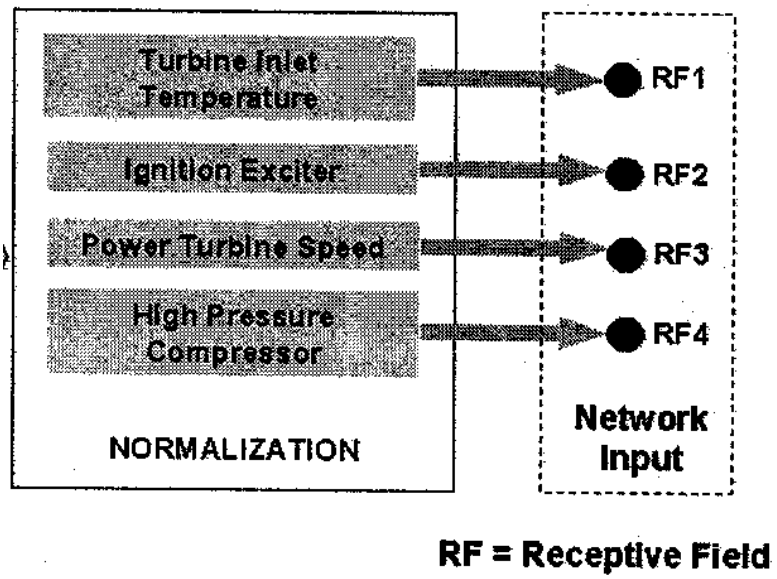

Figure 2-23. Transformation of data into neural network inputs [96]

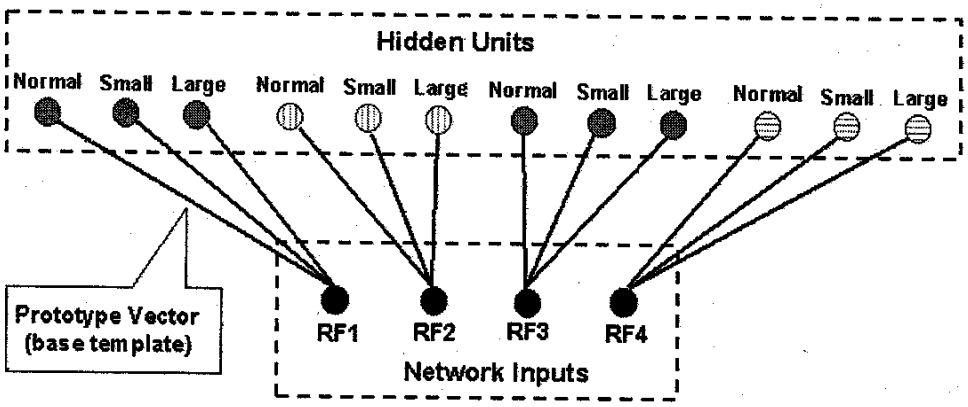

Figure 2-24. Bottom 2 layers of neural network architecture [96]

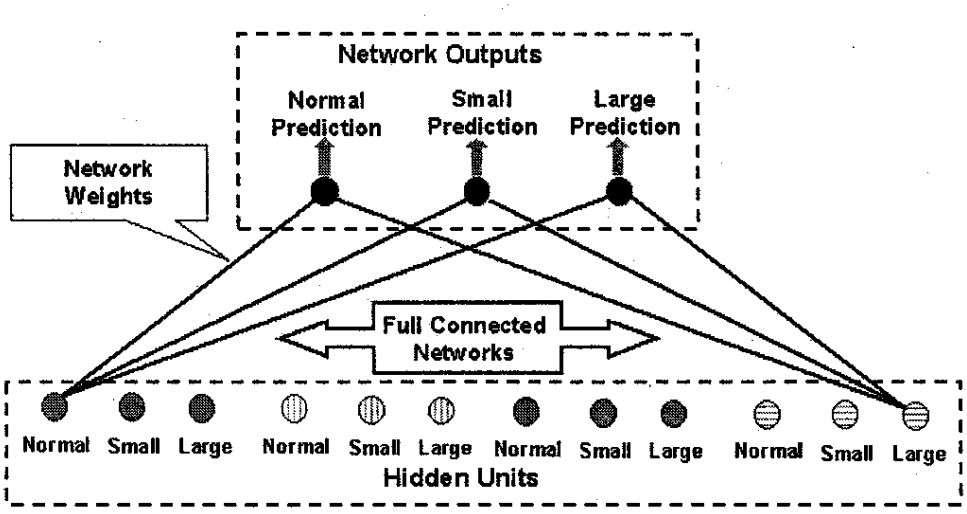

Figure 2-25. Top 2 layers of neural network architecture [96]

Figures 2-23 through 2-25 show the neural network architecture used in this study. Figure 2-23 shows the input scheme. The input consists of the normalized form of the signals reading. The number of input units is $s \times m$, where s is the number of sampled signals and m is the number of discriminator signals. In this example, s = 80 and m = 4.

The activation function in each hidden unit is a Gaussian:

$$\phi_{i,j} = \exp(-PI_i - \mu_{i,j}P^2/\sigma^2) \qquad (2\text{-}3)$$

where I_i is a vector of time series signals from receptive field i, and $\mu_{i,j}$ is the average prototype vector of signal type i that is known to have problem category j. (There are i = 4 receptive fields, shown in Figure 2-23. There are j = 3 problem categories, corresponding to engines with none, small, and large air leaks.)

Figure 2-24 shows how the input data feeds forward to produce a prediction of being from one of the three types of engines. This prediction is based on the linear combination of the hidden units' activation values, given by:

$$O_k = \sum_{i,j} w_{ijk}\phi_{ij} \qquad (2\text{-}4)$$

where $w_{i,j,k}$ is the connection weight between hidden units i,j and output unit k. The purpose of this layer is to perform approximation of the input signals to the prototype vectors. Since there is only one training signal for each signal type, setting $w_{i,j,k} = 1/m$ for $j = k$ and $w_{i,j,k} = 0$ for $j \neq k$, the training data can be perfectly predicted. However, this may cause problems for the neural net when the input data are different from the training signals. To avoid this difficulty, the authors generated six additional data points from the original twelve data points, and used an iterative training procedure that changes the weights to minimize the difference between the target outputs and the network outputs. How this training procedure changed the weights was unspecified.

The network was tested using 8 signals generated by interpolation from the original training data, ensuring that none of the training values were replicated in this test set. The authors computed a target value for each test signal, although how this was done is not explained in the paper. Using the rule that the largest predicted probability indicates the problem, the neural network correctly identified all 8 problems, as shown in the following table:

Table 2-2. Table showing expected and actual experimental results [96]

Signal Number	Output	Target Value	Prediction
1	Normal	0.1250	0.2666
	Small	**0.8750**	**0.9085**
	Large	0.0000	0.1123
2	Normal	0.0000	0.0330
	Small	0.1250	0.1099
	Large	**0.8750**	**0.8646**
3	Normal	**0.8750**	**0.8455**
	Small	0.1250	0.3515
	Large	0.0000	0.0686
4	Normal	**0.6250**	**0.5605**
	Small	0.3750	0.4828
	Large	0.0000	0.0823
5	Normal	0.3750	0.3907
	Small	**0.6250**	**0.6519**
	Large	0.0000	0.0970
6	Normal	0.0000	0.1854
	Small	**0.8750**	**0.7431**
	Large	0.1250	0.1520
7	Normal	0.0000	0.1091
	Small	**0.6250**	**0.4048**
	Large	0.3750	0.2615
8	Normal	0.0000	0.0584
	Small	0.3750	0.1977
	Large	**0.6250**	**0.5023**

Table 2-2 shows the results from the experiment, indicating the neural net performed correctly in each test case.

NASA scientists are also working on using models to interpret sensor data, though with a slightly different emphasis [4]. Their goal is to reproduce sensor readings that are missed, either by the recording unit, or because of a sensor malfunction. The objective of their High Reliability Engine Control (HERC) program is to develop and demonstrate advanced Fault Detection, Identification, and Accommodation (FDIA) algorithms that

will ultimately increase aircraft safety and improve engine reliability. The focus is validation of the sensors which report fault conditions. Validation, here means ensuring appropriate operation of the sensors which are monitoring the machine components, not the actual machine component itself. The authors contend that a complex dynamic system usually uses redundant sensors for measuring critical variables within the machine system. This is done to ensure reliable operation and to improve measurement accuracy. Since some of these measurements can be very critical to judging the health of the system, a redundant sensor set is implemented to ensure the measurement goal is met. This redundant sensor set makes it possible to validate measured data, to identify a sensor failure, and to recover the failed measurement. The authors claim this redundancy can also be met through the implementation of an auto-associative neural network.

The diagram in figure 2-26 shows the test schematic they used to develop and test their neural net, which is shown in figure 2-28:

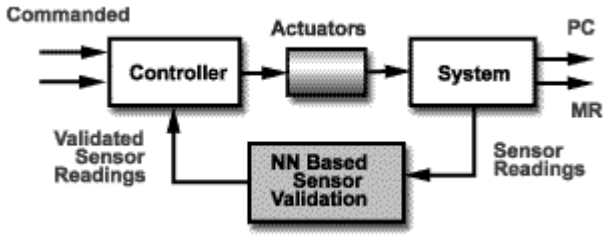

Figure 2-26. Test schematic [4]

The diagram in figure 2-27 shows the measurements taken based on the model shown in Figure 2-26.

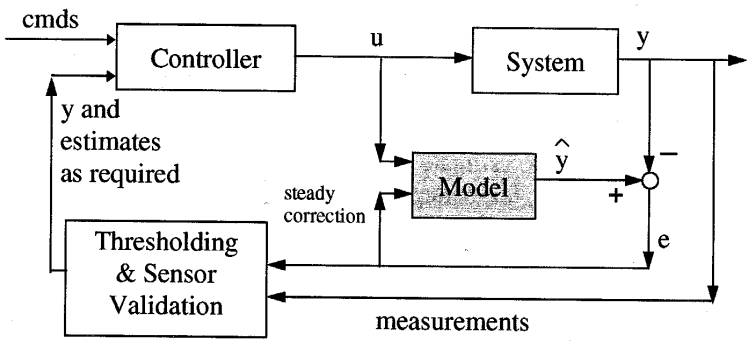

Figure 2-27. Data collection schematic [4]

The data from these sensors were input into the neural network, providing the aforementioned sensor redundancy without the implementation of an additional set of sensors.

The neural network (Figure 2-28) was a feed-forward network architecture with outputs that reproduce the network inputs.

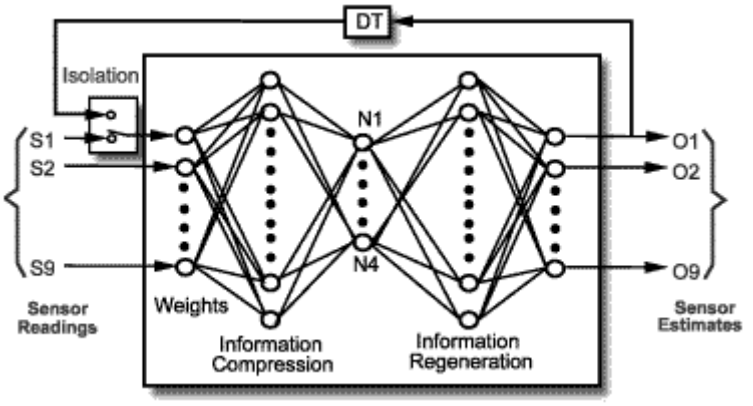

Figure 2-28. Feed-forward neural network design [4]

The diagram in Figure 2-29 depicts the purpose of each network component in more detail:

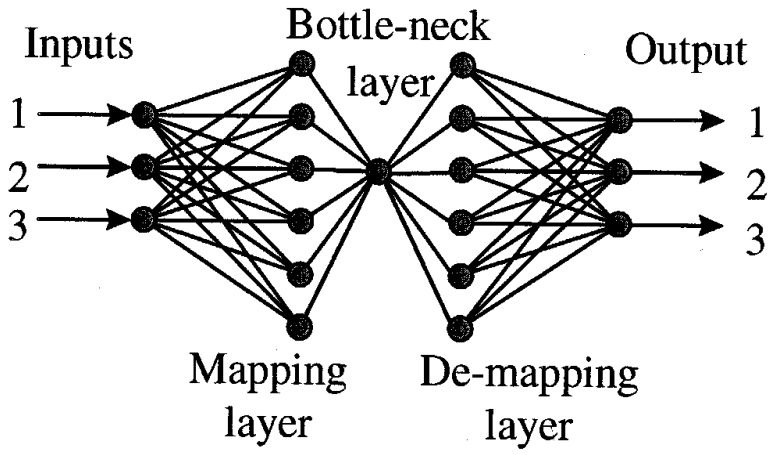

Figure 2-29. Purpose of each neural network layer [4]

As shown above, the left half is the mapping layer and the right half is the de-mapping layer. The bottle-neck layer captures the reduced order (principal components) representation of the data. In the mapping layer, the redundant sensor information is compressed, mixed and reorganized in the first part of the network. In the compression process, the sensor information is encoded into a significantly smaller representation. The compressed information is then used to regenerate the original redundant data at the output. Because of the information mixture, if a sensor fails, other redundant sensor data can still provide enough information to regenerate a good estimate for the faulty measurement. Because of its parallel-processing capability, the neural network can process real-time data for time-critical applications. Also, because it learns by example, the neural network does not require a detailed system model for sensor validation as is

often required. The neural network is then trained to learn the relationships between the inputs (sensors) such that if one sensor is bad, an estimate for that sensor can be found from the remaining valid sensors. The authors present a simple example of three temperature sensors. If the bottle-neck layer is a single node, then the mapping layer performs a weighted average of these measurements. Faulty information in one sensor is thus reduced by a third in the aggregation of all the sensor measurements, resulting in a measurement closer to the actual value.

The preceding example translates into the following general algorithm for a generic data collection scheme. During system operation, if a sensor signal is significantly different from the corresponding estimated value, the sensor signal is considered incorrect and a failed sensor is identified. The failed sensor reading is isolated (eliminated from consideration) by feeding the neural network its previous estimated value. The isolation of a failed sensor enables the neural network to detect subsequent sensor failures, since only properly working sensors are now considered for future measurements.

The automotive industry has attempted to apply sophisticated modeling techniques to diagnostics issues, because of the growing complexity of electronic control systems in today's vehicles [55]. Traditional diagnostic methods are less capable of correct diagnosis in complex systems due to the large volume of information exchanged between the vehicle's processor and the system under CPU control. Marko, *et al* [55], designed a data acquisition system for this high volume of information and used neural nets to analyze it since automobile trouble shooting is essentially a classification problem. The

data consists of inputs and outputs of the vehicle's electronic control system, known as the electronic engine control computer (EECC). This data is a mixture of high speed analog and digital signals which regulate the operation of the engine according to a proprietary strategy. (Exactly what these signals were was not specified.) This strategy optimizes engine performance while adhering to federal emissions regulations. For [56], engine performance data was collected for an engine initially in neutral, and then slowly accelerated. "Certain computational algorithms" (again, unspecified) were performed to give graphs similar to the one shown in Figure 2-30.

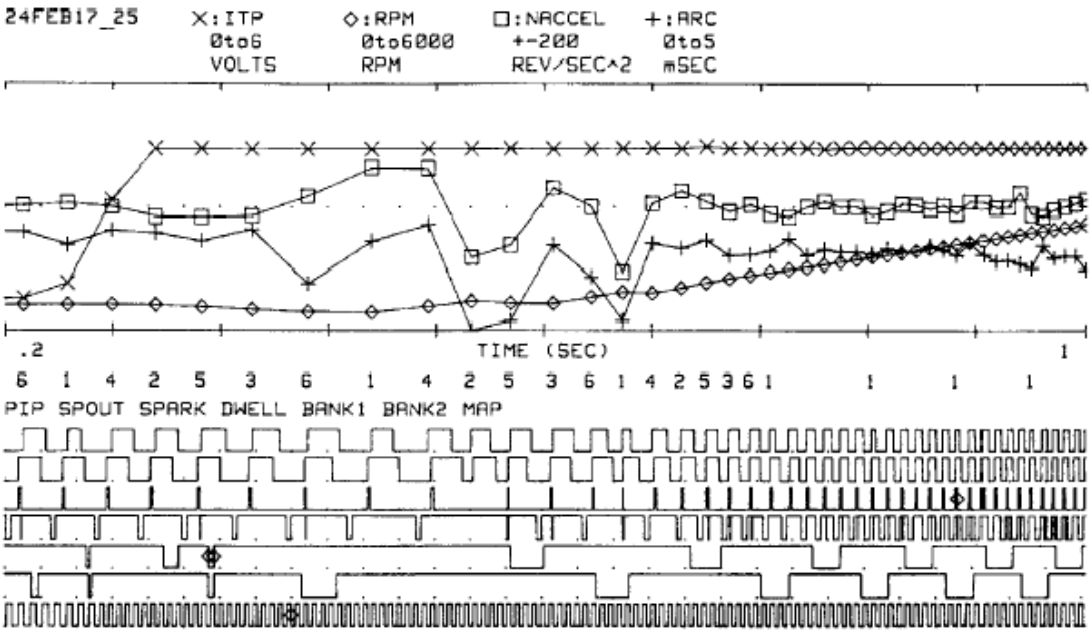

Figure 2-30. Data from a vehicle with no faults [55]

Figure 2-30 shows the data obtained from a vehicle with no faults. Although not explained in the paper, the interpretation of the elements of the top graph seems fairly

self-explanatory, as each element is plotted against time. The bottom part of the graph is less intuitive, but it seems that each square curve corresponds to one of the terms listed just above the first square curve. Other than SPARK, the third curve down, what the other curves are measuring is unclear.

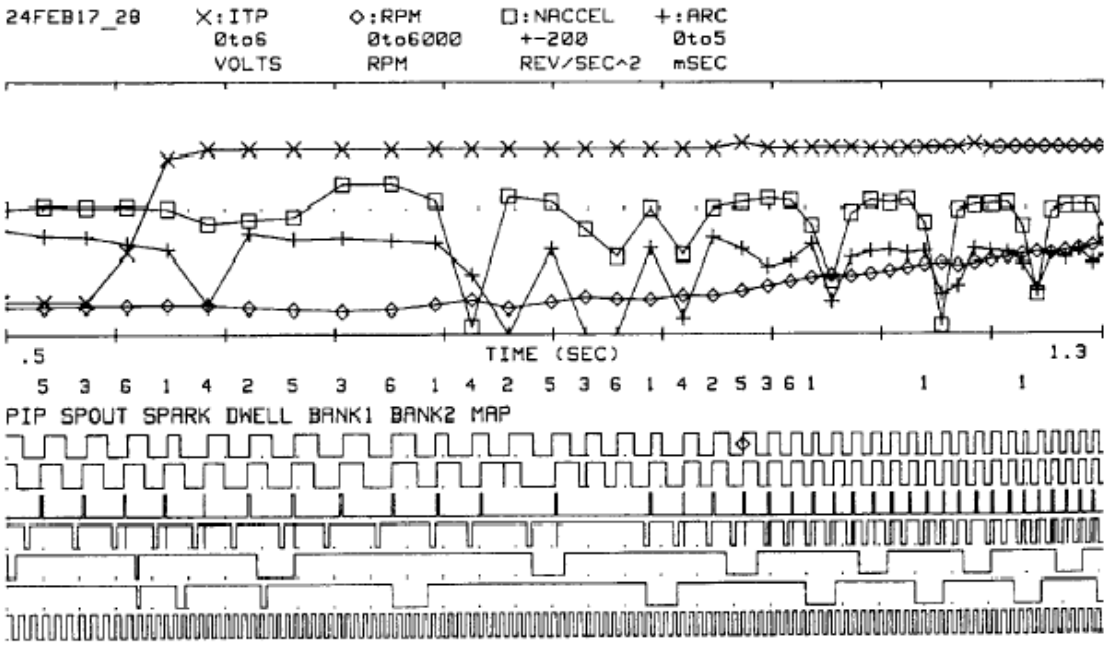

Figure 2-31. Data from a vehicle where spark plug number four is misfiring [55]

Figure 2-31 shows data from a vehicle where a spark plug is misfiring. In this curve, the difference in the ARC and NACCEL curves from the previous figure are clearly seen. No other differences are readily observable, even in the SPARK square curve. This is an example of a problem whose distinguishing features are clearly contained in only 2 data streams, and the features are a radical departure from fault free operation. Because of these attributes, this problem is easy for the fault detection algorithm to detect.

Traditional diagnosis methods require human expertise to formulate rules to guide the service technician through an analysis of the above problem graph to an appropriate conclusion. However, developing these rules is very time-consuming and requires expert understanding of the system operation, failure modes, and how those modes appear in graphs like the one above. The resulting diagnostic approach is still not satisfactory, since the number of resulting rules is quite large, and anything less than a rigorous analysis may result in a misdiagnosis. Furthermore, the number of vehicle-power train combinations is quite large, and each combination undergoes relatively constant modifications to improve performance and reliability. This situation motivates the research into finding better, faster, more accurate diagnosis techniques.

To test their fault detection algorithm, Marko *et al* introduced 26 different faults into the engine and observed the engine's operating characteristics at a fast idle. These faults included a plugged injector, broken manifold pressure sensor, and a shorted spark plug (no comprehensive list was provided). Each fault data set had 52 elements corresponding to the collected information (again, unspecified). 16 sets of data were collected for each of the 26 different faults. An equal number of sets was collected for testing the neural networks after training.

Marko *et al* [14] have found from previous experience (no work cited) that single component failures are much easier to find than multiple failures. In [14], a single fault mode is an unstated assumption, given the composition of the fault data training sets (only one fault at a time). Additionally, it is easier to detect faults if the signature of the

fault is contained in 2 or 3 of the 52 collected signals, rather than consisting of a number of small anomalies spread out over a larger number of signals.

The results presented in [14] show 100% accuracy on classification of their validation set after training the network. The network quickly trained to an accuracy rate of ~95%, but it required a number of modifications to the neural net to achieve 100% accuracy. These modifications included the use of continuous weight updating (not batch learning), and reducing the number of hidden nodes to less than the number of input nodes. This final accuracy result was matched by their best human performer, but at a far slower speed. This approach was then adapted to run on a vehicle in real-time, with similar results. Since the system is passively observing the signals passing between the EECC and the engine, this system may be ultimately capable of providing real-time diagnostics on any vehicle.

Marko *et al* updated their work [55] with a paper addressing the issue of which classifier to use, based on accuracy and expected degree of generalization [54]. In this paper, generalization is defined as a network which correctly classifies an input pattern that was not among the input patterns it was trained on. The neural network is assumed to have been trained on each problem category that may arise in the course of operation of an automobile engine. Of course, the input patterns themselves do not necessarily completely span the space of actual data. Hence, the network must have some capability to generalize by extrapolation—identify vectors near but not within regions occupied by the training patterns.

The data set for analysis remains the same as before—the data stream between the EECC and the engine. The authors chose a specific portion of the data, the portion that the EECC observes when the engine is in operation. 144 patterns were collected, containing 7 kinds of faults. For this data, unlike their previous data set, expert technicians could neither specify an algorithm for classification nor separate the data using graphical visualization.

A variety of different classifiers were tested on this data, including multi-layer feed-forward networks, nearest neighbor classifiers, and binary trees. A binary tree is generally applied to a two-class separation problem. All the data is gathered at the base of the tree (the root node). The data are divided into different groups termed branches, two branches at a time. If all the data along a branch belong to the same class, no further separation is possible. Otherwise, an additional node may be formed, leading to additional separation. A branch may also be terminated if it is judged that further separation is likely to lead to poor generalization. A node that separates into two branches is a terminal node. This process is carried out until all branches terminate. In this instance, the authors used the Fisher linear discriminant to separate the data. They then chose a particular class to separate from the rest of the data. Once that class was separated, another class was chosen for separation. The classes were chosen "shrewdly", so it only took a few branches before a chosen class was completely separated.

A binary tree classifier is considered similar to a feed-forward network. However, the binary tree approach uses far fewer weights, and correspondingly, generates decision

boundaries that are simpler than a feed-forward network. Training and execution of binary tree classifiers tend to be much faster than that of back-propagation and well-suited for time-critical applications.

Their results for this data showed multi-layer feed forward neural networks to be generally equal in classification power to the binary tree method (~90%). The nearest neighbor classifier only had an accuracy rate of 80%. In their conclusions, however, the authors declined to select a best classification method, stating rather that substantially more data is required before conclusions regarding the best classifier are possible.

Besides commercial industry, branches of the armed forces have also been developing prognostic based tools [37], [43], [50], [82], [83]. The following section reviews some of these efforts.

Smith, *et al* [82], [83] discuss the inclusion of a PHM system on-board a Joint Strike Fighter (JSF) aircraft. The JSF program has four pillars; lethality, survivability, supportability and affordability. Smith *et al* contend a PHM system is one of the keys to meeting two of these pillars; providing a supportable and affordable aircraft. As the performance of the fighter begins to degrade, the on-board PHM system is expected to sense these changes and inform the aircraft maintainers of an impending system failure. This system will also inform the maintainers of the actions required to prepare the fighter for its next sortie. The objective is to keep the sortie generation rate high through the use of support systems which allow a proactive response to the needs of the aircraft. This

capability should replace the current brute force approach to maintenance with a more affordable and reliable approach.

These objectives will be accomplished through a Joint Distributed Information System [82]. According to Smith *et al*, this concept is at the heart of the JSF information system. As well as providing internal aircraft data to the maintainers for their proactive action, it is also intended to provide multi-organizational information system operability. This capability will allow for more efficient planning of maintenance actions based on the availability of spare parts, a historical overview of failures allowing for more fighter-specific maintenance actions, and better sortie planning based on the knowledge of when fighters will return from maintenance to operational readiness. This architecture is expected to supply the right information to the right people at the right time.

In a related work [83], Smith, *et al* discuss the development of a Advanced Strike Integrated Diagnostics (ASID) project to develop a program for a "fully integrated systems solution to diagnostics". This program was intended to develop an integrated diagnostic architecture leading to an affordable JSF platform, and to evaluate and recommend integrated diagnostic design tools and techniques. In this context, the term "architecture" means the structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time. The intent is for this architecture to span the entire life cycle of the diagnostic/PHM system.

The ASID program led to an Integrated Diagnostics (ID) Concept Plan which identified the ten best technology maturation programs. These programs were identified as crucial to the success of an integrated diagnostics/prognostics system. They include developing a structural health monitoring system and a engine monitoring system for prognostics health management. Other programs include developing an information delivery system, creating a virtual test bench (for testing new concepts), and maximizing the use of commercial software in the PHM system. Once completed, these technologies are expected to provide superior weapon system supportability.

Schaefer and Haas [75] present a summary of efforts to include Health and Usage Monitoring Systems (HUMS) on the Army and Navy helicopter fleets. The goal of this endeavor is to reduce operational and support costs by transitioning from a time-based maintenance philosophy to a condition-based maintenance philosophy that relies on prognostic techniques to assess the health of aircraft components. Schaefer and Haas present a high fidelity simulation model to analyze the effect of HUMS technology on the existing maintenance process and to provide a means to optimize its use.

Their simulation model represents flight-line level maintenance in a discrete-event simulation. The model includes mission generation modules, a module to simulate in-flight failures, a number of maintenance modules, and a cost module for tracking the amount of resources required for the maintenance activities. The focus of this flight-line maintenance model is to examine how different maintenance policy philosophies impact operational readiness.

Although the Schaefer and Haas indicate their work is not fully complete, their initial simulation results show that specifying a certain range of performance for a helicopter system, and scheduling maintenance when that system is no longer performing in that range, can minimize maintenance costs. Additionally, their model shows that there is a limit to the utility of advanced diagnostics for certain helicopter components which affect other components. For example, it may be possible to specify helicopter operation procedures to produce low vibration levels to defer the maintenance action of balancing the main helicopter rotor, but the requirement for low vibration levels will affect the operational capability of the helicopter. In this case, specifying a particular performance range for minimization of maintenance activity is counter-productive.

The Office of Naval Research (ONR) has been developing a distributed shipboard system for diagnostics and prognostics on systems with rotating equipment [37]. Their system, termed a Machinery Prognostics/Diagnostics System (MPROS), is composed of two parts. The first is a data collection system, which collects data from vibration, temperature, pressure, electric current, and other (unspecified) sensors. The collection system also includes local intelligent signal processing devices called Data Concentrators (DC). The second part is a centrally located subsystem called the "Prognostics, Diagnostics, Monitoring Engine" [sic], or PDME. This system combines the results from the DCs to provide the best possible diagnosis.

The specific shipboard application is centrifugal chillers (air-conditioning systems). These systems combine several rotating machinery equipment types to form a complex

system with many different parameters available for monitoring. The parameters that are chosen for monitoring are combined along with diagnostic and prognostic algorithms into the MPROS. Since the MPROS can diagnose each component part of this system, as well as the whole system, it should be readily extendable to monitor any pump, motor, or compressor in the naval fleet. Additionally, there are a large number of facilities, both military and industrial, that use centrifugal chiller-based air-conditioning systems.

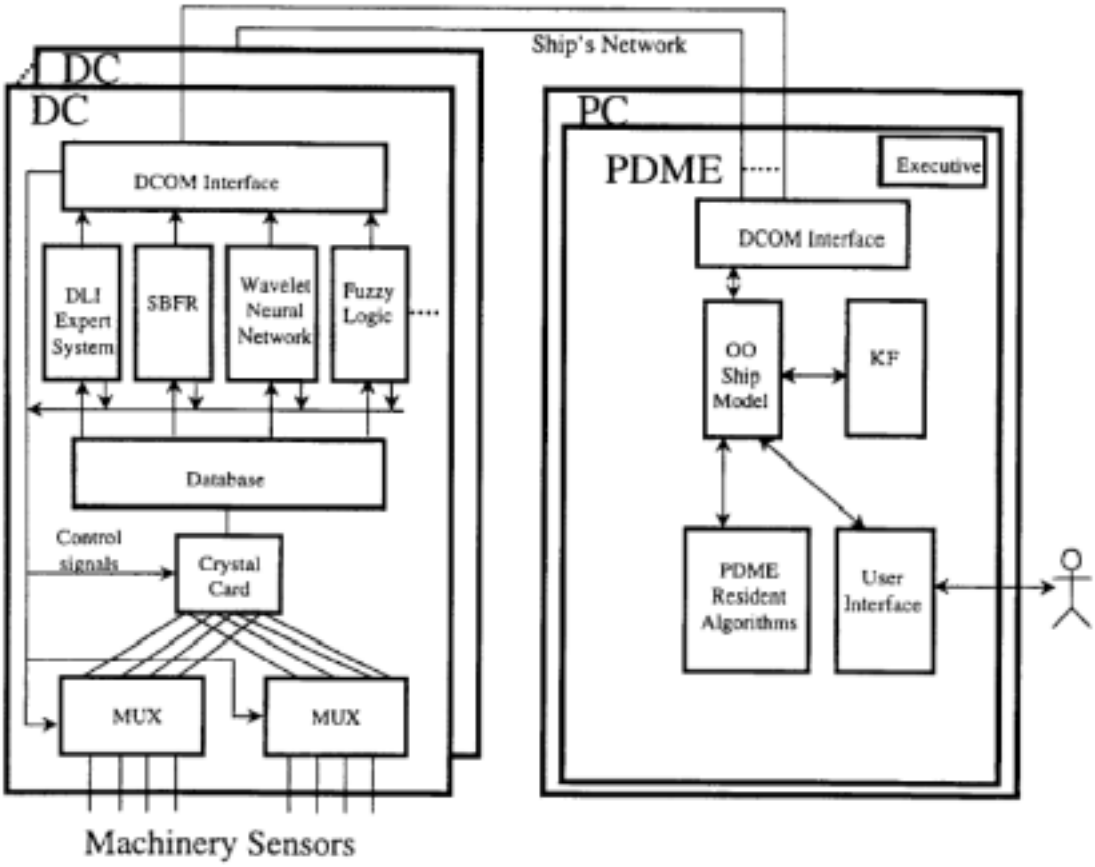

Figure 2-32. MPROS system [37]

In Figure 2-32, the sensors connected to the machinery are shown at the lower left. The sensors for a particular system capture the failure characteristics of a specific failure mode. There are two basic kinds of sensor data. The first kind includes low-bandwidth measurements, such as those from process variables, temperature, pressure, etc. Failure modes associated with this category usually develop slowly and consequently, data can be sampled at low rates without losing the pattern of a particular trend. The authors believe this kind of failure is best detected with a fuzzy-based rule set as an expert system. The second kind includes high-bandwidth measurements, such as vibrations and electrical current data. This type of data requires a much higher sampling rate in order to capture enough information to appropriately categorize the failure signature. These kinds of faults are best detected with a feature extractor/neural net classifier. The ONR used this second approach for this particular problem.

This data feeds into the left hand box , the DC (Data Concentrator), whose components are shown. Of most interest is the Database and the four data processing algorithms. The database stores information configuration, machinery configuration, test schedules, test measurements, diagnostic results, and condition reports. The DLI expert system (PredictDLI is a company with a Navy contract to develop these kinds of algorithms) is a vibration expert system adapted to run in a continuous mode. It detects departures from steady-state norms. The SBFR (State Based Feature Recognition) algorithm facilitates recognition of time-correlated events in multiple data streams. The wavelet neural network also analyzes vibration data, but it focuses on drawing inferences from transitory phenomena rather than steady-state data. The fuzzy logic algorithm draws diagnostic and

prognostic conclusions from non-vibrational data. Since these algorithms overlap in some areas, there is the potential for conflicting diagnoses (as well as reinforcing ones). The authors use Knowledge Fusion (KF) to combine the conclusions from the algorithms. The authors consider KF to be the coordination of reports from a number of sources, as opposed to the correlation of single platform data (similar to the function of the DC).

The PDME (Prognostics, Diagnostics, Monitoring Engine) contains the KF component, as well as resident algorithms for performing PDME functions and a couple other features. The DCOM and user interfaces interact with the DC DCOM element and the user, as one might expect (DCOM stands for Distributed Component Object Model, a communications standard developed by Microsoft). The OO Ship Model, or Object Oriented Ship Model, represents parts of the ship, such as the compressor, chiller, deck, machinery space, etc. It models the physical, mechanical, and energy characteristics of the machinery being monitored. It also stores diagnostic conclusions from the four algorithms and the KF component.

The system has been tested successfully in the laboratory, and the authors are preparing to install it on a hospital ship in San Diego [37].

The Army is also developing prognostic tools [43]. Their main emphasis is the M1A1 Abrams tank, and the diagnosis of fuel flow problems in the tank's gas turbine engine. The system collects data available in the turbine engine startup sequence to diagnose three types of faults in the main metering fuel valve: bouncing valve, sticking valve (later

referred to as fuel flow error), or stuck valve. These faults prevent fuel from being delivered to the tank's engine in accordance with a fuel flow algorithm, which sets fuel flow based on a number of different criteria, including the current demand on the engine, available air from the intake, etc.

Fuel flow faults can be detected in the signals from the Electronic Control Unit's (ECU) diagnostic connector. The ECU is an analog computer whose fuel flow algorithm is dependent upon throttle position, ambient air and turbine inlet temperatures, and compressor and turbine speeds. These voltage signals reflect the status of the Electro-Mechanical Fuel System (EMFS), which responds to ECU commands. The EMFS is a fuel metering device that delivers fuel to the engine under the management of the ECU. Each of the variables previously mentioned (throttle position, ambient air and turbine inlet temperatures, and compressor and turbine speeds) has a representative voltage signal available for collection and consequent analysis.

The initial data sets were obtained by starting the tank engine and recording the appropriate sensor data. Most of these data sets were fault-free, since the fuel flow problem apparently rarely occurs upon startup. Because accurately training a neural net on a particular problem requires a number of cases exhibiting the actual phenomena associated with the problem, the authors [43] seeded faults into the startup procedure. Additionally, they "translated" some data sets from fault-free starts to faulty starts (methodology unspecified).

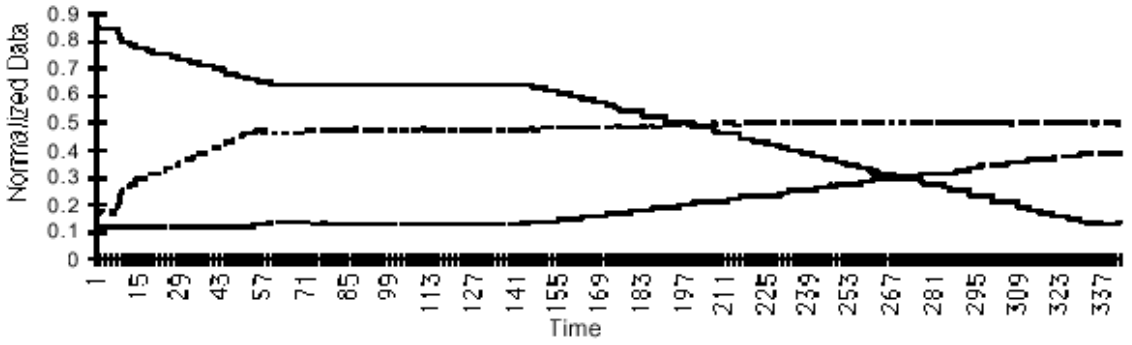

Figure 2-33. Normal tank start data [43]

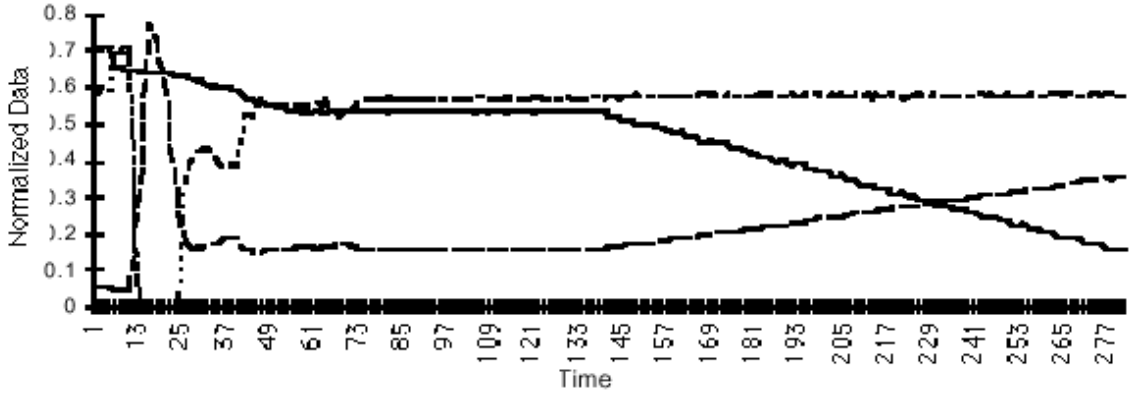

Figure 2-34. Bouncing valve tank start data [43]

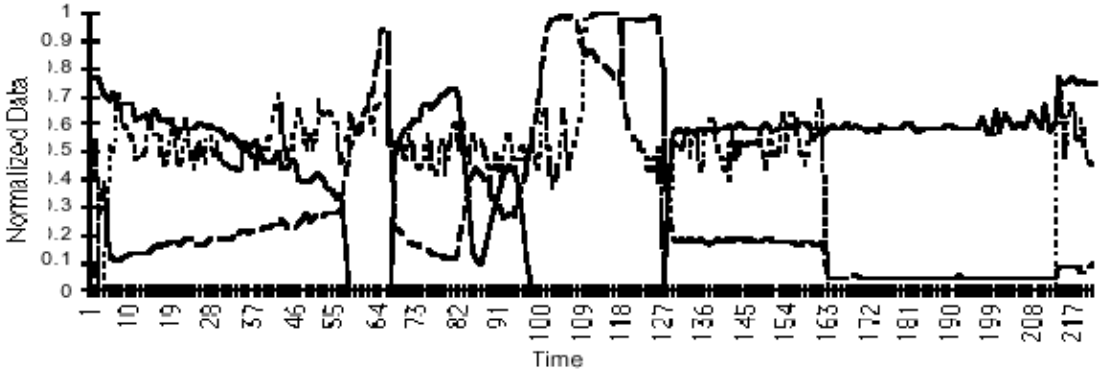

Figure 2-35. Stuck valve tank start data [43]

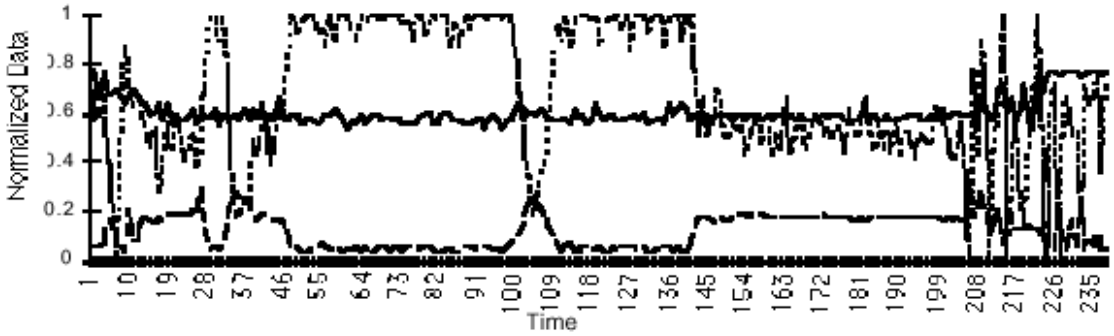

Figure 2-36. Fuel flow error tank start data [43]

Figures 2-33 through 2-36 show some of data that was collected. It is relatively easy to discern based on the collected and processed signatures what kind of fault is present. The curves include 3 different sensor streams, although the sensor streams are not individually identified. It is likely that they are graphs of the variables previously mentioned (throttle position, ambient air and turbine inlet temperatures, and compressor and turbine speeds).

The neural net tool used for the fuel valve diagnostic was the NeuroWindows Artificial Neural Network (ANN) simulator software. Visual Basic was employed as a user/computer interface development tool. Using the data sets as described above, they trained the neural network to distinguish between the three fault conditions. However, simply using the sensor values as the one input to a simple feedforward ANN does not capture all the information available in the time domain. To capture time dependent information, the input to the ANN included first derivatives of sensor values and first derivatives of differences between pairs of sensor values. How these first derivatives were calculated is not mentioned in the paper.

Based on the analysis by the ANN system, TEDANN (Turbine Engine Diagnostics Artificial Neural Network) determines which fuel flow voltage readings are out of tolerance with normal operational parameters. Upon this determination, TEDANN will display either a fault status message identifying the EMFS faults or a message stating that the EMFS is fully operational.

Table 2-3. TEDANN's diagnostic performance (severity) [43]

Diagnosis (across)/ Actual Conditions (below)	Bouncing valve	Stuck valve	Fuel flow error
Bouncing valve	1.00	0.00	0.00
Stuck valve	0.00	0.98	0.00
Fuel flow error	0.00	0.00	1.00
No fault	0.03	0.02	0.08

Table 2-3 results indicate TEDANN does remarkably well in diagnosing the individual faults. The entries in Table 2-3 are the neural network's assessment of how severe the fault is, using the following scale:

0.00-0.25 - no fault (normal)
0.26-0.75 - warning (fault)
0.76-1.00 - critical (fault)

The entries in each cell are an average over several data sets (variation is not specified). The table does show a completely accurate diagnosis based on the severity scales—all actual fault conditions would be detected and correctly diagnosed, and all actual non-fault conditions would be diagnosed as such, since the resulting severity figures are less than

the 0.25 threshold. The authors are continuing to refine their study, and hope to extend it to other tank components and Army systems.

Logan [50] describes a prognostics system currently in use. This system is assisting the Navy reduce both manning and maintenance costs. To that end, the Navy is implementing ship designs which support minimum crew sizes and minimum maintenance requirements, while maintaining mission readiness goals [50]. A major component of this strategy is the development and implementation of predictive maintenance (prognostic) systems. These systems can be exploited for monitoring, control, and condition assessment of critical shipboard systems. Artificial intelligence methods will provide the necessary assessment capabilities. These capabilities include the abilities to:

- Be initially deployed using existing experiential and empirical knowledge;
- Function properly with missing, noisy, or corrupted measurement data;
- Compute and assess uncertainty measures following valid statistical techniques;
- Infer measurements that are either too costly or too difficult to acquire.

Logan *et al* [50] believe artificial neural networks are particularly well-suited to diagnostic applications. They contend that neural nets can classify novel input patterns not included in training data, and that neural nets are tolerant of noisy or incomplete input patterns. In addition, system state recognition is usually performed in real time. Of course, the critical aspect of deploying neural networks is access to training data that

adequately represents the input/output state space the network is likely to encounter in the specific application.

There are problems with accumulating neural network training data. Since good maintenance practices tend to prevent failures from occurring, actual failure data is extremely scarce and very expensive to collect and/or create. The fault coverage of actual failure data is typically very narrow and it may require many years of data collection to obtain an adequate data set for neural network training. Unless the data is collected under controlled or known conditions, historical failure data may be incomplete or include unreliable measurement values. Additionally, the data will be insufficient to provide coverage for all possible machinery faults which might occur. If this data used for training the neural network, the network's fault classification performance may be adversely affected. Also, typical monitoring systems do not store data at adequate sampling rates to ensure that sufficient data are recorded to accurately classify the failure event, as well as events leading to the actual failure.

Logan *et al* [50] recommend an alternative, hybrid approach. The engineering knowledge of domain experts can be used to construct a diagnostic knowledge base suitable for neural network training. This can be accomplished by conducting a comprehensive Failure Mode And Effects Analysis (FMEA) on the appropriate mechanical system. A FMEA provides a comprehensive listing of probable failure modes of all "major" mechanical system components, where "major" is defined as the level of detail appropriate for that particular system. This information is obtained from

interviews with engineering crews and maintenance personnel. It also includes information on all available sensor measurements, and identifies the fault/symptom relationships required for an effective monitoring program.

The neural network of choice for this application is a probabilistic neural network (PNN). It has a number of favorable characteristics [96], [2]. PNN training is effectively instantaneous, as opposed to the slow error convergence training of other neural network techniques. Besides the reduced effort for system commissioning, instantaneous training is extremely attractive for allowing training data set modifications and PNN retraining in the field by end-users. The PNN outputs the fault classification probabilities, meaning it is easy for the end user to interpret the result. PNNs have strong generalization capabilities (as do other neural networks) which can handle situations in which one or more input variables are missing or are corrupted. This makes the method attractive for real-world applications where sensor failures occur on a regular basis, such as in a shipboard environment. Also, PNNs can be initially deployed using existing experiential and empirical knowledge and can be readily updated as new knowledge is acquired.

A PNN is designed to estimate the class conditional probability density functions according to the following equation:

$$f_A(X) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{m}\sum_{i=1}^{m}\exp\left[-\frac{(X - X_{Ai})^T(X - X_{Ai})}{(2\sigma^2)}\right] \qquad (2\text{-}5)$$

i = pattern number

m = total number of training patterns (1/m is a normalizing constant)

X_{Ai} = i^{th} training pattern from category A

σ = "smoothing parameter"

p = dimensionality of measurement space

Equation 2-5 defines the PDF for each fault as the sum of several multivariate Gaussian distributions centered at each training sample for a given class. In a typical problem, the PNN is trained using the results of the FMEA for the subject mechanical system. This effort typically results in a fault/symptom matrix in which only a single training vector is developed for each fault. In the case of only a single training pattern per class (i.e. m=1), the above equation simplifies to:

$$f_A(X) = \frac{1}{(2\pi)^{p/2}\sigma^p} \exp\left[-\frac{(X - X_A)^T (X - X_A)}{(2\sigma^2)}\right] \quad (2\text{-}6)$$

Conceptually, Equation 2-6 compares the input symptom vector to the training symptom vector for the fault class. The closer the match between the two, the larger the probability of that fault classification. Note that the fault probability can still be obtained even if one or more components of the input symptom vector *X* are unavailable or mismatched. In these cases, the resulting fault probabilities may be lower, but the method will still return a result.

Equation 2-6 is implemented in the pattern units of the PNN, as depicted in Figure 2-37.

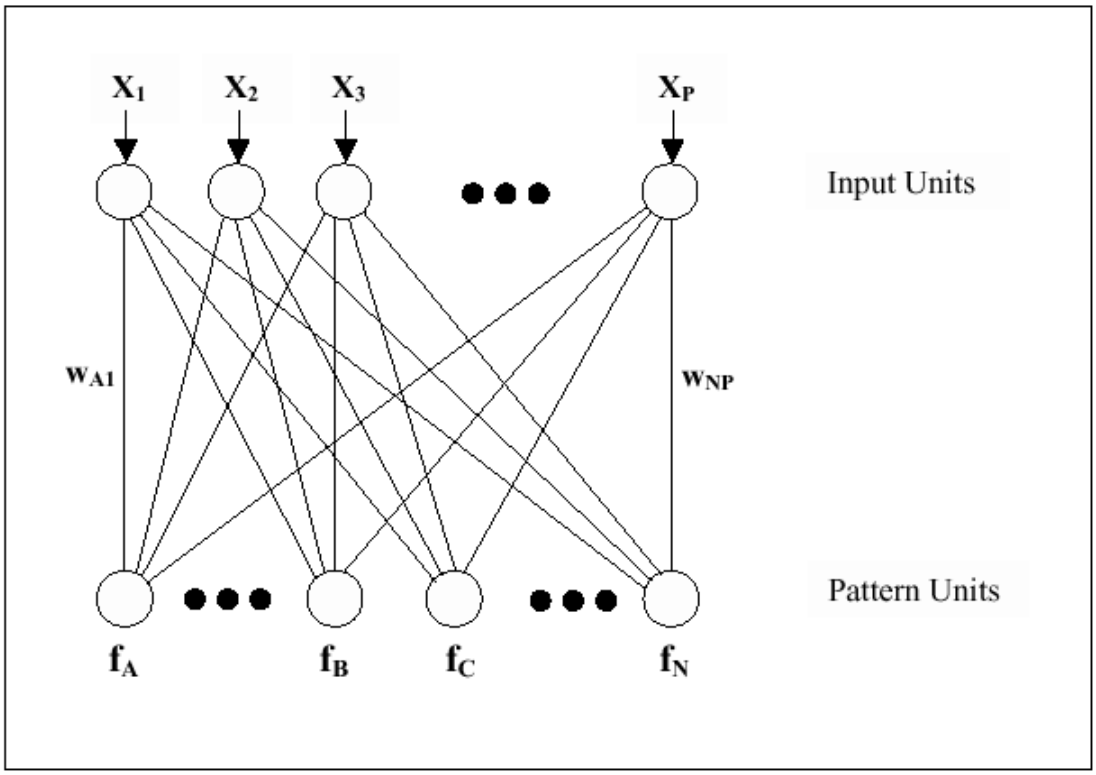

Figure 2-37. Network topology [50]

The network topology in Figure 2-37 differs from conventional neural nets in that the summation and output units are not used here, since there is only a single training example for each fault classification. The input units simply feed the input values to the pattern units. Each input unit has a connection with every pattern unit, and there is one pattern unit for each training pattern. The pattern units form the dot product of the input pattern vector, X, with a weight vector, $\mathbf{w_i}$, which is the training vector in this case. The dot product calculated in each pattern unit undergoes a nonlinear transformation in the PNN using an activation function similar to the form of the Gaussian PDF given in Equation 2-6.

The input vector *X* is comprised of the symptom pattern representing either current alarm conditions or predicted alarm conditions, depending on whether the system is performing a diagnostic or prognostic application. Quantitative alarm condition data are collapsed into categories. For this work, they are represented by a three-way classification as HIGH, LOW, or NORMAL states numerically encoded into the input vector. These classifications are performed by simple thresholding, as is done in most existing alarm monitoring systems.

Network training is accomplished by setting the weight vector of each pattern unit equal to the values of one of the training vectors. In this way, each training vector uniquely defines the weights of one pattern unit.

The only parameter adjusted in the PNN is the "smoothing" parameter σ, which is related to the variance of the underlying PDF. This parameter effectively controls the ability of the PNN to generalize when the input vectors do not exactly match the training vectors. Small values of σ result in poor generalization, causing the PDF to have distinct modes corresponding to the training sample positions in input space [86]. Larger values of σ produce greater degrees of generalization, with the PNN interpolating between training sample points. In this case, input vectors close to the training samples produce probability values close to that of the training points.

Logan *et al* [50] use this network for prognostic applications by performing a statistical regression analysis of each mechanical system parameter used in the network. The data points $\mathbf{x}_i$ from the sensor are used to create a regression equation (usually linear):

$$y = \beta_0 + \sum_i \beta_i x_i \tag{2-7}$$

where the βs represent the appropriate coefficients. Both raw measurements and time-based deviations from baseline conditions are analyzed over a pre-defined time interval. The length of this interval is determined by how much future warning is required for an actual alarm condition. The coefficients of trend equations are calculated from historical data within the pre-defined time interval and then tested at a 99% confidence level for statistical significance. If the coefficients are statistically significant, the trend equation is considered valid. Valid trend equations are then used for alarm prediction.

Each valid trend detected by the system is used to predict future alarm conditions within the mechanical system. The parameter associated with the trend is extrapolated out into the future using the estimated trend equation. If the predicted parameter value exceeds an alarm threshold within the pre-defined time interval, then the system inputs this alarm to the PNN-based inference engine. The PNN then uses its pattern recognition capabilities to predict plant fault conditions most closely associated with predicted alarms. The same PNN is used for both diagnostics and prognostics.

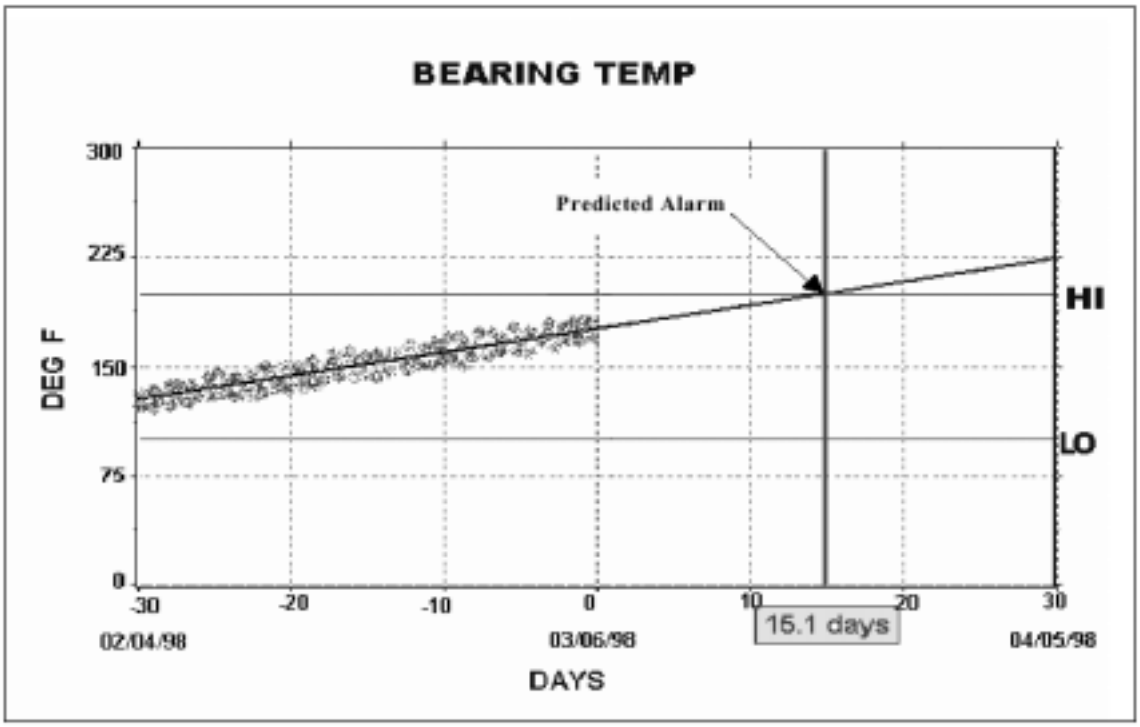

Figure 2-38. Example of a predicted bearing temperature alarm [50]

Figure 2-38 shows an example alarm prediction based on input data which were recorded for about a month. A trend is identified and modeled using linear regression. The regression line is projected out until an alarm threshold is encountered. If the trend continues over time, the bearing temperature will reach its HIGH threshold in approximately 15 days. A similar alarm prediction function is performed for all parameters having detected trends. For a prognostic application, the predicted HIGH bearing temperature alarm, along with other predicted alarms occurring in the same time frame, would be fed into the same diagnostic neural network to determine what system may be experiencing degraded performance.

Federici, *et al* [31] use a simulation model to determine problems in an electrical circuit. Their fault simulation process consists of simulating a circuit in the presence of faults, and comparing the results of fault simulation with the fault-free simulation of the same circuit with the same input test pattern. They propose the definition of a Behavioral Fault Simulation (BFS) technique which could be applied to VHDL (Very High Speed Integrated Circuit (VHSIC) Hardware Description Language) behavioral descriptions.

For clarity, VHDL is a large high-level VLSI design language with Ada-like syntax, and is the DoD standard for hardware description, now standardized as IEEE 1076. VLSI stands for Very Large Scale Integration and refers to semiconductor integrated circuits composed of hundreds of thousands of logic elements or memory cells [79].

The primary goal of the BFS as described in [31] is to determine the set of faults (belonging to the fault model) to be detected by a test pattern. A test pattern is a sequence of steps which are followed to test a circuit for faults. Different test patterns detect different faults. Their procedure submits faults from a global list to their simulator, in conjunction with the test pattern (shown as the test sequence). The aim of the test pattern generation process is to define patterns to test physical defects. The defects can be detected only if they induce an irregular behavior called a fault. The fault effect or error is measured by a difference between the state of the fault-free model (reference model) and the state of the faulty model (model in which a fault hypothesis is injected).

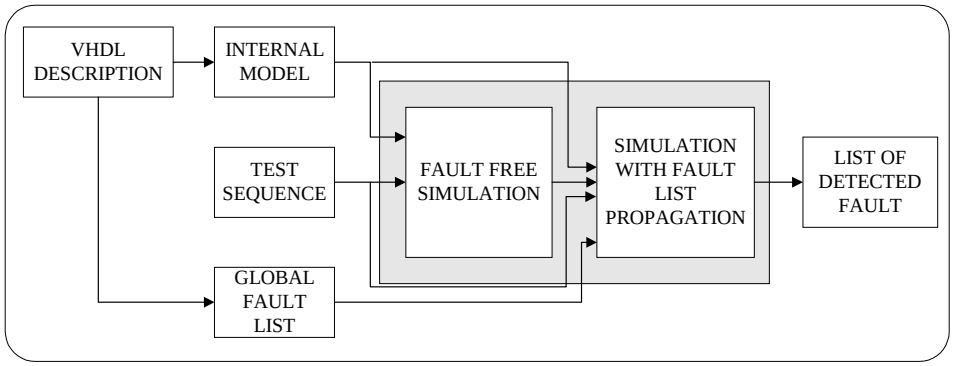

Figure 2-39. A schematic showing the experimental plan [31]

The experimental plan is shown in Figure 2-39. The test sequence process is the list of steps a test pattern takes to determine if a particular set of faults exist within the system. System defects can be detected only if they induce an irregular behavior, compared to normal functioning (found in a reference model called fault free simulation), which is then called a fault. The fault effect or error is measured by a difference between the state of the fault-free model (reference model) and the state of the faulty model (model in which a fault hypothesis is injected). All possible faults (from the global fault list) are systematically injected into the systems, and the specific test pattern is run to see if that particular fault is detected. The output is a list of faults the system actually detects. Ultimately, this simulation process could be used to evaluate and compare Behavioral Test Pattern Generation software via the different fault lists These lists would show the different faults each kind of test pattern would detect. Currently, this kind of capability does not exist [31].

Rebulanan [68] describes another simulation model. The focus of the simulation was on the PHM system, and the purpose was to assess an initial estimate of JSF supportability through the use of this system. The analysis compared the availability of four JSF aircraft with a PHM system with four JSF aircraft without a PHM system. The essential difference was that the PHM JSF aircraft provided a predicted component failure time before landing, while the aircraft without a PHM system did not. This reflects the expected difference between the two kinds of aircraft. A PHM equipped aircraft should provide fault reports before landing, providing additional lead time in the repair process. A non-PHM equipped aircraft will have to land and be inspected by a mechanic (the traditional/current diagnostic method) before any fault reports are available.

Relevant specifics of the simulation approach follow. The failure time of a particular aircraft component was assumed to be known, based on the Mean Time Between Failure (MTBF) measure associated with each component. The Mean Time To Repair (MTTR) was used to generate repair times. In the simulation, each time was generated from an associated probability distribution. The PHM system's detection of the impending component failure was assumed to be automatic and completely correct. The time the PHM model detects component failure was set to be 95% of the components useful lifespan. As an example, if a component's lifespan was 1000 minutes, the PHM system would automatically send a report at 950 minutes predicting this component's failure at 1000 minutes. A time to repair was also randomly generated from multiple single variable probability distributions based on multiple criteria. This criteria included the

component to be repaired (measured as probable in-stock availability of the component), transit time of the repair part to the flight line, and performing the actual repair.

As expected, the average availability of PHM-equipped aircraft is significantly higher than the availability of non-PHM equipped aircraft. A somewhat unexpected result was the higher variability in the availability rate of the PHM-equipped aircraft.

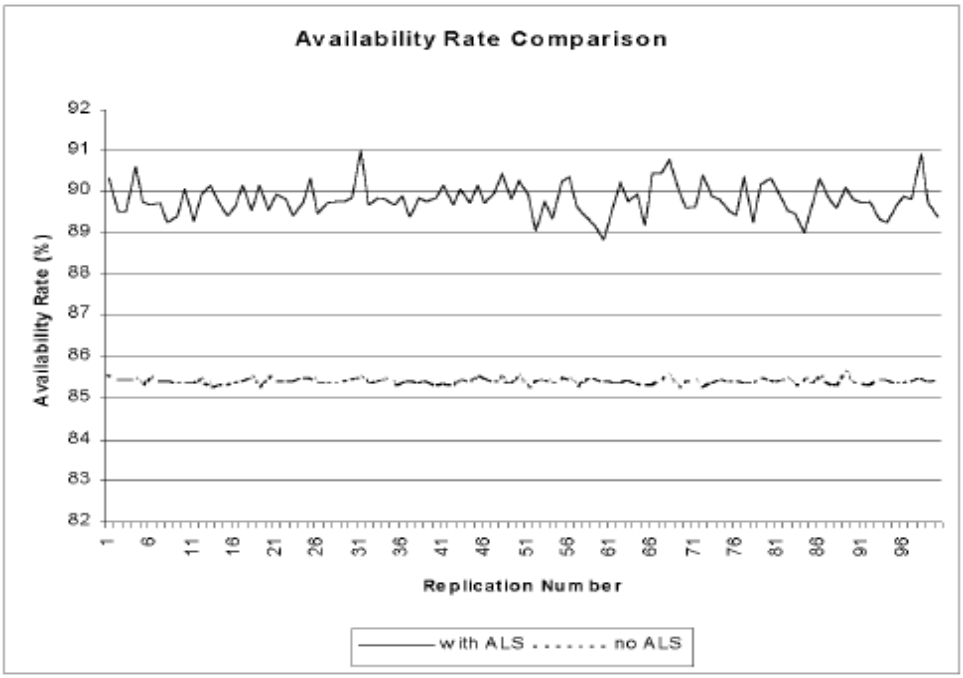

Figure 2-40. Comparison of Availability Rates between PHM (ALS) equipped aircraft and non-PHM (ALS) equipped aircraft [68]

Figure 2-40 shows that the availability rate varied between 89-91% for the PHM aircraft, while the rate was a practically constant 84% for the non-PHM equipped aircraft. Although Rebulanan [68] noted this variability existed for PHM aircraft, the variability

was not explained. However, this observed variability is likely due to the variability associated with the probability distribution used to determine the component failure time. The variability associated with the component failure time translated into variability associated with the prediction of the actual failure time on the PHM-equipped aircraft. This variance in the timing of the fault report, and consequent maintenance lead time, led to variance in overall aircraft availability. In contrast, the non-PHM equipped aircraft had no variability associated with maintenance lead time, since the aircraft had to land and be inspected before a fault report was generated. Based on Figure 2-40, it seems the time for this ground inspection was constant, although this is not explicitly stated in the paper.

Malley [53] followed Rebulanan's work on simulating an ALS system with a detailed computer model that simulated a PHM system. This PHM system model fit in the context of the previously developed ALS system. His simulation modeled the operations of one JSF wing and the activities of the corresponding support organizations for those aircraft. It used a neural network to analyze notional prognostic sensor signals to determine when an associated JSF system component (the engine, in his thesis) would fail. The simulation of these prognostic signals incorporated sensitivity to component wear-in, sensitivity to changing flight conditions, and a measure of variability as to when the component would begin exhibiting signs of failure. These measures were varied to produce different PHM signal sets. He found that averaging a number of these signals, or "batching" them, produced robust measures that a neural net could use to predict the JSF engine state with reasonable accuracy - about 82% of the time with his architecture.

These predictions of impending failures came when the engine was about 95% of the way through its expected life cycle, allowing enough time for the engine to be fixed before it failed in flight.

2.5 Literature Review Summary and Conclusions

Most published research concerning a prognostics effort is either concerned with a single component of a system (such as a rotor) or a single aspect of a system (such as startup data). Very few papers actually address the issue of what a complete prognostics system should contain. Most of those that do address these systems at a very high level. The literature apparently contains only one example of a complete prognostics system, Logan's DEXTER^{TM} system [50].

A fully developed prognostics system needs to be all-encompassing. It starts with the layout of the sensors within the system. This first step requires knowledge of the appropriate location for each sensor, the type (acoustic, electrical, etc.) of each sensor that should be used at a given location, and the total number of sensors that should be used (to avoid too little or too much information). Then, the data from these sensors needs to be captured and processed. Afterwards, the processed data is fed to a intelligent reasoner of some kind which interprets the data input and provides a system health assessment. This assessment may include a confidence level. Then, this assessment is reported to appropriate entities. These may include system operators, system mechanics, and system operations planners.

The prognostics reasoning capability is best described as the capability of a PHM system to extrapolate from current data streams to predict when a certain portion of the system is expected to fail. Of course, the biggest reason to monitor a system using PHM technology is to detect an impending component failure in time to prevent a system failure by replacing the affected component before it actually fails. Rather than repairing or replacing a component after it has failed, it can be repaired or replaced when the prognostics system indicates that probable time to failure (or probability of component failure) is within some critical threshold. The question is what is required for this impending component failure to be detected.

The nature of the prognostics reasoning problem is a difficult one. Experts in this field identified reliable estimation of time-to-failure as one of the greatest challenges in manufacturing and machine monitoring, and one of weakest areas in existing methods [82]. Furthermore, these experts state that methods used to analyze the data from mechanical system processes must be robust, i.e., methods which can tolerate significant deviations from assumed or nominal signal characteristics. In general, the signal and noise environment in these kinds of applications is highly complex, non-Gaussian, and exhibits large variability and/or non-stationarity. The operating conditions may vary dramatically between sensor locations. To ensure the user accepts these monitoring methods, low false alarm rates are an absolute necessity. This places an additional burden on the robustness of the methods. A successful prognostics system implementation must address all these issues.

The first requirement for a prognostics reasoning system is on-board sensors which record the performance of aircraft systems. This requirement in and of itself is a significant issue. The total number of sensors required for producing a prognostics capability is an open question. If there are too few sensors, not enough data will be collected for analysis and prognostic functions. If there are too many sensors, the prognostic system may be overcome by so much variation from the sensor reports that it fails to recognize any impending failures at all. The variation in the readings may also be coming from failed sensors, as opposed to systems which are actually failing. The recorded data may also exceed the capability of the system bus to report it, so data is lost before it is ever recorded. However, with modern technology, this "data overflow" issue is becoming less of a concern.

The next issue under this first requirement is which systems the sensors are attached to. There are a tremendous number of systems present on a modern aircraft (somewhere in the hundreds). Should all these systems be monitored, or just some of them? If only some systems should be monitored, which ones should they be? And given those systems, what kinds of measurements should be taken (acoustic, electrical, vibration, etc.)? These questions need to be answered to determine the proper scope of the data for a prognostics reasoning system.

Once the sensors are in place for aircraft data collection, the actual collected data will require pre-processing before submission to the prognostic system. Raw sensor data is typically very noisy, and key features describing the performance of the monitored

system are not readily apparent. The concern here is which kinds of processing techniques should be employed. By its very nature, data pre-processing modifies some of the collected data (hopefully removing the noise) while enhancing the rest of the data (the signal of interest). However, since many pre-processing techniques are well known and their effects are understood, this is not as significant of an issue as are other issues.

The biggest issue for a prognostics reasoning system is the interpretation of the collected and pre-processed data. In order to assess the health of an aircraft based on this information, it must be compared to previously existing information which has been classified as either representative of a healthy system, a degraded system, or a failing system. In order for this comparison to be done, this "previously existing" data must be collected from similar (if not exactly the same) systems that are operating in a known state.

At this point, a few words are in order about the presumed nature of general mechanical system faults. Most faults are believed to begin with small (but detectable) precursor events and to stem from a progressive (not necessarily linear) degradation of the system component. The degradation curve is usually assumed to follow some kind of exponential relationship [82], although some naval applications show a linear trend [51]. Thus, the tracking of this degradation along with an ongoing prediction of the time-to-failure is of great importance to a prognostics system. Additionally, as previously stated [60], the signal and noise environment in these kinds of applications is highly variable and complex. Also, the signal characteristics from many types of degradations are non-

monotonic. Consequently, an understanding of the overall trend, as well as continuous monitoring to track the history of the developing fault, is essential [82]. Faults that are neglected are those which develop rapidly without any forewarning (such as the effects of combat). Clearly, no prognostics system can predict rapidly developing events which occur completely within a time window that is considerably less than a single operational cycle.

In order to make sense of this data, there must be a reasoning function in the PHM system. This reasoning function is required to identify normal behavior and system faults with high confidence. To accomplish this, there must be patterns present within the reasoning function which represent functional and failed behavior. The reasoning function for a PHM system is also expected to predict when component failure will occur. This requires clear patterns of how system faults develop. How these patterns can be captured is addressed below.

The patterns for a functional state are thought to be the easiest to collect. Once an expert (probably human) has assessed the system as working correctly, the data from the system are fed to the reasoning system, which encodes the data as representative of a functional state. Should there be more than one functional state, conditions in which these functional states exist can also be replicated and encoded within the reasoner. As the system operates, comparisons between this part of the prognostic reasoner and the system data should clearly indicate whether the system is in a functional state or not. This is one

way a PHM system can provide an instantaneous (simple yes/no) assessment of system health.

The collection of failure patterns is a somewhat more difficult problem. When systems are in a failure state, by definition they are not operating. This may prevent the collection of certain kinds of system data. To overcome this, outside expertise is required to supplement the data patterns recorded when a system is in a failure state. Additionally, it is difficult to record every conceivable failure state *a priori*. The prognostic reasoner must be able to accept new failure states as they appear during the operation of the system. Using this data, the reasoner can provide instantaneous estimates of system failure by comparing it to known functional and failed states, if the failure status is not readily observable.

Collecting patterns of how system faults develop is difficult, but essential in order for a PHM system to accurately predict when a failure will occur. For this predictive capability to be developed, there must be a well-defined path (henceforth called a "failure path") from current operational conditions to the many fault conditions, and all variations along these failure paths must be understandable and detectable. Collecting the data to meet this requirement is the most difficult technical challenge of these three. Mechanical systems undergo preventive maintenance to avoid failures, which interrupt the collection of data along fault paths. Actually operating a functional system to observe the failure path of a single component can result in ruining the entire system. Re-running the same experiment to note any variations in the failure path of the same component will double

the costs. And, as previously discussed, the data along these paths is highly complex, non-Gaussian, and exhibits large variability and/or non-stationarity.

To overcome these problems and collect the required data, most failure paths are mapped based on performance of an individual component on a test bench. There are two potential problems with this approach. The first potential problem here is that the individual component is being assessed independently of the overall system; interactions are not captured. And secondly, most components are very durable, and take a very long time to fail when subject to normal operational stresses. To save experimental time and cost, components are overloaded with operational stresses that are multiples of the normal values. The resulting failure path may not represent what really happens to the component for this specific type of failure. It also may mask other failures that would normally occur before the specific type of failure under consideration.

Another way to obtain failure information from system data is to use the known failure points of the system components, and not use any computed failure path patterns at all. These failure points may consist either of the time which a particular component is expected to last, or component readings at failure.

If only the time that a particular system component is expected to last is being used to compute a possible failure point, then the system simply keeps track of the amount of operational time a component has been in use. This is compared to the distribution of failure times for this component. When an appropriate threshold is reached, the system

indicates it is time to replace the component. This threshold may be expressed as the point at which a certain percentage of the components have failed, or how long it will be until failure is virtually certain. The potential problem here is that all the aircraft components are usually manufactured at the same time. The initial failure time distribution becomes less and less representative of the actual population as these components age. In the process of maintenance, some components are refurbished with new units, so averaging their performance together with the unrefurbished units leads to a distribution that is not really representative of either population. However, the Air Force is tracking some of its electronic components by barcode. There could be two failure distributions; one for refurbished units, and one for the others. Although this does require a lot of bookkeeping, tracking the different maintenance actions by electronic unit has been shown to be feasible.

If the component readings at failure are being used, trend analysis is applied to the data being collected from these system components. If the PHM system detects a "definite" trend towards a failure point, this would be reported as negative system health. A projection along this "definite" trend will give an assessment of how long it will be before the component fails. The advantage of this approach is that failure path generation is not required. Disadvantages include the need to know precisely what a component's failure point is. Projecting the "definite" trend is also a disadvantage since it requires extrapolation beyond the original data set. As an example, what may have been originally thought to be a linear trend may turn into an exponential trend, leading to failure much sooner than anticipated. The reverse situation also leads to problems, as

maintenance action is scheduled sooner than required, leading to the replacement of a component with remaining usable life.

Assuming that patterns for the functional state, the failure paths, and the failure states all exist within the prognostic reasoning system, assessments can be made of instantaneous system health and time to component failure. As previously mentioned, comparisons between the system data and the functional patterns present within the prognostic reasoner can give a simple yes/no indication of system health. Another way is to compare current system readings, or operational time deployed, with known failure points for these systems. This information can provide a simple yes/no assessment of system health as well, if the proximity of the sensor reading is "close" to the known failure reading. (The same holds true for comparing time deployed to the time-to-failure distribution.) This information can also provide a probability assessment of impending failure. The third way is to compare trends (or current values) in the system data with the previously defined failure paths. The data of any component that doesn't indicate normal operation can be mapped to the failure path. This provides an instantaneous (negative) health assessment. It also provides an estimate of time remaining to failure, based on the distance remaining on the failure path. Of course, this assumes the failure path and/or fault condition is known for the specific event. If not, the PHM system will only be able to provide a (negative) assessment of system health (what the PHM system is seeing doesn't match the data for normal operation).

III. Data Fusion Methods

3.1 Background

Multi-sensor data fusion is a field that has experienced rapid growth comparatively recently. The problem of merging similar (or disparate) information from multiple sources has grown in importance as the number of information sources available to the decision maker has significantly increased in the past 20 years. In past years, decision makers would assess written or verbal reports, with or sometimes without certain levels of confidence, and decide on a course of action based on their internal "fusion" of the information. As computer power has increased through the years, the automated computation of the "best" estimate of what all these sensors say has become more and more possible. The number of methods used to assimilate the data into a unified assessment of a given situation has also increased greatly in recent years. Arguably, it is no longer humanly possible to correlate all the data streams available to provide the best interpretation of the data, without computational assistance.

Data fusion is required because of data fission. The total signature of an entity is usually manifested in many separated types. Since most sensors only collect one type of information, the complete entity signature can only be reconstructed through fusing these collected types to reconstruct the original entity. The information decomposition can be attributed to different types of phenomena. These include different characteristics under consideration, such as shape or motion; detection of different information types, such as electromagnetic or acoustic radiation; detection of different parts of the frequency

spectrum, such as electrical current or infrared data; restricted spatial or temporal coverage; and an historical legacy of separate processing systems. Rarely does one sensor embody more than one collection technique. Consequently, a single sensing mechanism is unlikely to be capable of capturing all the desired information on an entity at a given instant of time. Data fusion brings this information back together to provide the picture of the original entity.

The methods of data fusion depend on the situation. There may be several similar sensors providing information on the same entities. In this case, the sensors detect the same features on the entities, yielding what is termed competitive data. The overlapping features of the data must be correctly merged to identify the data sources. The other case occurs when different types of sensors collect different features on the same entities, yielding complementary data. In this case, the data between the different sensors does not overlap. In both cases, however, a single sensor usually collects data on more than one entity, so the data is almost always dependent.

Data fusion techniques are also dependent on the type of data present. The preceding paragraph discusses a situation in which signal processing techniques would be quite helpful (signal filtering, spectral analysis, time-domain fusion). To estimate the state of a given system, Kalman filters or some other kind of Bayesian reasoning may be most appropriate. If there is more background knowledge available, then what may be called a "cognitive technique" can be used. These techniques can include neural nets, clustering/genetic algorithms, or fuzzy logic. If expert knowledge can help determine the

exact state of affairs, expert systems or case-based reasoning may be applicable. However, there is no one “golden method” which applies in all situations. Most problems will require a combination of the above techniques to provide an accurate solution. In the example of the preceding paragraph, a combination of an expert system (previously existing signatures) could be combined with time-domain fusion to provide a fused picture of the environment.

Of course, the methods chosen to fuse the data also depend on the kind of data available. For most military applications, the data comes from multiple sensors collecting information throughout the electromagnetic spectrum, as well as audio, motion, and vibration detectors. This includes sensor location and at times, a level of confidence in the collection. However, sensor reliability, previously analyzed data, large databases, expert systems, and other types of pre-existing information are also candidates for data fusion. The degree to which each data stream is weighted compared to the other streams is of central importance. Of course, data fusion can never totally recover the loss introduced by the original data fission.

There are varying definitions of what constitutes multi-sensor data fusion, but these definitions differ primarily only in technical details. For example, the International Society of Information Fusion defines it as follows [25]: “Information Fusion, in the context of its use by the Society, encompasses the theory, techniques and tools conceived and employed for exploiting the synergy in the information acquired from multiple sources (sensor, databases, information gathered by human, etc.) such that the resulting

decision or action is in some sense better than (qualitatively or quantitatively, in terms of accuracy, robustness and etc.(sic)) than would be possible if any of these sources were used individually without such synergy exploitation." The USAF Research Lab [21] defines it as: "Information Fusion: Events, activities and movements will be correlated and analyzed as they occur in time and space, to determine the location, identity and status of individual objects (equipment and units), to assess the situation, to qualitatively and quantitatively determine threats and to detect patterns in activity that reveal intent or capability. Specific technologies are required to refine, direct and manage the information fusion capabilities." In essence, data fusion is the management (and consequent minimization) of uncertainty associated with the input data. The goal is to obtain the best assessment of the system under consideration with a minimal amount of uncertainty.

The use of the data in data fusion has widely varying adherents throughout the community. There are those who advocate a "sensor to shooter" data fusion architecture. The raw data from the sensor is sent directly to the warfighters who put ordnance on the target. Unfortunately, with the tremendous amount of data being collected on the modern battlefield, the warfighter cannot hope to keep up with the flow of information. And that is ignoring the issue of contradictory and/or simply incorrect sensor reports. As some leaders in this community have said, the warfighter is awash in information but starved for knowledge. What a sensor report means in the context of other sensor reports is far more valuable than an individual report standing alone.

The data in data fusion are useless unless they are placed in context, then the data may be considered information. Knowing what the data indicates and the associated level of confidence are essential. In turn, when this information is placed in its proper context, it may be considered knowledge. An indicator from a ships' radar of tank activity would be expected if the ship was close to shore, but perhaps not if the ship was in the middle of the ocean The knowledge of what the sensor indicates and whether that is reasonable given current surroundings is also important. This idea can extended to knowledge of multiple activities, which could be called understanding. Perceiving what purpose underlies the knowledge of the enemies' activities is yet another level of fusion. However, interpretation of purpose exceeds current computational capabilities.

3.2 Neural Network Methods

The term "artificial neural network" (ANN) refers to a wide range of analog computational schemes that are loosely based on biological nervous systems. These schemes are generally built to classify an unknown object into a particular class of objects based on observations (input data) obtained from that object. Neural nets can also be used to classify a system's operation into one of a number of operational modes (e.g., running efficiently, nearing failure, non-operational, etc.) based on data obtained from system components.

A typical ANN consists of a web of interconnected simple mathematical processors called "neurons" or "units" or "nodes". Three components are required to describe a network:

1. The neural units, the number of layers in the network, and their "activation" functions.

2. The connections between units, known as the neural architecture.

3. A training algorithm to develop the most appropriate weights for connections between units.

The following section describes each of these three components in turn.

3.2.1 Neural Units

A single-layer neural net (also known as a "perceptron") looks like the following figure.

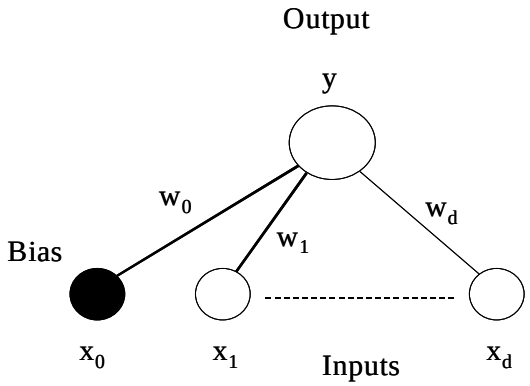

Figure 3-1. Single-layer neural net [15]

Figure 3-1 shows a single layer neural net. This architecture is also known as a "perceptron." The bias node is a constant value specified by the user. The inputs are weighted to give an output. The net is trained on known data so the weights on each branch are the best for classifying that particular data set (training will be addressed in more detail later). The initial set of weights is usually chosen randomly.

A multi-layer neural net schematically looks like the following figure.

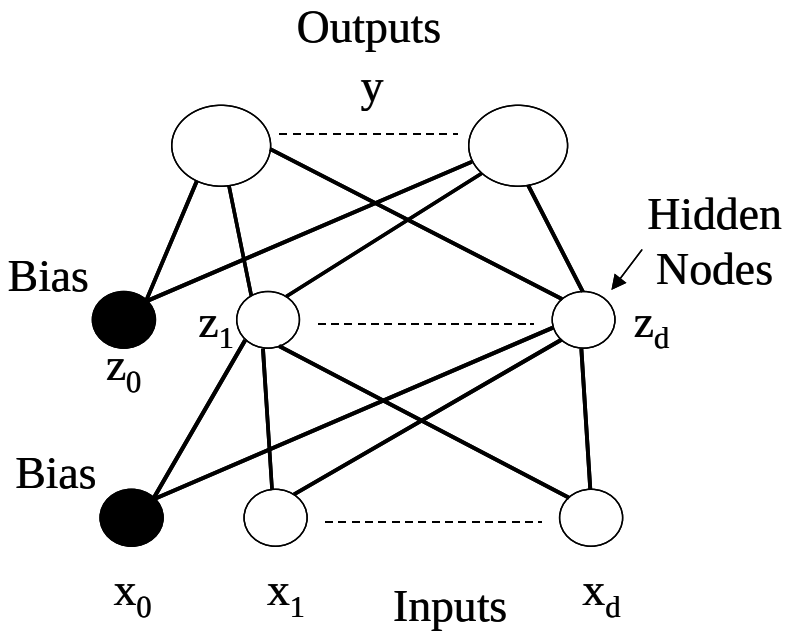

Figure 3-2. Multi-layer neural net [15]

Figure 3-2 shows the input layer, hidden layer, and output layer of a typical multi-layer neural net. This type of architecture is also known as a multi-layer perceptron neural net. There are many more weights in this type of architecture. Again, a set of data where the actual outcome is known for each set of input data is used to train the network.

In the type of ANN considered here (multi-layer perceptron), the neural net node takes the weighted sum of its inputs and feeds that value into an activation function (which is typically nonlinear. The activation function transforms the weighted input from other nodes into a new value.

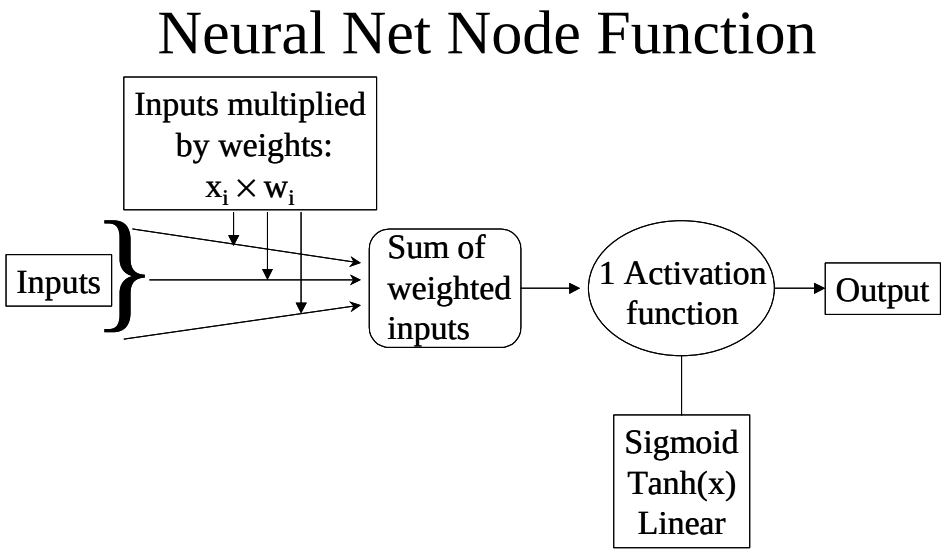

Figure 3-3. Activity performed in a typical neural network node

Figure 3-3 shows the usual function of a neural network node. Usually, there are many inputs into a single node. Each input is multiplied by a weight. Then, the resulting products are added to form a single sum. This sum is then input into the activation function. The result is computed and sent forward as the output of that particular node. The output may also be sent to many nodes.

An activation function commonly used in these kinds of neural nets is the sigmoid (logistic) function:

$$f(x) = \frac{1}{1+e^{-x}} \tag{3-1}$$

Other commonly used activation functions include the hyperbolic tangent (tanh(x)) function. In some cases, researchers also use units with linear activation functions. Linear activation functions are most commonly used in the output layer of the network.

3.2.2 Network Connections

Nodes (represented below by circles) are connected to propagate a signal from the inputs to the outputs of the net.

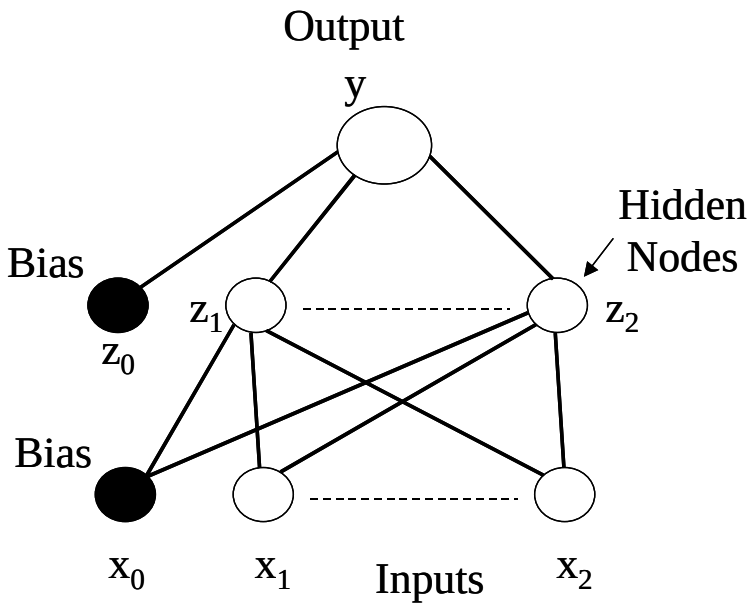

Figure 3-4. Single output neural net [15]

The network shown in Figure 3-4 could be used to approximate a function of two variables, Y=f(X1,X2). The input values (X1, X2) are appropriately weighted and fed into the nodes above them. Subsequent units compute their values according to the

weighted connections and activation functions. The answer, Y, is read from the unit in the output area.

ANNs are often partitioned into distinct sets of related neural units, called “layers” or “areas”. For example, all of the units used as inputs to the unit constitute the “input layer”; likewise units used as outputs make up the “output layer”. All other units are organized into one or more “hidden layers”. The resulting arrangement of nodes and connections in a network is known as the network topology.

Layers are connected by groups of lines (loosely, the “nerves”) called projections. A non-zero weight is usually assigned to each projection. For ANNs, units in a particular layer are usually connected to every other unit in each adjacent layer. A notable exception is what is termed the “bias node” or “bias unit”. The weight attached to this value is usually set at 1, and the negative of this value is usually known as the “threshold”.

Many neural networks have the structure given in Figure 3-2 with an additional hidden layer. This is because of a theoretical result which states that a neural network with three layers of weights can produce an arbitrarily complex decision boundary [90]. In other words, it can correctly classify objects no matter how tightly they may be grouped together in real life. Unfortunately, the theorem only states that the network exists-finding it is another matter altogether. In a similar result, a network with two layers of weights (just like Figure 3-2) and sigmoid activation functions can approximate any

decision boundary to arbitrary accuracy. So using sigmoid activation functions allows the use of a smaller network, but with the same guarantee that the perfect neural net for a particular problem exists [15], [40], [90]. Again, finding that neural net is another issue altogether. That issue is partly addressed by how the network is trained, which leads into the next section.

3.2.3 Training Neural Networks

Making a network perform useful work, e.g. correctly classifying a large number of unknown entities, involves finding good values for the weights of the connections between units. While commonly referred to as "training", this is basically an optimization problem, and has been addressed in several different ways:

> Local methods, such as backpropagation and its many variants. These methods focus on a small area of the solution space at a time.
>
> Global gradient-based methods, e.g. conjugate gradient, Levenberg-Marquardt. These methods focus on a larger area of the solution space.
>
> Stochastic methods, e.g. genetic algorithms, simulated annealing. These methods use some form of a random process to generate better and better weights.

These training methods in general involve an iterative procedure for minimization of an error function, with the weights being adjusted in a sequence of steps [15].

3.2.4 Different Neural Network Methods

There are many different implementations of the neural network architecture in the literature. There are two major classes of neural network models. The first uses nodes (units) which compute a non-linear function (usually sigmoid) of the product of an input vector and a weight vector. The main example of this technique is the multi-layer perceptron. The other class of neural networks uses the distance between the input vector and another generalized vector (usually the average of the input vectors) for the computation at the node (unit). Radial basis function neural networks and probabilistic neural networks are examples of this latter type. The following list briefly summarizes some of these network methodologies with are considered to be suitable for automated machine learning [73].

The multiplayer perceptron with backpropagation learning is probably the most commonly applied ANN model [74]. When a neural net is being trained, input data and the associated desired network output values (called targets) are presented to the network. The backpropagation algorithm, in general, feeds the error (distance from the target) associated with a particular input vector back through the network. The out put layer computes its error, and feeds this back to the previous layer, which computes its error, and feeds back its error, until the first layer in network has computed its error. Once each individual neuron has computed its error, it estimates a change for the weight vector that would reduce its error. This change is typically multiplied by a learning rate which is significantly less than one (usually 0.1). The learning rate reduces the amount of change

to produce a neural network that can classify many similar inputs well, instead of one input perfectly.

The functional link neural network (FLNN) performs least squared error learning like that of a backpropagation neural net, but no learning takes place in the hidden layer. Instead, the hidden layer combines the inputs using various nonlinear functions [45].

The probabilistic neural net (PNN) is an ANN implementation of the Parzen windows method. The output is a weighted sum of all training points, where the weighting is exponential according to the distance of an unclassified input from a given training point [85], [86]. The general regression neural network (GRNN) is the PNN augmented by a normalizing factor [84].

Radial basis function neural networks (RBFNN) contain a set of uniformly distributed processing units each with a radially symmetric response. During training, the algorithm adjusts the amplitude of the response to estimate the function [69].

Radial basis functions (RBF) are used as activation functions in this second class of neural networks.

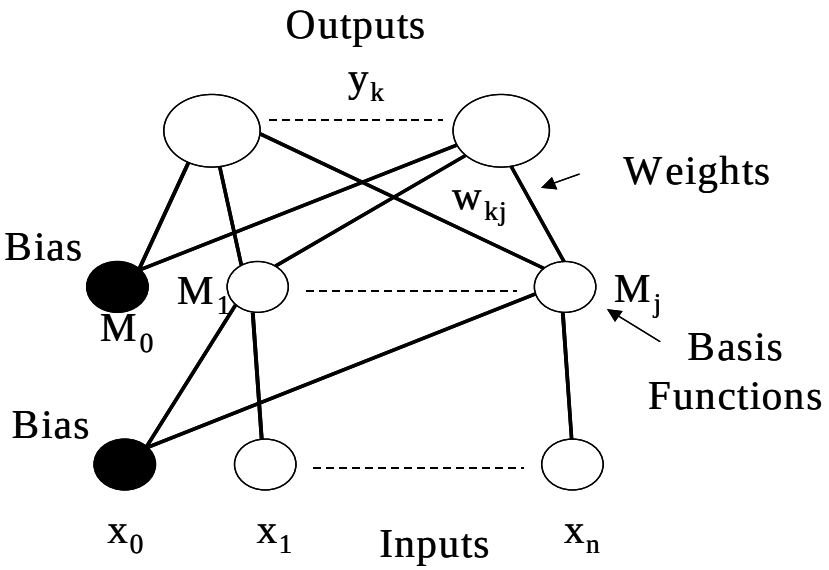

Figure 3-5. Typical RBF network [15]

Figure 3-5 shows a typical RBF network. There are N inputs, indexed 1 to n, which are combined with M basis functions (M =N in almost all cases), indexed 1 to j. There are K output nodes, indexed from 1 to k.

The general problem radial basis function neural networks are used to solve is the mapping from a d-dimensional input space **x** to a one-dimensional target space **t**. The input data consists of N input vectors x_n, and corresponding targets t_n. The object is to find a function h(x) such that $h(x_n) = t_n$, for n = 1 to N [65]. The radial basis function approach [65] assigns a basis function to each of the N data points. The basis function has the form $\phi(|\mathbf{x} - x_n|)$, where ϕ is usually Gaussian, the distance function $|\mathbf{x} - x_n|$ is usually Euclidean, and **x** is usually either the average of the input vectors or the center of the assigned basis function. The output of the mapping is a linear combination of all M basis functions (at present, M =N):

$$h(x) = \sum w_n \exp([-1/2\sigma_n^2]*|\mathbf{x} - x_n|^2) \quad (3\text{-}2)$$

The weights w_n are found via a two-stage process [15]. In the first stage, the input data set is used to determine the parameters of the basis functions (μ and σ if the function is Gaussian). The basis functions are then kept fixed while the second layer weights are found in the second training phase. Mathematically, if the radial basis function is written as:

$$y_k(x) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) \quad (3\text{-}3)$$

then the matrix representation is:

$$\mathbf{y(x)} = \mathbf{W}\boldsymbol{\phi} \quad (3\text{-}4)$$

where $\mathbf{W} = w_{kj}$ and $\boldsymbol{\phi} = \phi_j$. The error function is a sum of squares expression:

$$E = .5\sum_n \sum_k \{y_k(x_n) - t_{nk}\}^2 \quad (3\text{-}5)$$

where t_{nk} is the target value for output unit k, corresponding to the input vector x_n. The weights are found from a set of linear equations

$$\mathbf{\Phi^T \Phi W^T} = \mathbf{\Phi^T T} \quad (3\text{-}6)$$

where $(T)_{nk} = t_{nk}$ and $(\mathbf{\Phi})_{nj} = \phi_j(x_n)$. The formal solution is given by:

$$\mathbf{W^T} = \mathbf{\Phi^* T} \quad (3\text{-}7)$$

where the $\mathbf{\Phi^*}$ notation denotes the pseudo-inverse of $\mathbf{\Phi}$. In practice, the equations given above are solved using singular value decomposition to avoid problems associated with the possible ill-conditioning of the matrix $\mathbf{\Phi}$.

Typically, for radial basis function neural network, the number of basis functions is much less than the number of data points [15]. In general, the radial basis function neural networks learns quicker than multi-layer perceptron neural networks. The trade-off is that the multi-layer perceptron neural networks exhibit improved generalization properties, especially for regions not sufficiently represented in the data set [47]. To obtain this improved generalization, an RBF network has to have more functions to better characterize the input space [39]. The number of functions exhibits a direct exponential dependence on the dimension of the input space. The benefit of using radial basis function networks is the property of best approximation: the function with minimum approximating error is in the set of approximating functions this network may adopt [39]. Girosi and Poggio [39] also showed that the multi-layer perceptron does not share this property.

Similar to the an RBFNN, the k-nearest radial basis function network (KNRBF) learns like the RBFNN. Its output is computed the same way, except only the k nearest basis functions are used in the exponentially weighted sum [73].

The dynamic radial basis function neural network with locally tuned units (LTRBFNN) uses a clustering technique on the input data to determine optimal placement of its non-symmetric basis functions. Then, it uses heuristics to determine the widths of the basis functions. On a second pass through the data, it uses least mean squares to determine the amplitude of the basis functions [59].

The dynamically stable learning neural network (DYSTALNN) was derived from the actual wiring of a simple invertebrate nervous system and the details of mammalian learning at a cellular level. A DYSTALNN maps an input vector to the processing unit that stores a cluster center vector that matches the input best. The output is the product of the measure of similarity and the output vector stored at the processing unit. This architecture adds new processing units whenever it encounters an input significantly different from any previous inputs [3].

The restricted coulomb energy neural network (RCENN) allocates regions to some training inputs. RCE allocates the first input to a large region, but ignores subsequent inputs that fall inside that region unless they are associated with a different output value. In this occurs, the RCENN divides the previous region and allocates a portion to the new input. The training technique requires several passes through the training data to ensure that all training data falls inside some allocated region. When the network is trained, input vectors (with unknown targets) will fall into some region with a training input at its center. The output is what was pre-defined for that region during training [70].

The cerebellar model articulation controller (CMAC) was inspired by the architecture of the mammalian cerebellum [2]. This architecture maps input values to a particular bin, represented by a fixed integer value. The minimum value in the input range maps to 0, and the maximum value maps to the bin associated with the largest value. The number of bins used for the mapping depends heavily on the application. For training, all entries in the bins are initialized to 0. When a bin encounters a training input, the bin value of 0 is

replaced by the desired output. If a bin does not encounter a training value, the value remains 0. If the bin encounters multiple different outputs, on the last output recorded is retained. Various generalization algorithms are used to compensate for this. The chief advantage of this technique is that the error surface has a unique minimum that is "down the slope" from every other point on the curve, and that learning process converges to this unique value fairly rapidly. The technique is not susceptible to local minima in the error surface, unlike other neural network architectures.

3.2.5 Combining Neural Networks

Opitz and Maclin [61] discusses the comparison of 2 different data fusion techniques, known as Bagging and Boosting. These techniques combine the predictions of multiple classifiers to produce a single classifier. The resulting classifier, which is referred to as an ensemble, is generally more accurate than any of the individual classifiers making up the ensemble. Theoretical and empirical research has demonstrated that a good ensemble is one where the individual classifiers in the ensemble are both accurate and make their errors on different parts of the input space. The Bragging and Boosting methods rely on resampling techniques to obtain different training sets for each of the classifiers.

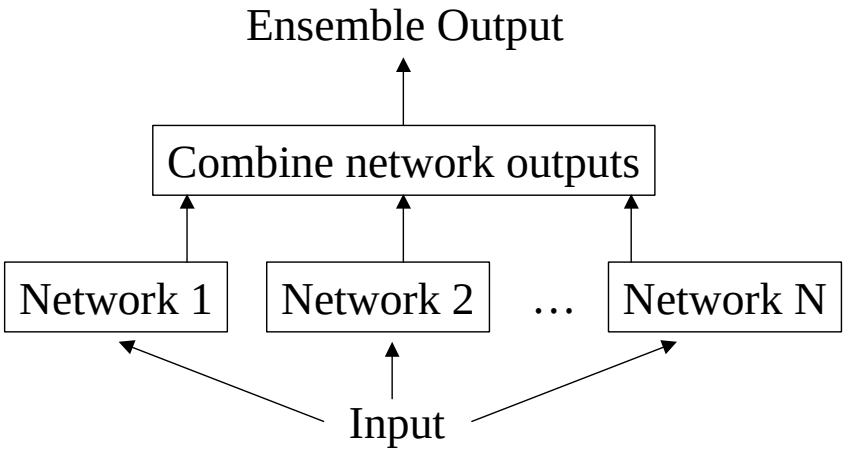

Figure 3-6. A classifier ensemble of neural networks. [61]

Figure 3-6 illustrates the basic framework for a classifier ensemble. In this example, neural networks are the basic classification method, though conceptually any classification method (such as decision trees) can be substituted in place of the networks. Each network in the figure's ensemble is trained using the training instances for that network. Then, for each example, the predicted output of each of these networks is combined to produce the output of the ensemble. The consensus among many researchers [61] is that an effective combining scheme is to simply average the predictions of the ensemble.

Of course, combining the output of several classifiers is useful only if there is a reasonable amount of disagreement among them. Obviously, combining several identical classifiers produces no gain. Hansen and Salamon [38] proved that if the average error rate for an ensemble is less than 50% and the component classifiers in the ensemble are independent in the production of their errors, the expected error for that example can be

reduced to zero as the number of classifiers combined goes to infinity. However, such assumptions rarely hold in practice. Krogh and Vedelsby [46] later proved that the ensemble error can be divided into a term measuring the average generalization error of each individual classifier and a term measuring the disagreement among the classifiers. They formally showed that an ideal ensemble consists of highly correct classifiers that disagree as much as possible. Other researchers [61] have empirically verified that such ensembles generalize well.

As a result, methods for creating ensembles center around producing classifiers that disagree on their predictions. Generally, these methods focus on altering the training process in the hope that the resulting classifiers will produce different predictions. For example, neural network techniques that have been employed include methods for training with different topologies, different initial weights, different parameters, and training only on a portion of the training set. The remainder of [61] focuses on two methods (Bagging and Boosting) that try to generate disagreement among the classifiers by altering the training set each classifier sees.

Bagging is a bootstrap ensemble method that creates individuals for its ensemble by training each classifier on a random redistribution of the training set. Each classifier's training set is generated by randomly drawing, with replacement, N examples, where N is the size of the original training set. Many of the original examples may be repeated in the resulting training set while others may be left out. Each individual classifier in the ensemble is generated with a different random sampling of the training set.

Boosting encompasses a family of methods. The focus of these methods is to produce a series of classifiers. The training set used for each member of the series is chosen based on the performance of the earlier classifier(s) in the series. In Boosting, examples that are incorrectly predicted by previous classifiers in the series are chosen more often than examples that were correctly predicted. Thus Boosting attempts to produce new classifiers that are better able to predict examples for which the current ensemble's performance is poor. (Note that in Bagging, the resampling of the training set is not dependent on the performance of the earlier classifiers.)

In [61], the authors also examine two new forms of Boosting: Arcing and Ada-Boosting. Like Bagging, Arcing chooses a training set of size N for classifier number K+1 by probabilistically selecting (with replacement) examples from the original N training examples. Unlike Bagging, the probability of selecting an example is not equal across the training set. This probability depends on how often that example was misclassified by the previous K classifiers. Ada-Boosting uses either the approach of (a) selecting a set of examples based on the probabilities of the examples, or (b) simply using all of the examples and weight the error of each example by the probability for that example (i.e., examples with higher probabilities have more effect on the error). This latter approach has the clear advantage that each example is incorporated (at least in part) in the training set. This form of Ada-Boosting can be viewed as a form of additive modeling for optimizing a logistic loss function. In this paper, the authors have chosen to use the approach of subsampling the data to ensure a fair empirical comparison (in part due to the restarting reason discussed below).

Both Arcing and Ada-Boosting initially set the probability of picking each example to be 1/N. These methods then recalculate these probabilities after each trained classifier is added to the ensemble. For Ada-Boosting, E_k is the sum of the probabilities of the misclassified instances for the currently trained classifier C_k. The probabilities for the next trial are generated by multiplying the probabilities of C_k 's incorrectly classified instances by the factor $B_k = (1 - E_k)/E_k$ and then renormalizing all probabilities so that their sum equals 1. Ada-Boosting combines the classifiers C_1, ... ,C_k using weighted voting where C_k has weight $\log(B_k)$. These weights allow Ada-Boosting to discount the predictions of classifiers that are not very accurate on the overall problem.

In this paper, the authors use a revision where all the weights are reset to 0 to be equal and restart if either E_k is not less than 0.5 or E_k becomes 0.1. By resetting the weights they do not disadvantage the Ada-Boosting learner in those cases where it reaches these values of E_k. The Ada-Boosting learner always incorporates the same number of classifiers as other methods we tested. To make this feasible, they use the approach of selecting a data set probabilistically rather than weighting the examples, otherwise a deterministic method such as C4.5 would cycle and generate duplicate members of the ensemble. That is, resetting the weights to 1/N would cause the learner to repeat the decision tree learned as the first member of the ensemble, and this would lead to reweighting the data set the same as for the second member of the ensemble, and so on. Randomly selecting examples for the data set based on the example probabilities alleviates this problem.

Arcing started out as a simple way for evaluating the effect of Boosting methods where the resulting classifiers were combined without weighting the votes. Arcing uses a simple mechanism for determining the probabilities of including examples in the training set. For the ith example in the training set, the value m_i refers to the number of times that example was misclassified by the previous K classifiers. The probability p_i for selecting example i to be part of classifier K+1's training set is defined as the value of the power empirically after trying several different values.

The paper gives the following sample of how Bagging and Boosting might work on a imaginary set of data. Since Bagging resamples the training set with replacement, some instance are represented multiple times while others are left out. So Bagging's training set 1 might contain examples 3 and 7 twice, but does not contain either example 4 or 5. As a result, the classifier trained on training set 1 might obtain a higher test-set error than the classifier using all of the data. In fact, all four of Bagging's component classifiers could result in higher test set error; however, when combined, these four classifiers can (and often do) produce test set error lower than that of the single classifier (the diversity among these classifiers generally compensates for the increase in error rate of any individual classifier).

A sample of a single classifier on an imaginary set of data.	
	(Original) Training Set
Training-set-1:	1, 2, 3, 4, 5, 6, 7, 8

A sample of Bagging on the same data.	
	(Resampled) Training Set
Training-set-1:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	7, 8, 5, 6, 4, 2, 7, 1
Training-set-3:	3, 6, 2, 7, 5, 6, 2, 2
Training-set-4:	4, 5, 1, 4, 6, 4, 3, 8

A sample of Boosting on the same data.	
	(Resampled) Training Set
Training-set-1:	2, 7, 8, 3, 7, 6, 3, 1
Training-set-2:	1, 4, 5, 4, 1, 5, 6, 4
Training-set-3:	7, 1, 5, 8, 1, 8, 1, 4
Training-set-4:	1, 1, 6, 1, 1, 3, 1, 5

Figure 3-7. Hypothetical runs of Bagging and Boosting [61]

Figure 3-7 shows hypothetical runs of Bagging and Boosting algorithms. Assume there are eight training examples. Assume example 1 is an outlier and is hard for the component learning algorithm to classify correctly. With Bagging, each training set is an independent sample of the data; thus, some examples are missing and others occur multiple times. The Boosting training sets are also samples of the original data set, but the ``hard" example (example 1) occurs more in later training sets since Boosting concentrates on correctly predicting it.

The authors draw several conclusions from their analysis. The first is that a Bagging ensemble generally produces a classifier that is more accurate than a standard classifier. For Boosting, however, they note more widely varying results. For a few data sets Boosting produced dramatic reductions in error (even compared to Bagging), but for other data sets it actually increases in error over a single classifier (particularly with

neural networks). In further tests they examined the effects of noise and determined that Boosting's sensitivity to noise may be partly responsible for its occasional increase in error.

Their results also show that the ensemble methods are generally consistent (in terms of their effect on accuracy) when applied either to neural networks or to decision trees. However, there is little inter-correlation between neural networks and decision trees except for the Boosting methods. This suggests that some of the increases produced by Boosting are dependent on the particular characteristics of the data set rather than on the component classifier. In further tests they demonstrated that Bagging is more resilient to noise than Boosting.

The authors also investigated how many component classifiers should be used in an ensemble. Consistent with previous research, their results show that most of the reduction in error for ensemble methods occurs with the first few additional classifiers. With Boosting decision trees, however, relatively large gains may be seen up until about 25 classifiers.

3.2.6 Fuzzy Logic

Fuzzy logic [42] was developed to handle problems which have incomplete, imprecise, vague or uncertain information inherent in the problem statement. These problems involve data which are at times best described by linguistic terms rather than numbers. As an example, a hospital describes patients' conditions as good, fair, serious, poor, etc.

The problem is: describing in an absolute sense these terms which are not precisely defined, and contain a significant element of subjectivity.

The originator of fuzzy logic, Zadeh [98], proposed the following approach to deal with the above problem; in particular, dealing with linguistic variables. He defined a fuzzy set as a set which allows for an object to be a member of a set to some degree. This is unlike classical set theory, which only allows for an object to be either a member of the set or excluded from the set. This "black and white" characterization, in many applications, is unsatisfactory. As an example, consider the set that describes all males who are tall as those whose height is greater than or equal to 5'8". Then a 6'0" male is a member of the set. However, a male whose height is 5'7" is not a member of the set. This implies that a man who is 1" shorter than a tall man is not tall. By the same token this approach does not differentiate between members. An individual who is 7'6" and an individual who is 6'1" are both "equal" members of the set "tall". Information about relative sizes has been lost once members have been conglomerated into a set.

Fuzzy sets differ from classical sets in that they allow for an object to be a partial member of a set. This approach can preserve relative sizing information. The relationship is defined by a membership function. For any fuzzy set A the function represents the membership function for which $\mu_A(x)$ indicates the degree of membership that x, of the universal set X, belongs to set A and is, usually, expressed as a number between 0 and 1:

$$\mu_A(x): X \rightarrow [0, 1] \qquad (3\text{-}8)$$

These sets can be either discrete or continuous. The “degree of membership” represented by the value between zero and one can be arbitrarily selected by the user or assigned according to some scale. For example if Jack is 6'3", one can arbitrarily decide that Jack is a member of the set “tall” to degree 0.8. Alternatively, a scale could be used which relates all members’ heights to that of the tallest person in the set.

To formalize the idea conveyed by classifying set members in different ways, Zadeh [97] later proposed fuzzy sets of type 2. Here, the membership grades themselves are fuzzy sets. A fuzzy set A of type 2 in a set X is the fuzzy set characterized by the fuzzy membership function as:

$$\mu_A: X \rightarrow [0, 1]^{[0, 1]} \tag{3-9}$$

where μ_A is known as a fuzzy grade, a fuzzy set in [0,1]. Mizumoto and Tanaka [58] discuss the properties of these sets and give the example of the set X = [Susie, Helen, Ruth, Pat] and A is the fuzzy set of beautiful women in X:

$$\text{A=beauty=\{middle/Susie + low/Helen + very high/Ruth + high/Pat\}} \tag{3-10}$$

where middle, low and high are fuzzy sets. As an example, instead of saying Helen is “beautiful to degree 0.3”, she is “beautiful to degree ‘low’”, thus associating a fuzzy set as opposed to a specific value. These fuzzy sets of type 2 allow for classifications of members of a fuzzy set with another fuzzy set.

The goal behind defining fuzzy sets (besides usefully describing imprecise, incomplete or vague information) is to use them to make inferences about a particular real-life problem which cannot be easily addressed using conventional mathematical models. The construction of a Fuzzy Information System (FIS) begins with determining the fuzzy sets that describe the problem. Continuing with a medical example, these may involve various qualitative measurements about a patient (low temperature, serious fracture, fair condition) which ultimately will lead to a diagnosis and then a treatment plan. Then the rules describing how these fuzzy sets interact are determined. These rules usually have an IF....THEN.... nature. The rules are then combined in some way. This process is referred to as rule composition. Finally, conclusions have to be drawn in a process known as defuzzification. The answer to the problem is typically found as a fuzzy set, and the answer needs to be "defuzzified" to provide a clear, unambiguous course of action.

Fuzzy logic is often used in conjunction with artificial neural networks (ANNs). The neural nets are used to aid in the development of FISs. As Takagi and Hayashi [89] point out, fuzzy reasoning presents particular problems:

1. the lack of a definite method for determining the membership function;
2. the lack of a learning function.

They then go on to describe an approach for using ANNs to overcome these problems. The method is to investigate if-then rules by using neural networks to determine the

membership functions of the antecedent and then determine the consequent component as the output for each rule. The approach they use is to take raw data (say, in a control problem), apply a conventional clustering algorithm to group the data into clusters and to apply an ANN to this clustered data to determine the membership of a pattern within particular fuzzy sets.

The authors apply this approach to two real-world problems - estimation of chemical oxygen demand density in Osaka Bay and the estimation of the roughness of a ceramic surface. Their method in both cases out-performed more conventional methods. This combination of neural networks and fuzzy reasoning does allow for automatic generation of μ in certain applications.

As has previously been stated, finding a solution to a fuzzy logic problem requires defuzzification. There are various techniques available. Lee [48] describes the three main approaches as the max criterion, mean of maximum and the center of area (most common). The max criterion method finds the point at which the membership function is a maximum. The mean of maximum takes the mean of those points where the membership function is at a maximum. The most common method is the center of area method which finds the center of gravity of the solution fuzzy sets. Lee states, "Unfortunately, there is no systematic procedure for choosing a defuzzification strategy." Although the process of reducing the final fuzzy set to a crisp value does seem appropriate for control problems much information is lost by doing this and further work needs to be done on how to use the information available in the solution fuzzy set.

In the main, the approaches adopted in fuzzy logic problems have been very domain specific, not applied to large complex problems and the evaluation of the efficacy of their approach is often not systematic enough for conclusions to be drawn. Determining the membership functions, the rules, the operators and the defuzzification strategy is a difficult task that requires a good deal of effort before it can be said that any particular system is the optimal fuzzy system for that particular application.

3.2.7 Summary of Other Methods

The most common methods in the literature at present for analyzing system data are variations of neural network and/or fuzzy logic techniques. However, there are a number of other techniques which can be used to analyze system data. Some of these methods are summarized in a table in the appendix for chapter 3.

IV. Mathematical Programming Model

4.1 Model Development

A prognostics system, at an abstract level, is composed of two parts. The first part consists of sensors which are attached to various parts of a mechanical, electrical, or other kind of system, and report the system data. The second part is a reasoning function which interprets this data to provide an assessment of current and future system health. This section develops a mathematical model of the former part to determine a "best", latter reasoning function configuration. The objective function calculation approach is present in the next chapter.

Different types of models can be used to represent a particular system. For the purposes of this discussion, a model which emphasizes a system's components and subcomponents is used.

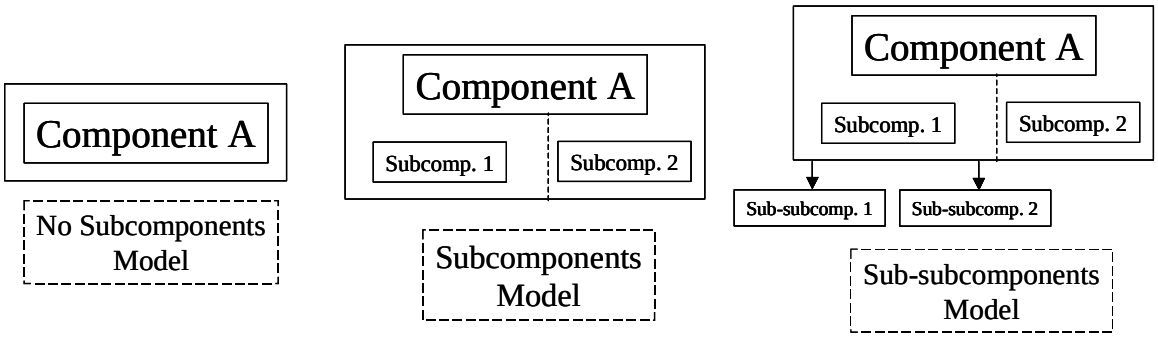

Figure 4-1. Different levels of detail for modeling a system

Figure 4-1 shows three different possible levels of detail for modeling a system. A system can be modeled at a component level, as shown in the left side of Figure 4-1. In

the middle of Figure 4-1 is a system in which components are divided into subcomponents. This model will be the focus of this discussion, and will be used to model a notional prognostics system. The right-hand side of Figure 4-1 shows a system model where the subcomponents are further decomposed into sub-subcomponents. This level of abstraction can continue for any number of levels to the required level of detail.

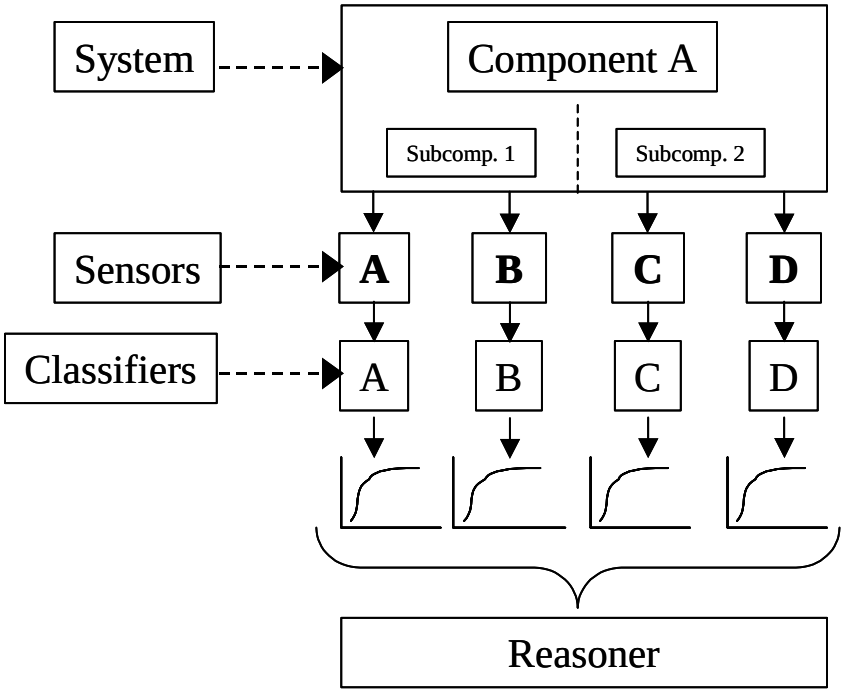

Figure 4-2. A pictorial representation of a simple system

Figure 4-2 shows a simple, generalized system. This simple system consists of one component and two subcomponents. Each subcomponent may have up to two sensors attached, each providing information to a classifier. The classifier then determines the subcomponent state based on the sensor information. The reasoner combines all the information from the classifiers and makes a final determination about the system state.

The reasoner also serves as the interface between the system and the human operators. The challenge is determining whether all sensors are needed or whether there is a sufficient subset of sensors. A mathematical model can help answer the challenge.

For the purposes of this representation, a component is considered an abstract grouping of less complicated, smaller substructures. These substructures are represented as subcomponents in Figure 4-2. As an example, an aircraft engine may be considered as a component. One subcomponent might be the fuel delivery and ignition system; another subcomponent might be the turbine blades and the associated control mechanism. Of course, these definitions of component and subcomponent can be applied to any desired system at any level, depending on the level of detail/complexity/aggregation required for a particular application.

The following assumptions underlying the subsequent mathematical formulation are in keeping with a general philosophy of the prognostics community at the present time. In this particular model, all the subcomponents are considered critical parts of the system component. If any subcomponent fails, the parent component and the system will also fail. System parts which are not critical to component/system functionality are not addressed with this model. A specific term used to describe this principle is "Failure Mode and Effects Criticality Analysis" (FMECA) [12]. FMECA analysis is concerned solely with different system failure modes, as opposed to system operations which may be aberrant, but do not affect system operation or induce system failure modes. In the FMECA, the system's different failure modes are ranked according to severity, likelihood

of occurrence, and observability. For each failure mode, a group of system experts determines preliminary symptoms (if any), and which system sensors would be useful in detecting these symptoms. System modes/conditions which do not significantly affect the operation of the system are not considered in the FMECA [9].

Logan, et al [50], [51] recommend a similar modeling approach. They use the engineering knowledge of domain experts to construct a diagnostic knowledge base suitable for neural network training. They call their approach a comprehensive "Failure Mode and Effects Analysis" (FMEA) on the appropriate mechanical system. Like the FMECA, a FMEA provides a comprehensive listing of probable failure modes of all "major" mechanical system components, where "major" is defined as the level of detail appropriate for that particular system. This information is obtained from interviews with engineering crews and maintenance personnel. Technical orders are also reviewed to ensure the information is correct and complete. The review also includes information on all available sensor measurements, and identifies the fault/symptom relationships required for an effective monitoring program. Similar to the FMECA approach, non-failure modes are not considered.

In Figure 4-2, each of the two subcomponents have potentially two sensors. These sensors represent the collection and reporting of appropriate information about the specific part of the subcomponent they are monitoring. Typically, the sensors are assigned to collect a specific type of phenomenology from the subcomponent. These phenomenologies may include pressure, temperature, vibration, and electrical current.

Returning to the preceding example of an aircraft engine, if one subcomponent represents the fuel delivery and ignition system, one sensor may monitor the pressure within the fuel delivery system, and the other sensor may record the timing and strength of the spark (the electrical current) the ignition system produces.

The sensors' collected data are sent to the classifier functions. The classifier checks the reported data to ensure the sensor is functioning correctly, processes the raw signal data, and then uses this processed data to assess the current subcomponent state and predict the future subcomponent state. The reasoner accumulates these assessments and predictions from the classifiers, and uses them to assess the current system state and predict the future system state. (Correctly functioning sensors send two data streams to the classifier. The main data stream is the subcomponent data. The second data stream verifies the sensor's functionality. A correctly functioning sensor sends a specific bit every x^{th} bit interspersed with the main data stream to verify the sensor is functioning correctly. If the classifier does not receive this specific bit, it will disregard the incoming data stream until it again receives this bit from the sensor.) The methods the classifier may use to interpret the processed data can be quite varied. These methods can range from mathematical techniques such as neural nets and Bayesian networks to case-base reasoning and/or expert systems, or any combination of techniques.

The analytical tool used in the model represented in Figure 4-2 represents is the Receiver Operating Characteristic (ROC) curve. A ROC curve is the graph of a relation which summarizes the range of performance of a particular signal detection algorithm. The

signal algorithm is designed to detect a particular signal of interest among other signals which may serve to mask the signature of the desired signal. A ROC curve typically compares the classifier's signal of interest detection rate to the classifier's false alarm rate (reporting a signal of interest when that signal has not actually occurred). ROC curves are commonly used to describe the performance of imperfect diagnostic systems, especially in the fields of automatic target recognition and biomedical research [5].

In the models considered here, each system will typically have more than one component, each component will typically have more than one subcomponent, and each subcomponent will typically have more than one sensor/classifier pair. For a given subcomponent, all possible sensors of the appropriate type (pressure, temperature, etc.) are possible candidates. As before, every subcomponent is assumed critical for system operation. Further, each subcomponent of a particular system is assumed to have at least one sensor attached to it (the mathematical formulation will explicitly enforce this structural requirement).

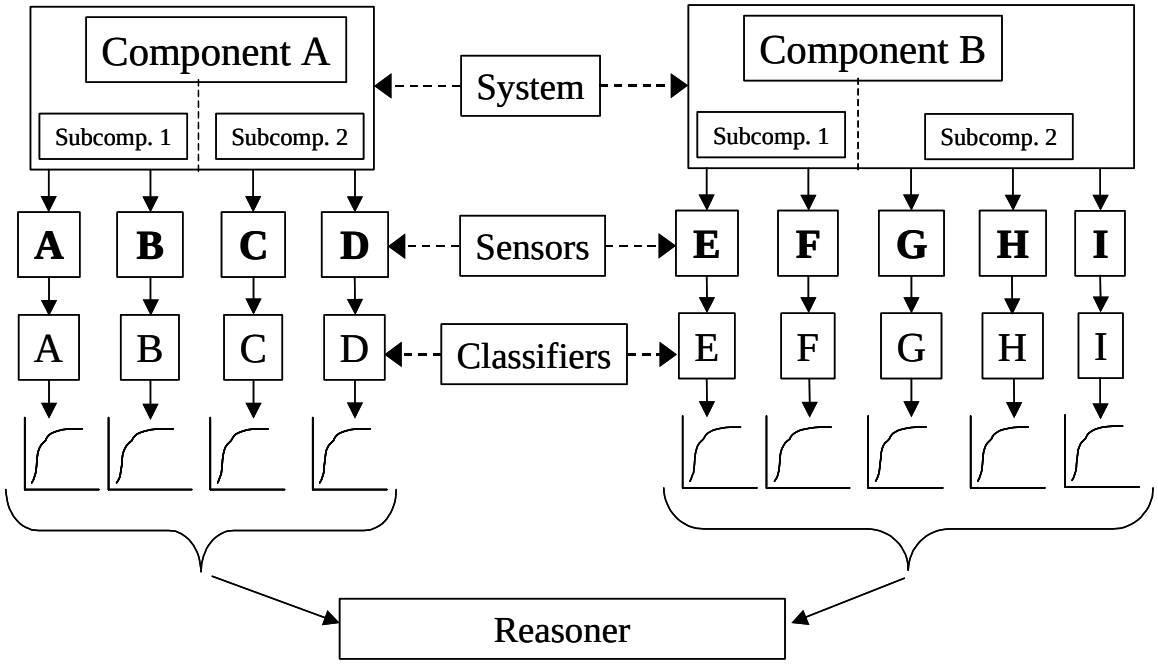

Figure 4-3. A pictorial representation of a system with multiple components

Figure 4-3 expands the model structure found in Figure 4-2. There are now two components, each with two critical subcomponents, and each subcomponent has multiple candidate sensors.

In an ideal environment all sensors are included in a system. However, weight, space, and data processing limitations prohibit such a configuration in actual systems. Thus, expert judgment may be used to pick a subset of sensors. Mathematical modeling provides a means to improve upon expert judgment to prescribe some best subset of sensor/classifier pairs to include in a system. The next section develops a mathematical

formulation to accomplish this task. This formulation uses the model structure presented in Figure 4-2 as a basis.

4.2 Formulation

A mathematical programming formulation is used for selecting an optimally sized sensor set. Let M denote the number of sensors available for use, and define $\mathbf{S} = \{\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_M}\}$ to be the set of sensors available. Define $\mathbf{A} = \{S: S \text{ is a non-empty subset of sensors of } \mathbf{S}\}$, and note that $\mathbf{A}$ is the power set of $\mathbf{S}$, excluding the empty set, denoted as $\mathbf{A} = P(\mathbf{S}) - \varnothing$. Note that $\text{card}(\mathbf{A}) = 2^M - 1$, that is, there are $2^M - 1$ different sets in $\mathbf{A}$. Let $S_i \in \mathbf{A}$, $i = 1, 2, \ldots 2^M - 1$ be an enumeration of $\mathbf{A}$.

Each sensor has its own classifier. The terminology $\mathbf{A_i}$ is understood to refer to any specific sensor-classifier pair. For a set $S_i \in \mathbf{A}$ containing more than one sensor, a fusion rule R will be used to fuse the classifiers for each sensor into a single classifier. This activity will be denoted as $R: \mathbf{A} \rightarrow \mathbf{G_R}(S)$,

where $\mathbf{G_R}(S) = \{\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_M}, R(\mathbf{A_1}, \mathbf{A_2},), \ldots, R(\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_M})\}$

$= \{R(S) \mid S_i \in \mathbf{A}\}$.

The set $\mathbf{G_R}(S)$ contains all the fused classifiers for each $S_i \in \mathbf{A}$. Note that ensembles consisting of a single sensor-classifier pair do not undergo fusion since the ensemble already has a single classifier.

This paragraph defines the variables and constants found in the formulation below. The objective is to find a sufficient sensor/classifier subset for the given system. Thus, the objective function value P_{TP} is the probability of obtaining a true positive (the prognostics system indicates a system failure when the system has actually failed). The variable P_{FP} is the probability of obtaining a false positive (the prognostics system indicates a system failure when the system has not failed). The value P_{FP}^{*} is defined as the maximum acceptable P_{FP} for any ensemble. The value of P_{TP} depends on F_i, a probability function that depends a particular ensemble $S_i \in \mathbf{A}$, and P_{FP}. The evaluation of P_{TP} is developed in Chapter 5. The variable d_{sm}^{c} is an indicator variable that is 1 if the m^{th} sensor is retained for the s^{th} subcomponent on the c^{th} component, and 0 otherwise. The variable c_{sm}^{c} is the cost of employing the m^{th} sensor on the s^{th} subcomponent on the c^{th} component. This fixed cost is assumed to be independent of the other sensors in the ensemble. The variable SC_s^c denotes the maximum number of sensors considered for the s^{th} subcomponent of the c^{th} component. The parameter SC^c is the number of subcomponents present on the c^{th} component. The parameters c_i^{FP} and c_i^{FN} denote the cost of an erroneous prognostics system reading associated with the i^{th} ensemble $S_i \in \mathbf{A}$, $i = 1, 2, \ldots 2^M - 1$. The errors are defined as follows: either the system indicates a fault when no fault is present (cost denoted by c_i^{FP}), or fails to indicate a fault when a fault is present (cost denoted by c_i^{FN}). The constant B_E^c is the budget (maximum allowable cost) for the costs of retaining a given sensor ensemble on the c^{th} component. The constant B_O is the budget (maximum allowable cost) for the sensor errors. The mixed-integer nonlinear programming (MINLP) formulation is then given by

$$F(P_{FP}^{*}) \equiv \max_{\mathbf{A} \in \mathbf{G}_R(S)} P_{TP}(\mathbf{A}) \quad \text{subject to} \quad P_{FP}(\mathbf{A}) \le P_{FP}^{*} \tag{4-1}$$

(structural constraints—there are $SC = \sum_{c=1}^{C} SC^c$ of these constraints, one for each component.)

$$\sum_{m=1}^{SC_s^c} d_{sm}^c \ge 1 \qquad c = 1, \ldots, C;\ s = 1, \ldots, SC^c$$

$$d_{sm}^c = \begin{cases} 1 \text{ if } m\text{th sensor retained on } s\text{th} \\ \text{sub-component of } c\text{th component} \\ 0 \text{ otherwise} \end{cases}$$

(employment cost constraints—there are C of these constraints, where C is the number of components)

$$\sum_{s=1}^{SC^c} \sum_{m=1}^{SC_s^c} d_{sm}^c c_{sm}^c \le B_E^c \qquad c = 1, \ldots, C$$

(operational cost constraint)

$$c_i^{FP} + c_i^{FN} \le B_O \qquad S_i \in \mathbf{A}$$

$$0 \le P_{FP} \le P_{FP}^{*} \le 1$$

$$d_{sm}^c \in \{0, 1\} \qquad c = 1, \ldots, C;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

$$c_{sm}^c, c_i^{FP}, c_i^{FN}, B_E, B_O \ge 0 \qquad S_i \in \mathbf{A};\ c = 1, \ldots, C;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

This formulation accommodates two key requirements associated with this general problem. The first requirement is to consider all appropriate sensor ensembles for a given system (not necessarily all possible ensembles). This requirement is met with the employment cost and structural constraints. The employment cost constraint ensures that

budget associated with a particular sensor ensemble is not exceeded, and the structural constraint ensures that each critical subcomponent is assigned at least one sensor. The second requirement is to ensure a given sensor ensemble does not exceed the maximum allowable error rate. The operational cost constraint ensures this requirement is met. There is more discussion of the operational cost constraint in section 4.5.

This formulation apportions employment costs to specific system components (recall that employment cost does not refer just to the actual monetary expense). Size, power, weight, and similar constraints are likely to be different for any given system component. Accordingly, this formulation enforces a specific budget for each component.

4.3 Towards a Heuristic Solution Procedure: Subset Generation

This section details a methodology for partitioning the solution space, and indexing the possible solutions in the resulting subspaces. A subset ordering method is presented to ensure each subset $S_i \in \mathbf{A}$ is considered during the solution process.

If there are M defined sensors, there are $2^M - 1$ possible sensor combinations containing at least one sensor within the system (the trivial case of an empty ensemble is omitted). There are also M different sensor ensemble sizes, ranging from one sensor throughout the system, to all M sensors employed. Formulation (4-1) can then be partitioned into M subproblems, one partition for each sensor ensemble size, in order to conveniently enumerate the solution space, and to partition the solution space into more manageable subspaces. Define an index *j* as the number of sensors contained in a particular partition

(*j*= 1 to M). Each of the j partitions contains ${}_M\mathbf{C}_j$ possible solutions, where ${}_n\mathbf{C}_k = \binom{n}{k}$.

More formally, this can be expressed as

$$\mathbf{A}_j = \{S: S \neq \varnothing, \text{card}(S) = j\} \quad j = 1,\ldots, M. \tag{4-3}$$

$\mathbf{A}_j \subset \mathbf{A}$ is the set of sets corresponding to the partition consisting of *j* sensors selected among the M sensors available. Note that card($\mathbf{A}_j$) = ${}_M\mathbf{C}_j$ and $\mathbf{A} = \bigcup_{j=1}^{M} \mathbf{A}_j$.

4.3.1 Subset Ordering

A logical ordering of all the sensor ensembles allows for a quick and thorough evaluation of the solution space. To this end, this section develops a notation for tracking each ensemble, and presents two different ordering methods.

Each particular ensemble in $\mathbf{A}$ can be given a unique index. One indexing scheme is a natural indexing scheme defined as follows. Recall *j*= 1 to M (sensors). When the index *j* is equal to 1, the M ensemble sensor sets are each of size 1, and so are indexed from 1 to M. When the index j is equal to 2, the ${}_M\mathbf{C}_2$ ensemble sensor sets are of size 2, and *i* is indexed as

$$i = M + 1, M + 2, \ldots, M + {}_M\mathbf{C}_2. \tag{4-4}$$

When the index *j* is equal to 3, the ${}_M\mathbf{C}_3$ ensemble sensor sets are of size 3, and *i* is indexed as

$$i = M + 1 + {}_M\mathbf{C}_2, M + 2 + {}_M\mathbf{C}_2, \ldots, M + {}_M\mathbf{C}_2 + {}_M\mathbf{C}_3. \tag{4-5}$$

In general, when the index *j* is equal to n, where M > n ≥ 3, the ensemble sensor sets are of size n, with *i* is indexed as ${}_{M}\mathbf{C}_{k-1}$

$$i = M + 1 + \sum_{k=3}^{n} {}_{M}C_{k} \, , \, M + 2 + \sum_{k=3}^{n} {}_{M}C_{k} \, , \, M + \sum_{k=3}^{n+1} {}_{M}C_{k} \, . \qquad (4\text{-}6)$$

The natural ordering sequence is completed by maintaining a lexicographic order within any S_i. A natural ordering sequence is a particular lexicographical method that orders all subsets of a given set according to the number of items in the subset, from the smallest number of items to the largest. This ordering allows for the potential elimination of all sensor subsets of the same size.

Table 4-1. "Natural" sequence for a set of 6 sensors

Index	Sensor Ensemble
1	s1
2	s2
3	s3
4	s4
5	s5
6	s6
7	s1s2
8	s1s3
9	s1s4
10	s1s5
11	s1s6
12	s2s3
13	s2s4
14	s2s5
15	s2s6
16	s3s4
17	s3s5
18	s3s6
19	s4s5
20	s4s6
21	s5s6

Index	Sensor Ensemble
22	s1s2s3
23	s1s2s4
24	s1s2s5
25	s1s2s6
26	s1s3s4
27	s1s3s5
28	s1s3s6
29	s1s4s5
30	s1s4s6
31	s1s5s6
32	s2s3s4
33	s2s3s5
34	s2s3s6
35	s2s4s5
36	s2s4s6
37	s2s5s6
38	s3s4s5
39	s3s4s6
40	s3s5s6
41	s4s5s6
42	s1s2s3s4

Index	Sensor Ensemble
43	s1s2s3s5
44	s1s2s3s6
45	s1s2s4s5
46	s1s2s4s6
47	s1s2s5s6
48	s1s3s4s5
49	s1s3s4s6
50	s1s3s5s6
51	s1s4s5s6
52	s2s3s4s5
53	s2s3s4s6
54	s2s3s5s6
55	s2s4s5s6
56	s3s4s5s6
57	s1s2s3s4s5
58	s1s2s3s4s6
59	s1s2s3s5s6
60	s1s2s4s5s6
61	s1s3s4s5s6
62	s2s3s4s5s6
63	s1s2s3s4s5s6

Table 4-1 shows a natural ordering for a system with six sensors. As the table shows, sensor subsets of the same size are grouped together.

There are other subset ordering methods. According to the paper by Furnival and Wilson [34], a lexicographic ordering method would look like the ordering depicted in Table 4-2. This ordering method groups the subsets by sensors-the first grouping of subsets all contain sensor 1, the next grouping contains sensor 2, and so forth. In their paper, Furnival and Wilson include FORTRAN code to generate these different subset orderings. Their code has been modified to generate the natural ordering sequence for up to nine sensors. Other sources also present these ordering techniques as ways to codify a number of different subsets [92], [23].

Table 4-2. "Lexicographic" sequence for a set of 6 sensors

Index	Sensor Ensemble
1	s1
2	s1s2
3	s1s2s3
4	s1s2s3s4
5	s1s2s3s4s5
6	s1s2s3s4s5s6
7	s1s2s3s4s6
8	s1s2s3s5s6
9	s1s2s3s5
10	s1s2s3s6
11	s1s2s4
12	s1s2s4s5
13	s1s2s4s5s6
14	s1s2s4s6
15	s1s2s5
16	s1s2s5s6
17	s1s2s6
18	s1s3
19	s1s3s4
20	s1s3s4s5
21	s1s3s4s5s6

Index	Sensor Ensemble
22	s1s3s4s6
23	s1s3s5
24	s1s3s5s6
25	s1s3s6
26	s1s4
27	s1s4s5
28	s1s4s5s6
29	s1s4s6
30	s1s5
31	s1s5s6
32	s1s6
33	s2
34	s2s3
35	s2s3s4
36	s2s3s4s5
37	s2s3s4s5s6
38	s2s3s4s6
39	s2s3s5
40	s2s3s5s6
41	s2s3s6
42	s2s4

Index	Sensor Ensemble
43	s2s4s5
44	s2s4s5s6
45	s2s4s6
46	s2s5
47	s2s5s6
48	s2s6
49	s3
50	s3s4
51	s3s4s5
52	s3s4s5s6
53	s3s4s6
54	s3s5
55	s3s5s6
56	s3s6
57	s4
58	s4s5
59	s4s5s6
60	s4s6
61	s5
62	s5s6
63	s6

The natural ordering scheme is used for this presentation. In the natural ordering scheme, within each sensor size, the ensembles are ordered from the smallest number to the largest number.

This methodology is used in the appendix to develop a methodology to quickly reduce the size of the solution space that must be searched, if certain conditions about the system and its operation hold.

4.4 A Sample Formulation Example

This section illustrates the mathematical formulation with an example. The development of the solution computation techniques is presented in Chapter V.

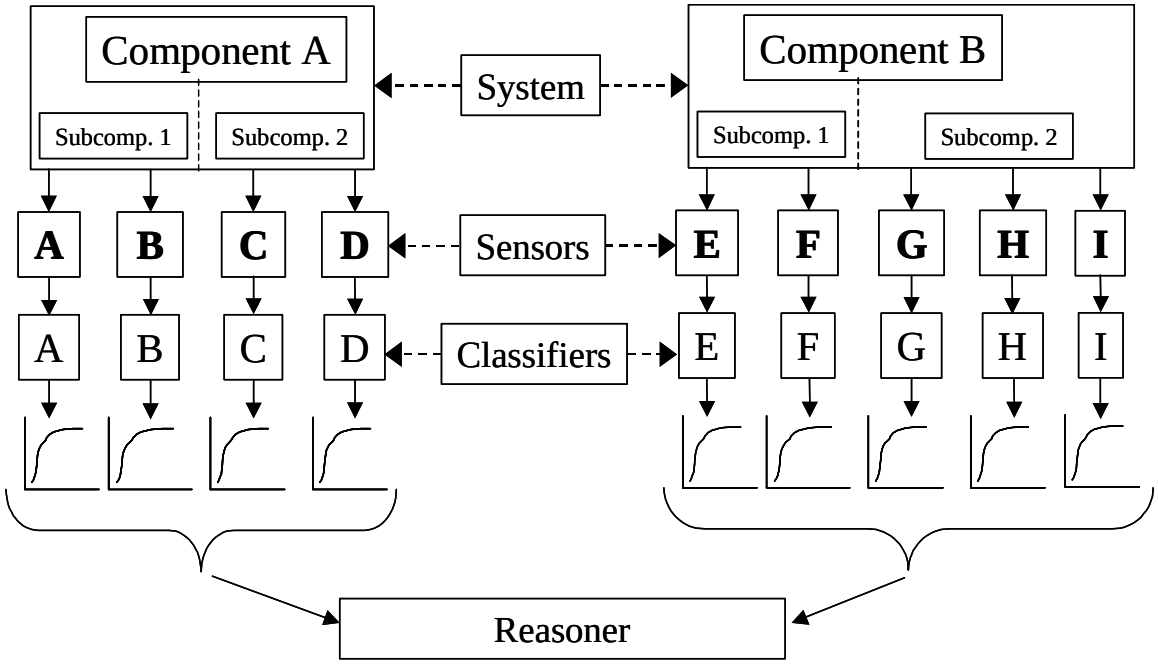

Figure 4.4. Figure 4.3 reproduced for clarity

In this example, there are nine sensors and corresponding classifiers (M), four critical subcomponents (SC), and two system components (C). The specific formulation is:

$$F(P_{FP}^{*}) \equiv \max_{\mathbf{A} \in \mathbf{G}_R(S)} P_{TP}(\mathbf{A}) \qquad (4\text{-}7)$$

subject to $P_{FP}(\mathbf{A}) \le P_{FP}^{*}$

(structural constraints)

$$P_{FP}(R(S)) \le p$$

$$\sum_{m=1}^{SC_1^A} d_{sm}^A \ge 1 \qquad SC_1^A = 2$$

$$\sum_{m=1}^{SC_2^A} d_{sm}^A \ge 1 \qquad SC_2^A = 2$$

$$\sum_{m=1}^{SC_1^B} d_{sm}^B \ge 1 \qquad SC_1^B = 2$$

$$\sum_{m=1}^{SC_2^B} d_{sm}^B \ge 1 \qquad SC_2^B = 3$$

$$d_{sm}^c = \begin{cases} 1 \text{ if mth sensor retained on sth} \\ \text{sub-component of cth component} \\ 0 \text{ otherwise} \end{cases}$$

(employment cost constraints)

$$\sum_{s=1}^{SC^A} \sum_{m=1}^{SC_s^A} d_{sm}^A c_{sm}^A \le B_E^A$$

$$\sum_{s=1}^{SC^B} \sum_{m=1}^{SC_s^B} d_{sm}^B c_{sm}^B \le B_E^B$$

(operational cost constraint)

$$c_i^{FP} + c_i^{FN} \le B_O \qquad S_i \in \mathbf{A}$$

$$0 \le P_{FP} \le P_{FP}^{*} \le 1$$

$$d_{sm}^c \in \{0, 1\} \qquad c = 1, 2;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

$$c_{sm}^c, c_i^{FP}, c_i^{FN}, B_E, B_O \ge 0 \qquad S_i \in \mathbf{A};\ c = 1, 2;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

Note that there are two employment cost constraints corresponding to the two system components, and there are four structural constraints, corresponding to the four critical system subcomponents. The operational cost constraint remains the same. Assume that each $S_i \in \mathbf{A}$ is indexed in natural order.

4.5 A Possible Modification to The Operational Cost Constraint

The formulation presented in section 4.2 is time independent. It may be useful for a particular system to model time in the operational cost constraint. This section builds a methodology to accommodate that capability.

There are four possible outcomes for the prognostic system's assessment of the data stream. These outcomes are summarized in the table below.

Table 4-3. Summary of sensor readings and their associated probabilities

Reality (Truth)	Sensor Report	
	No Fault (*N* readings)	Fault (*n* readings)
No Fault	True Negative (P_{TN})	False Positive (P_{FP}) Cost c_i^{FP}
Fault	False Negative (P_{FN}) Cost c_i^{FN}	True Positive (P_{TP})

- P_{TN} is the probability that the prognostics system does not report a fault when no fault is present.
- P_{FP} is the probability that the prognostics system reports a fault when a fault is not present. The cost of this event is c_i^{FP}.

- P_{FN} is the probability that the prognostics system does not a report a fault when one is actually present. The cost of this event is c_i^{FN}.
- P_{TP} is the probability that the prognostics system reports a fault when one is actually present.

The c_i^{FP} and c_i^{FN} costs may be more appropriately expressed as a function of P_{FP} and P_{FN}, respectively. The larger P_{FP} and P_{FN}, the more often the cost will be incurred. However, the idea of "often" introduces a time element into the formulation. Let *N* be the total number of no-fault readings for a given time period, and let *n* be the number of fault readings for the same time period. Let the total number of readings be represented by T = *N* + *n*. Then the quantities *N* and *n* can be considered the expected number of "no fault" and "fault" readings, respectively, per T trials.

Estimates for the number of failure readings which might occur during a given sortie can be obtained from Mean Time Between Failures (MTBF) information. MTBF is the number of time units (usually hours) that pass before a component, assembly, or system fails. It is a measure of hardware product or component reliability, and is a commonly-used variable in reliability and maintainability analyses. The MTBF for a particular component can be used to determine estimates for *N* and *n*, given the rate at which system readings are collected. Let t_T denote the system reading rate and S be the length in time of the sortie. Then

$$T = \lceil t_T S \rceil \tag{4-8}$$

$$n = \left\lceil \frac{S}{MTBF} \right\rceil \tag{4-9}$$

and

$$N = \left\lceil T - \frac{S}{MTBF} \right\rceil \tag{4-10}$$

As a specific example, assume a 20 hour sortie (S), a system reading (t_T) every second, and an MTBF of 10 hours. Then T = 72,000, N = 71,998, and n = 2.

The modified form of the operational cost constraint would be:

$$P_{FP} n c_i^{FP} + P_{FN} N c_i^{FN} \leq B_O \tag{4-11}$$

4.6 A More General Formulation

The formulation presented in section 4.2 apportions employment costs among the different system components. The underlying rationale is that size, power, weight, and similar constraints are likely to be different for any given system component. However, there are parts of the cost of employing a sensor ensemble that might be freely transferred among system components, such as monetary costs. Additionally, there may be system components where size, power, weight, and similar constraints are not limiting factors. Here, the employment cost constraint is relaxed to allow for an overall system budget. The new formulation is given by

$$F(P_{FP}^{*}) \equiv \max_{\mathbf{A} \in \mathbf{G}_R(S)} P_{TP}(\mathbf{A}) \quad (4\text{-}11)$$

subject to $P_{FP}(\mathbf{A}) \le P_{FP}^{*}$

(structural constraint--there are SC of these constraints, where SC is the number of subcomponents)

$$\sum_{m=1}^{SC_s^c} d_{sm}^c \ge 1 \qquad c = 1, \ldots, C;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

$$d_{sm}^c = \begin{cases} 1 \text{ if mth sensor retained on sth} \\ \text{sub-component of cth component} \\ 0 \text{ otherwise} \end{cases}$$

(employment cost constraint—there is now only one constraint)

$$\sum_{s=1}^{SC} \sum_{m=1}^{SC_s} d_{sm}^c c_{sm}^c \le B_E \qquad s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

(operational cost constraint)

$$c_i^{FP} + c_i^{FN} \le B_O \qquad S_i \in \mathbf{A}$$

$$0 \le P_{FP} \le P_{FP}^{*} \le 1$$

$$d_{sm}^c \in \{0, 1\} \qquad c = 1, \ldots, C;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

$$c_{sm}^c, c_i^{FP}, c_i^{FN}, B_E, B_O \ge 0 \qquad S_i \in \mathbf{A};\ c = 1, \ldots, C;\ s = 1, \ldots, SC^c;\ m = 1, \ldots, SC_s^c$$

Note that only the employment cost constraint was modified from the general formulation. The solution details are presented in Chapter V.

V. Fusion Rule Assessment

5.1 Fusion Rule Definitions

Given a system like that shown below in Figure 5-1, the objective is to find the optimum allocation of sensors that provides the "best" ROC curve for determining the system status. This notion of a "best" ROC curve is developed in a later section. The ROC curve for each classifier under consideration is assumed to be known for the discussion that follows.

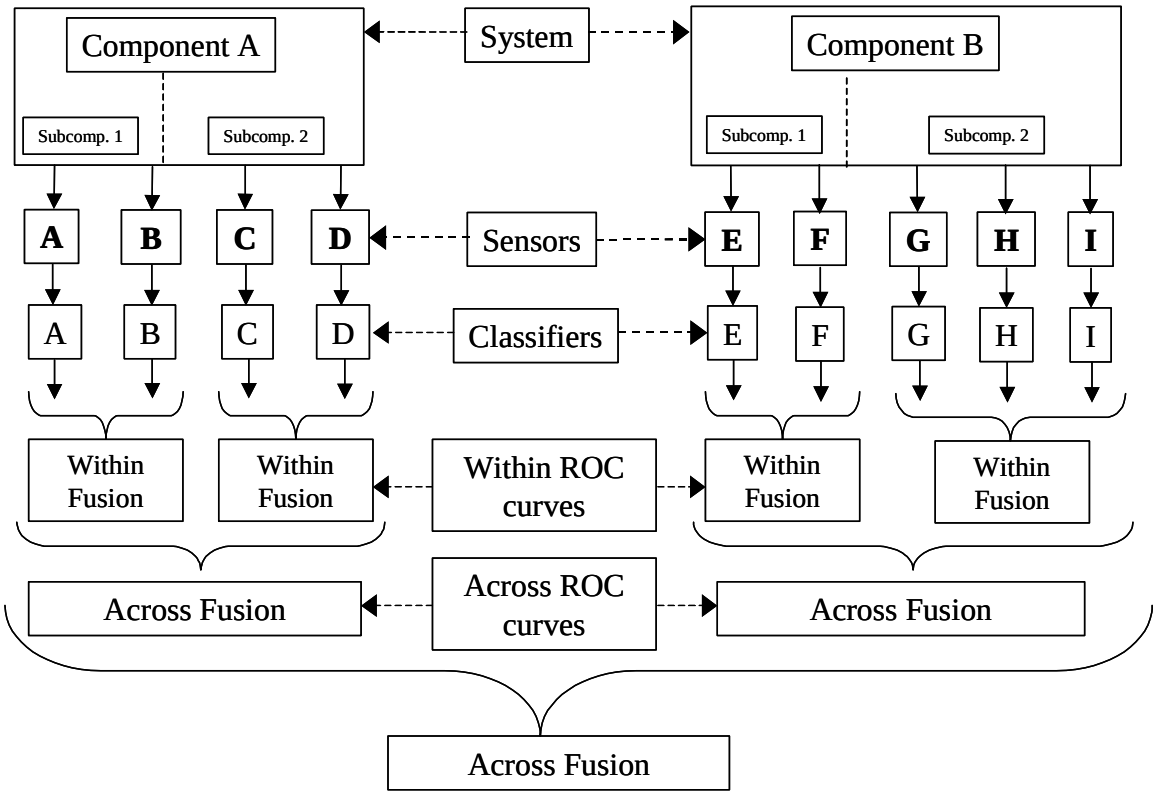

Figure 5-1. Graphic showing the terms for the different fusion operations

Figure 5-1 shows the terminology developed for each fusion method. The ROC curves associated with each classifier need to be combined to provide a single ROC curve associated with the subcomponent. This first fusion method will be called *within* fusion. The *within* fusion method creates a ROC curve for each subcomponent that has multiple (or redundant) sensors, although a subcomponent does not necessarily require multiple (or redundant) sensors. The ROC curves for the subcomponents (whether they have multiple sensors or not) will be called *within* ROC curves.

The ROC curves associated with each subcomponent need to be combined to provide a single ROC curve associated with their common component. This second fusion method will be called *across* fusion. The *across* fusion method is used to combine *within* ROC curves. The *across* fusion method creates a ROC curve for each system component. (A component does not necessarily have to have multiple subcomponents). The ROC curves resulting from this operation will be called *across* ROC curves. Each of these two fusion methods (*within* and *across*) is described in detail in the next section.

5.2 Fusion Methods

At the lowest level of system decomposition (the subcomponent level in this model), there are a significant number of options for sensor allocation, even on a single subcomponent. To accurately categorize the current and future states of a particular system, sensors must be appropriately placed on all subcomponents. As a reminder, referring back to Figure 5-1, all subcomponents in this model are assumed to be critical to component and system operation. In any given system, there is a balance between using

enough sensors to ensure a high level of confidence in the prognostic system's reports on system status, while not exceeding power, weight, bandwidth, and other limitations which restrict the number of sensors which may be used. Although it may be desirable to measure the performance of every part of every subcomponent, and include redundant sensors on the most important subcomponents, such configurations are not likely to be feasible. The underlying assumption of this desire for these types of redundant configurations is that multiple sensors will provide a higher level of confidence and accuracy in the prognostic system's reported results. To reflect that, the *within* fusion method creates a ROC curve which is always equal to or greater than each of the ROC curves of the individual classifiers which contributed to it. This topic is explored further at the end of section 5.2.1.

5.2.1 Within Fusion

The *within* fusion methodology is developed using the following definitions. Let Ξ be the event set. Let $\mathbf{X}$ be the feature space, and let x be a specific instantiation of this set $\mathbf{X}$. Let $\mathbf{X}_f$ be the set of system feature vectors indicating a system failure. Let $p_f = Pr(x \in \mathbf{X}_f)$ be the prior probability that a critical subcomponent part will fail. The corresponding definition and prior probability of the critical subcomponent part not failing (operating nominally) is $\mathbf{X}_n$ and $p_n = (1 - p_f) = Pr(x \in \mathbf{X}_n)$. The critical subcomponent part is assumed to only take on these two states (nominal or failed). These two states will be termed a label set, and will be denoted by $\mathbf{L} = \{F, N\}$.

These two values of the label set are mutually exclusive and collectively exhaustive; i.e., $\mathbf{L} = \mathbf{L}_n \cup \mathbf{L}_f$, and $\mathbf{L}_n \cap \mathbf{L}_f = \varnothing$.

The critical subcomponent part is assumed to have two sensors **A** and **B** attached to it. Let A_θ and B_ϕ refer to the classifiers for sensor **A** and sensor **B** on the system, respectively, where $\theta \in \Theta$ and $\phi \in \Phi$, where Θ and Φ are admissible sets of parameters associated with tuning each classifier [5]. These classifiers are assumed to assess failure or non-failure independently (this assumption will be addressed in more detail in section 5.2.3).

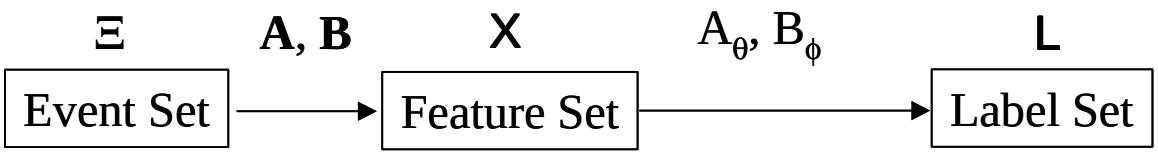

Figure 5-2. Methodology summary

Figure 5-2 summarizes the methodology presented to this point. System events are detected by sensors A and B. These sensors report their collected data to the classifiers, which assign a label (either nominal-N or failed-F), to the data stream.

The expression $C_{\theta,\phi}$ will be used to denote the concatenated classifier of the classifiers A_θ and B_ϕ.

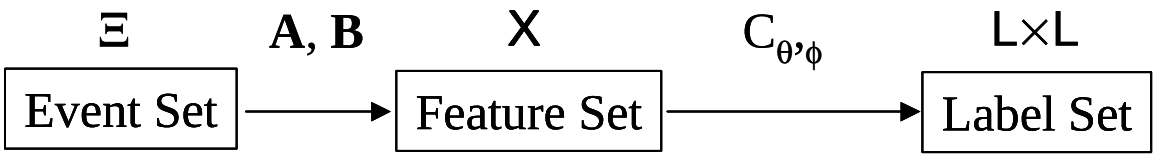

Figure 5-3. Function of the concatenated classifier

Figure 5-3 shows the transformation of system event data into a label set via the concatenated classifier. Since the concatenated classifier consists of both classifiers A_θ and B_ϕ, the label set consists of two distinct labels.

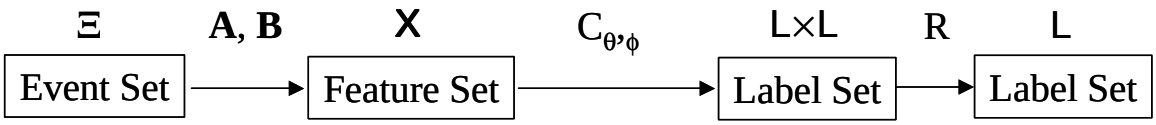

Figure 5-4. Transformation of the system event to a final system functionality classification

Figure 5-4 shows the complete notional flow of information through this model. Once the concatenated classifier has determined two distinct labels, a rule R transforms these two labels into a single label. Specifically, R(L, L) = L ∨ L, where the ∨ operator is defined as in Table 5-1 below.

Table 5-1. Definition of the ∨ operator

∨	**F**	**N**
F	F	F
N	F	N

Table 5-1 shows the label obtained from each classifier on the exterior of the table. The combination of the two labels is shown in the interior of the table. In this table, the $\vee$ operator is defined as a "logical or" rule. A "logical or" rule is used to declare a system failure; if either or both of the classifiers indicate a failed condition, the system is assumed to have failed. This formulation is consistent with the FMECA assumption that every component is critical. Only if both classifiers consider the system to be operating nominally is the output from the rule R a nominal reading.

The expression $D_{\theta,\phi}$ will be used to denote this fused classifier. Note that $D_{\theta,\phi} = R \circ C_{\theta,\phi}$, and

$$D_{\theta,\phi}(x) = R \circ C_{\theta,\phi}(x) = R(A_\theta(x), B_\phi(x)) = A_\theta(x) \vee B_\phi(x) \qquad (5\text{-}1)$$

The operator $\circ$ denotes the transformation of the concatenated classifier $C_{\theta,\phi}$ to the fused classifier $D_{\theta,\phi}$ using the rule R.

There are certain probabilities associated with each possible classification event, given the single subcomponent's operational state. The probability of a true positive is defined to be a classifier declaring a failure, given the system has failed. The probability of a false positive is defined to be a classifier declaring the system has failed, given the system is operating nominally. The probability of a true negative is defined to be a classifier declaring the system is operating nominally, given the system is operating

nominally. The probability of a false negative is defined to be a classifier declaring the system is operating nominally, given the system has failed. These probabilities are defined mathematically below. To simplify the notation, the probability of classifier A_θ providing a correct positive reading will be denoted as $P_{TP}^{A} = P_{TP}(A_\theta)$. The parameter θ is suppressed in this new expression. The probability of classifier B_ϕ providing a correct positive reading will be denoted as, $P_{TP}^{B} = P_{TP}(B_\phi)$, and so forth. Similarly, the parameter ϕ is suppressed in this new expression. More rigorously, the definitions for the classifier A_θ are

$$P_{TP}^{A} = \Pr((A_\theta(x) \in L_f | x \in \mathbf{X}_f) \tag{5-2}$$

$$P_{FP}^{A} = \Pr((A_\theta(x) \in L_f | x \in \mathbf{X}_n) \tag{5-3}$$

$$P_{TN}^{A} = \Pr((A_\theta(x) \in L_n | x \in \mathbf{X}_n) \tag{5-4}$$

$$P_{FN}^{A} = \Pr((A_\theta(x) \in L_n | x \in \mathbf{X}_f) \tag{5-5}$$

The definitions for classifier B_ϕ are obtained by replacing A with B and θ with ϕ in equations 5-2 through 5-5.

The following table summarizes these eight conditional probabilities as measures of distinct system events. Again, the classifiers are assumed independent.

Table 5-2. Conditional probability table for one system component and two classifiers

Classifier Report $C_{\theta,\phi} = (A_\theta, B_\phi)$ True State	F, F	F, N	N, F	N, N
Nominal	$P_{FP}^{A}\ P_{FP}^{B}$	$P_{FP}^{A}\ P_{TN}^{B}$	$P_{TN}^{A}\ P_{FP}^{B}$	$P_{TN}^{A}\ P_{TN}^{B}$
Failed	$P_{TP}^{A}\ P_{TP}^{B}$	$P_{TP}^{A}\ P_{FN}^{B}$	$P_{FN}^{A}\ P_{TP}^{B}$	$P_{FN}^{A}\ P_{FN}^{B}$

Table 5-2 shows the conditional probability for each possible event, where the classifier's responses are conditioned on the subcomponent's true state.

The joint probability table in Table 5-3 lists the possible outcomes as disjoint events. The general formulation is

$$\Pr(C_{\theta,\phi}(x) \in (L_i \times L_j) \cap (x \in \mathbf{X}_k)) \tag{5-6}$$

$$= \Pr((A_\theta(x), B_\phi(x)) \in (L_i \times L_j) \mid (x \in \mathbf{X}_k)) \Pr(x \in \mathbf{X}_k) \tag{5-7}$$

$$= \Pr(A_\theta(x) \in L_i \mid (x \in \mathbf{X}_k)\ \Pr(B_\phi(x)) \in L_j \mid (x \in \mathbf{X}_k)\ \Pr(x \in \mathbf{X}_k) \tag{5-8}$$

where $i, j, k \in \{f, n\}$.

Table 5-3. Joint probability table for one system component and two sensors

Classifier Report $C_{\theta,\phi} = (A_\theta, B_\phi)$ True State	F, F	F, N	N, F	N, N
Nominal	$P_{FP}^{A}\ P_{FP}^{B}\, p_n$	$P_{FP}^{A}\ P_{TN}^{B}\, p_n$	$P_{TN}^{A}\ P_{FP}^{B}\, p_n$	$P_{TN}^{A}\ P_{TN}^{B}\, p_n$
Failed	$P_{TP}^{A}\ P_{TP}^{B}\, p_f$	$P_{TP}^{A}\ P_{FN}^{B}\, p_f$	$P_{FN}^{A}\ P_{TP}^{B}\, p_f$	$P_{FN}^{A}\ P_{FN}^{B}\, p_f$

Table 5-3 shows the probability of occurrence for each possible event as a product of individual probabilities. The events in this table are mutually exclusive and collectively exhaustive. The first column lists the two possible states of the system, nominal or failed. The top row lists the four different aggregate classifier reports. An "F" means the classifier has reported a failed condition. An "N" means the classifier has reported a nominal condition. The reports are listed at the top of each column as 'classifier A_θ report', 'classifier B_ϕ report'. For example, the third column lists the possible outcomes if A_θ reports a failed condition, and B_ϕ reports a nominal condition.

As an example, the entry in the third row and the third column denotes the specific event where A_θ indicates failed operation and B_ϕ indicates nominal operation, and the system has failed. Mathematically, the expression is:

$$P_{TN}^{A}\ P_{FN}^{B}\ p_f = \Pr(A_\theta(x) \in L_f \mid (x \in \mathbf{X}_f)\ \Pr(B_\phi(x)) \in L_n \mid (x \in \mathbf{X}_f)\ \Pr(x \in \mathbf{X}_f) \qquad (5\text{-}9)$$

The ROC curves for each classifier consist of a set of points where a probability of true positive value (ordinate) is specified for each probability of false positive value (abscissa). The *within* fusion methodology uses these coordinate pairs, at common set points along the abscissa, to create the new ROC curve. The mathematical method used to combine the abscissas and ordinates into a new point is described below.

The pair of points used to develop the methodology will be denoted as (P_{FP}^{A} , P_{TP}^{A}) and (P_{FP}^{B} , P_{TP}^{B}), following the notation from Table 5-3. The point resulting from this fusion process will be labeled (P_{FP}^{C} , P_{TP}^{C}). The probability of false positive for the combined

classifier $C_{\theta,\phi}$ is the probability that $C_{\theta,\phi}$ declares a failure, given that the system is operating nominally. This classifier will declare a failure in three cases: if either A_θ, B_ϕ, or both, declare a failure. Again, this is the "logical or" failure rule. Note that

$$P_{FP}^{C} = 1 - P_{TN}^{C} \tag{5-10}$$

This suggests the following formulation using the probability structure suggested in Table 5-3. The definition of true negative is the declaration of nominal system operation, given the system is operating nominally. Note that

$$P_{TN}^{D} = \Pr(D_{\theta,\phi}(x) \in L_n \mid (x \in \mathbf{X}_n)) \tag{5-11}$$

$$= \Pr((A_\theta(x) \vee B_\phi(x)) \in L_n \mid (x \in \mathbf{X}_n)) \tag{5-12}$$

$$P_{TN}^{C} = \Pr((A_\theta(x) \in L_n) \cap (B_\phi(x) \in L_n) \mid (x \in \mathbf{X}_n)) \tag{5-13}$$

$$\frac{\Pr(A_\theta(x) \in L_n \mid (x \in \mathbf{X}_n)) \cap \Pr(B_\phi(x)) \in L_n \cap (x \in \mathbf{X}_n))}{\Pr(x \in \mathbf{X}_n)} \tag{5-14}$$

$$= [\, P_{TN}^{A} \,][\, P_{TN}^{B} \,] \tag{5-15}$$

as is evident from Table 5-3

$$= [1 - P_{FP}^{A}\,]\,[1 - P_{FP}^{B}\,]. \tag{5-16}$$

Finishing, note that

$$P_{FP}^{C} = 1 - [1 - P_{FP}^{A}\,]\,[1 - P_{FP}^{B}\,] \tag{5-17}$$

$$P_{FP}^{C} = [\, P_{FP}^{A} + P_{FP}^{B} - P_{FP}^{A}\, P_{FP}^{B}\,]. \tag{5-18}$$

The corresponding true positive values, for the identical probability of false positive values on each ROC curve, are combined in the same fashion. The preceding derivation is repeated below with appropriate changes in notation.

$$P_{TP}^{C} = 1 - P_{FN}^{C} \tag{5-19}$$

$$P_{FN}^{D} = \Pr(D_{\theta,\phi}(x) \in L_n \mid (x \in \mathbf{X}_f)) \tag{5-20}$$

$$= \Pr((A_\theta(x) \vee B_\phi(x)) \in L_n \mid (x \in \mathbf{X}_f)) \tag{5-21}$$

$$P_{FN}^{C} = \frac{\Pr((A_\theta(x) \in L_n) \cap (B_\phi(x) \in L_n) \cap (x \in \mathbf{X}_f))}{\Pr(x \in \mathbf{X}_n)} \tag{5-22}$$

$$\Pr(A_\theta(x) \in L_n \mid (x \in \mathbf{X}_f)) \cap \Pr(B_\phi(x)) \in L_n \mid (x \in \mathbf{X}_f)) \tag{5-23}$$

$$= [\, P_{FN}^{A} \,][\, P_{FN}^{B} \,] \tag{5-24}$$

as is evident from Table 5-3'

$$= [1 - P_{TP}^{A}\,]\,[1 - P_{TP}^{B}\,] \tag{5-25}$$

$$P_{TP}^{C} = 1 - [1 - P_{TP}^{A}\,]\,[1 - P_{TP}^{B}\,]. \tag{5-26}$$

As expected, the formula is

$$P_{TP}^{C} = [\, P_{TP}^{A} + P_{TP}^{B} - P_{TP}^{A} P_{TP}^{B}\,]. \tag{5-27}$$

The point on this fused ROC curve is given by

$$(P_{FP}^{C},\, P_{TP}^{C}) = (P_{FP}^{A} + P_{FP}^{B} - P_{FP}^{A} P_{FP}^{B},\, P_{TP}^{A} + P_{TP}^{B} - P_{TP}^{A} P_{TP}^{B}). \tag{5-28}$$

Again, these results assume the classifiers A and B are independent in their measurements, and that their respective operating points are set *a priori*. This is not

likely to be the case in a real system. This *within* fusion rule is therefore a weak upper bound for the fused ROC curve, C. This is explored further in section 5.3.3.

This *within* fusion method allows for the combination of any number of classifiers. Once the ROC curves associated with the classifiers for two sensors have been combined into a single ROC curve, this single curve can be combined with another ROC curve associated with the classifier for another sensor. This iterative process continues until all the classifiers associated with the sensors on a particular subcomponent are represented by a single ROC curve. Using a similar iterative process, any number of these *within* ROC curves may be combined to form an *across* ROC curve, and so forth.

As an example of the *within* fusion rule, consider a critical subcomponent with two sensors and two classifiers. Let the ROC curve for classifier A be given by $y_1 = x^{0.1}$, and let the ROC curve for classifier B be given by $y_2 = \left(\left(\frac{2}{\pi}\right)\arcsin(x)\right)^{1/6}$. These are reasonable choices for ROC curve models because like ROC curves, they begin at the origin and end at the point (1, 1). Also, these curves are a reasonable estimate for actual classifier performance.

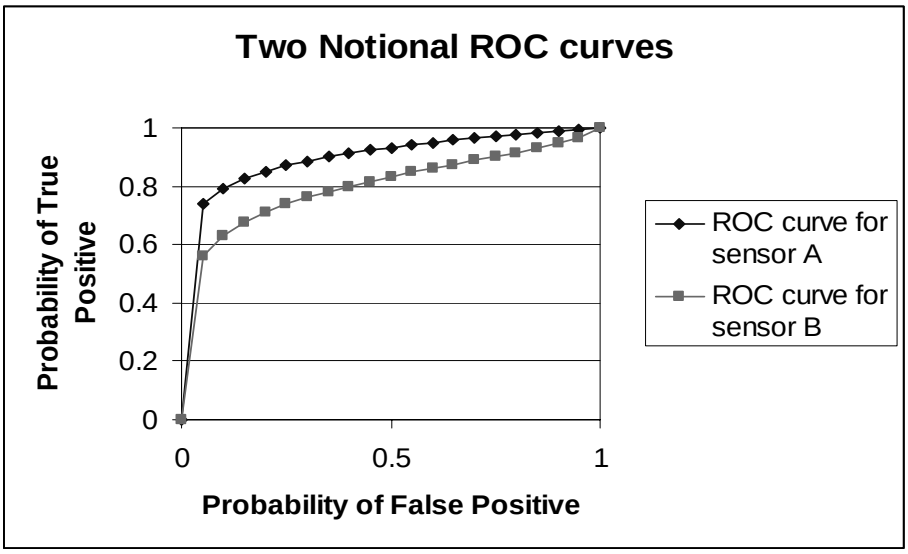

Figure 5-5. Two notional ROC curves

Figure 5-5 shows the two notional ROC curves. Notice that the P_{TP} values for classifier A exceed those for classifier B at every P_{FP} value. Classifier A is said to *dominate* classifier B.

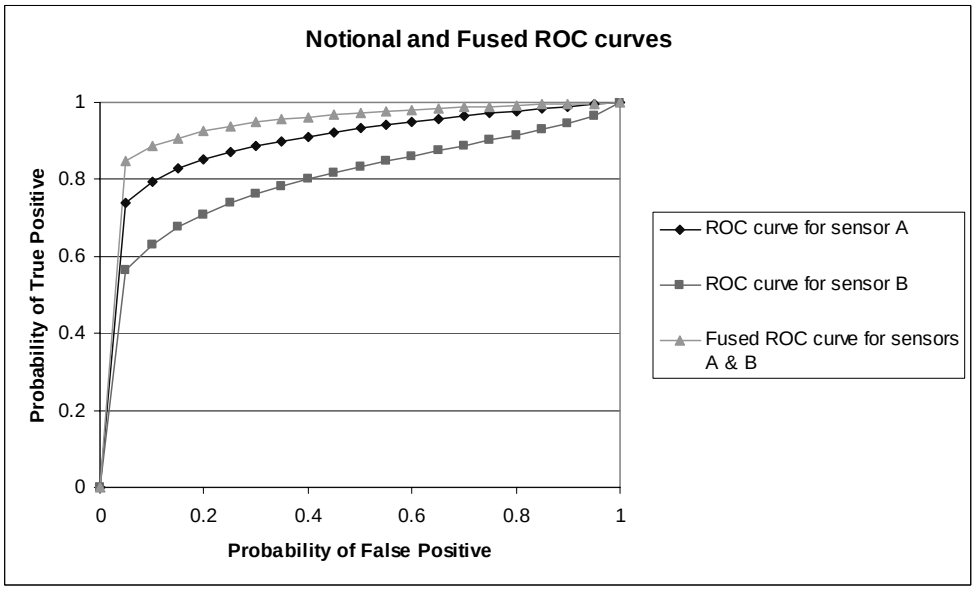

Figure 5-6. Graph of the two notional and fused ROC curves

Figure 5-6 shows the two notional ROC curves and the *within* ROC curve. Note that the *within* curve dominates both of the other curves over all of the operating range. The values for the *within* ROC curve have been linearly interpolated from the values obtained from the within fusion process.

Consider another example where one of the notional ROC curves is significantly dominated by the other curve. Let the ROC curve for classifier C be given by $y_1 = \tanh(4x)$, and let the ROC curve for classifier D be given by $y_2 = x^{0.13}$.

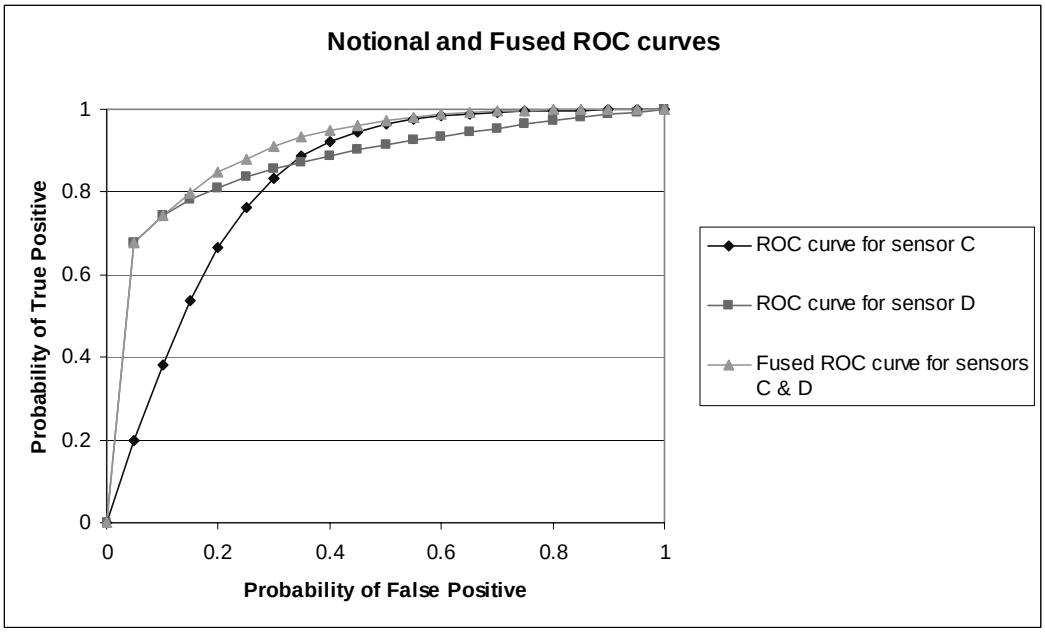

Figure 5-7. Graph of two more notional and fused ROC curves

Figure 5-7 shows the comparison of the *within* ROC curve to the original ROC curves. Again, despite the disparity in the two original curves, the *within* curve still dominates both other curves over all of the operating range. It seems from these examples the

within ROC curve will always equal or dominate each of the ROC curves of the individual classifiers which contributed to it. To demonstrate this in general, consider two notional classifiers that are independent. Let these classifiers have associated probabilities of true positive of p_1 and p_2 at any given probability of false positive value. Without loss of generality, let $p_1 \geq p_2$, and recall that $p_1, p_2 \in [0,1]$. Consider the quantity $p_2(1-p_1)$; this value is clearly greater than or equal to 0. Since $p_2(1-p_1) \geq 0$, adding p_1 to both sides gives

$$p_1 + p_2(1-p_1) \geq p_1 \tag{5-29}$$

or

$$p_1 + p_2 - p_1p_2 \geq p_1 \tag{5-30}$$

Equation 5-30 shows that the probability of true positive value for the *within* ROC curve generated from these two independent classifiers will equal or exceed the probability of true positive value of the individual classifier.

5.2.2 Across Fusion

The *across* fusion methodology, as previously stated, addresses the combination of the *within* ROC curves, *when the classifiers are independent*. It also addresses the combination of *across* ROC curves. The essential system difference between the *across* fusion technique and the *within* fusion technique is that the *within* fusion technique only deals with classifiers on one critical subcomponent. The *across* fusion technique focuses on combining ROC curves from at least two different system parts (subcomponents and components).

This methodology is based on a monograph by Oxley and Bauer [63]. In this monograph, Oxley and Bauer use a 'logical or' rule to combine two ROC curves and produce a third ROC curve. Their underlying assumptions about this situation are summarized below.

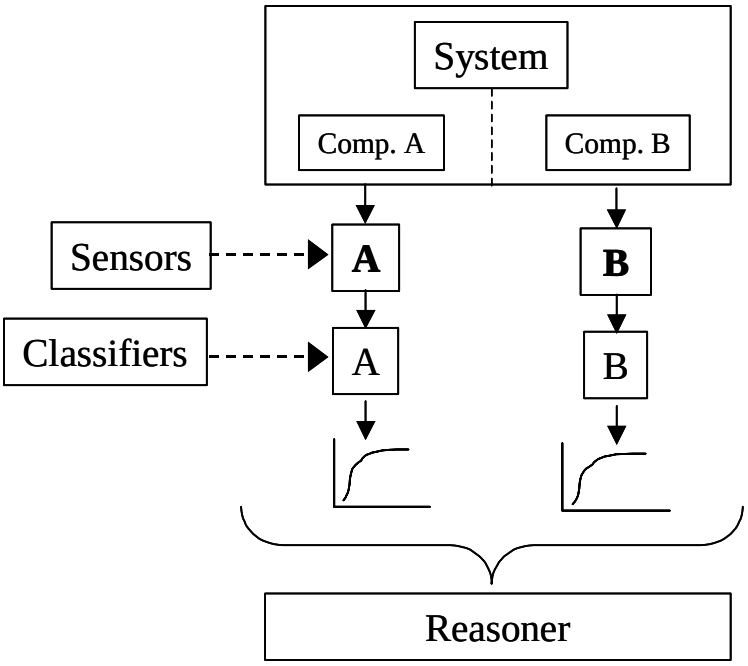

Figure 5-8. Notional prognostics diagram with a two component system and two sensors

Figure 5-8 shows a diagram that illustrates the notional system used for the fusion technique presented in [63]. The system, represented by the large box at the top of the figure, contains two components. Each component sends data to a sensor, which records this information and sends it to a classifier. The classifier uses the sensor data to report the current condition of the component and the overall system. The sensors are assumed to operate independently of each other, as are the system components.

Based on this figure, conditional probabilities are defined below. The labeling convention conditions the classifier output on the actual system data.

P(classifier declares failure|component j is actually failed) = P_{TP}^{j}, j ∈ {A, B}

P(classifier declares failure|component j is actually nominal) = P_{TP}^{j}, j ∈ {A, B}

P(classifier declares nominal|component j is actually nominal) = P_{TP}^{j}, j ∈ {A, B}

P(classifier declares nominal|component j is actually failed) = P_{TP}^{j}, j ∈ {A, B}

In Table 5-4, the first entry in the "True State" column refers to the true state coming from "component A". The second entry column refers to the true state of "component B". The first entry in the "Classifier Reports" row refers to the report from the classifier based on data from sensor A. The second entry refers to the report from the classifier based on data from sensor B. These reports are component specific. The mapping between these system reports and an actual determination of system failure has not yet been specified. However, regardless of the system report, the system has actually failed if there is an "F" in the "True State" column (the middle three rows of the table).

Table 5-4. Conditional probability values [63]

Classifier Reports (A, B) True State	F, F	F, N	N, F	N, N
F, F	$P_{TP}^{A}\ P_{TP}^{B}$	$P_{FP}^{A}\ P_{TN}^{B}$	$P_{FN}^{A}\ P_{TP}^{B}$	$P_{FN}^{A}\ P_{FN}^{B}$
F, N	$P_{TP}^{A}\ P_{FP}^{B}$	$P_{TP}^{A}\ P_{TN}^{B}$	$P_{FN}^{A}\ P_{FP}^{B}$	$P_{FN}^{A}\ P_{TN}^{B}$
N, F	$P_{FP}^{A}\ P_{TP}^{B}$	$P_{FP}^{A}\ P_{FN}^{B}$	$P_{FN}^{A}\ P_{TP}^{B}$	$P_{FN}^{A}\ P_{FN}^{B}$
N, N	$P_{FP}^{A}\ P_{FP}^{B}$	$P_{FP}^{A}\ P_{TN}^{B}$	$P_{TN}^{A}\ P_{FP}^{B}$	$P_{TN}^{A}\ P_{TN}^{B}$

Table 5-4 shows the conditional probability values for the two classifiers A and B in the presence of both failure and nominal system data. For instance, the first entry in the cell in the third row and fourth column, P_{FN}^{A}, represents the probability that the classifier reports nominal functionality of component A given component A has failed. The second entry in the cell, P_{FP}^{B}, represents the probability that the classifier reports a failure of component B given component B is operating nominally.

The following joint probability table combines these values with the *a priori* probabilities. Again, both of the two failure types are assumed to be independent of each other.

Table 5-5**. Joint probability values [63]**

Classifier Reports (A, B) True State	F, F	F, N	N, F	N, N
F, F	$P_{TP}^{A}\ P_{TP}^{B}\ p_f q_f$	$P_{TP}^{A}\ P_{FN}^{B}\ p_f q_f$	$P_{FN}^{A}\ P_{TP}^{B}\ p_f q_f$	$P_{FN}^{A}\ P_{FN}^{B}\ p_f q_f$
F, N	$P_{TP}^{A}\ P_{FP}^{B}\ p_f q_n$	$P_{TP}^{A}\ P_{TN}^{B}\ p_f q_n$	$P_{FN}^{A}\ P_{FP}^{B}\ p_f q_n$	$P_{FN}^{A}\ P_{TN}^{B}\ p_f q_n$
N, F	$P_{FP}^{A}\ P_{TP}^{B}\ p_n q_f$	$P_{FP}^{A}\ P_{FN}^{B}\ p_n q_f$	$P_{TN}^{A}\ P_{TP}^{B}\ p_n q_f$	$P_{TN}^{A}\ P_{FN}^{B}\ p_n q_f$
N, N	$P_{FP}^{A}\ P_{FP}^{B}\ p_n q_n$	$P_{FP}^{A}\ P_{TN}^{B}\ p_n q_n$	$P_{TN}^{A}\ P_{FP}^{B}\ p_n q_n$	$P_{TN}^{A}\ P_{TN}^{B}\ p_n q_n$

Table 5-5 summarizes these joint probabilities as a series of disjoint events. The third row indicates the actual data shows a failure on component A and no failure on component B. The failure on component A is reflected with the *a priori* probability p_f, and the nominal condition on component B is reflected with the *a priori* probability q_n. These are expected to be small and large probability values, respectively.

In their monograph, Oxley and Bauer [63] use the preceding table to develop an expression for the fused ROC curve for two mechanical system components. Let f_A and f_B represent the two original ROC curves. Also, as before, let p_f be the prior probability of failure of component A, and let q_f, be the prior probability of failure of component B. Let the following relationships hold:

$$\gamma = p_f + q_f - p_f q_f,\ r \in [0, 1],\ s \in [0, r] \tag{5-31}$$

Then the fused ROC curve is given by

$$f_C(r) = \left(\frac{1}{\gamma}\right) - \left(\frac{r(1-\gamma)}{\gamma}\right) - \left(\frac{1}{\gamma}\right)\min_{0\le p\le r}(z) \qquad (5\text{-}32)$$

where

$$z = \left[1-\left(p_f\, f_A(s) + (1-p_f)s\right)\right]\left[1-\left(\left(q_f\, f_B\left(\frac{r-s}{1-s}\right)\right) + (1-q_f)\left(\frac{r-s}{1-s}\right)\right)\right] \qquad (5\text{-}33)$$

Equation 5-37 is the relation used to combine two *within* or *across* ROC curves to produce another *across* ROC curve.

5.2.3 Dependent Sensors

This section develops bounds for the effects of dependent sensors within a given system. Consider a system where the sensors **A** and **B** are completely dependent. This would occur if two sensors were both measuring the same phenomenology on the same component, as they would if the sensors are redundant. In such a system, accurate readings from sensor **B** would match accurate readings from sensor **A** in every possible operating condition. In effect, sensor **B** provides no new information on the condition of the system. Note that this condition does <u>not</u> assume the accuracy of the sensors would be the same, just that their accurate readings would be the same. In this case with completely dependent sensors, the logical decision is to chose the sensor with the better accuracy, and discard the other one. Hence, a lower bound on the fused ROC curve C is the best ROC curve associated with the classifier for one of the two original sensors **A**

and **B**. In passing, it is worth noting that the only time the accuracy of the sensors would be the same is when the sensors are identical, AND have identical operating conditions.

As previously stated, the *within* fusion methodology provides a weak upper bound on the fused ROC curve C. This is because the methodology uses set operating points from both ROC curves to generate the fused ROC curve. This methodology is in contrast to the *across* fusion methodology, which takes a specific operating point from one ROC curve and searches along the entire length of the other ROC curve to choose the best point to obtain the best probability of true positive value. If the *within* fusion methodology had been developed using a similar technique, the fused ROC curve C would be optimal, relative to the classifier thresholds. (This result is found in Oxley and Bauer [63].) This means this optimal *within* ROC curve would have probability of true positive values that are at least equal to the values of the fused *within* ROC curve generated using set operating points, and potentially have a number of values that exceed the values of this fused *within* ROC curve. However, the *within* (and *across*) fusion methodology assumes the sensors that provide data to the classifiers operate independently. This assumption may not always hold, particularly if the sensors on a subcomponent are intended to be redundant. If there is some degree of dependency between the sensors, then the optimal *within* ROC curve will overestimate the actual *within* ROC curve. The fused *within* ROC curve generated using set operating points may also overestimate the actual *within* ROC curve, but to a smaller degree than the actual optimal *within* ROC curve. Hence, the fused *within* ROC curve is used to provide the estimate of the actual *within* ROC curve.

With an established lower bound (the best single ROC curve) and a weak upper bound (the fused *within* ROC curve) for the fused *within* ROC curve C, it is obvious that the actual *within* ROC curve for a system with dependent sensors would lie between these two extremes. Precisely where it would be located depends on the amount of dependency between the two sensors. This amount of dependency may change from one operating condition to another. There also may be a minimum level of dependency which is present in every operating condition. The actual *within* ROC curve is probably best determined through empirical observation of the actual system in question.

5.3 Application to a Two-Component System

In this section, two problems are constructed and solved using the *within* and *across* fusion methods described in this chapter. Additionally, a solution algorithm is presented for solving these problems. Section 5.3.1 presents the across fusion methodology, using a simple system as an example. This simple system has a single component with two subcomponents and two sensors on each subcomponent, as shown in Figure 5-9. Section 5.3.2 presents the general solution algorithm for solving these types of problems. Section 5.3.3 uses the solution algorithm presented in section 5.3.2 to solve a second, more complicated problem. This second problem expands the first problem by adding a second component to the system. This new component also has two subcomponents, but with two sensors on the first subcomponent and three sensors on the second subcomponent. Section 5.3.4 presents an excursion where the per component budget constraint is relaxed to apply only to the overall system.

5.3.1 A Single Component Problem

This section presents a simple problem to demonstrate the application of the two fusion methods. The notional system used for this problem has a single component and two subcomponents.

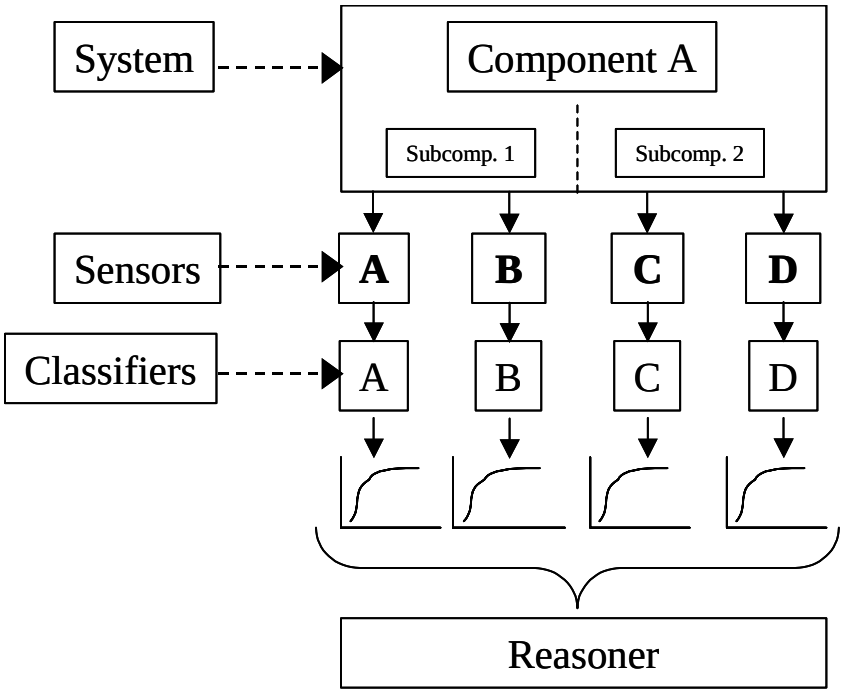

Figure 5-9. Figure 4-2 reproduced for clarity

Figure 5-9 shows the design of the simple system which will be used to demonstrate solving the across fusion problem. Solving this problem will require four notional ROC curves (one for each sensor), two fused ROC curves (*within* curves) using the *within* fusion methodology, and ultimately one ROC curve (*across* curve) using the *across* fusion methodology. The objective is for this *across* ROC curve to be the best one

possible. To accomplish this objective will only require three fused ROC curves since there is no budget constraint on the number of sensors that may be considered per subcomponent.

The curves that were used to produce the illustrative examples in section 5.2.1 will be used to solve this problem. As a reminder, the ROC curve for classifier A was given by $y_1 = x^{0.1}$, and the ROC curve for classifier B was given by $y_2 = \left(\left(\frac{2}{\pi}\right)\arcsin(x)\right)^{1/6}$.

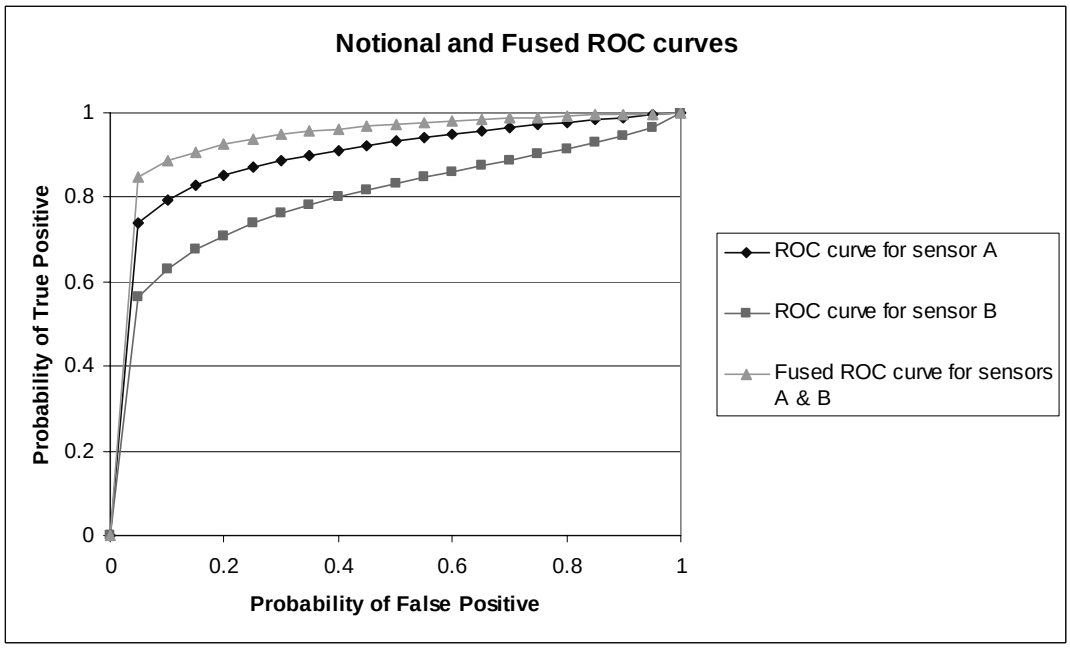

Figure 5-10. Figure 5-6 reproduced for clarity

Figure 5-10 shows these two notional ROC curves and their associated *within* ROC curve.

As before (section 5.2.1), let the ROC curve for classifier C be given by $y_3 = \tanh(4x)$, and let the ROC curve for classifier D be given by $y_4 = x^{0.13}$.

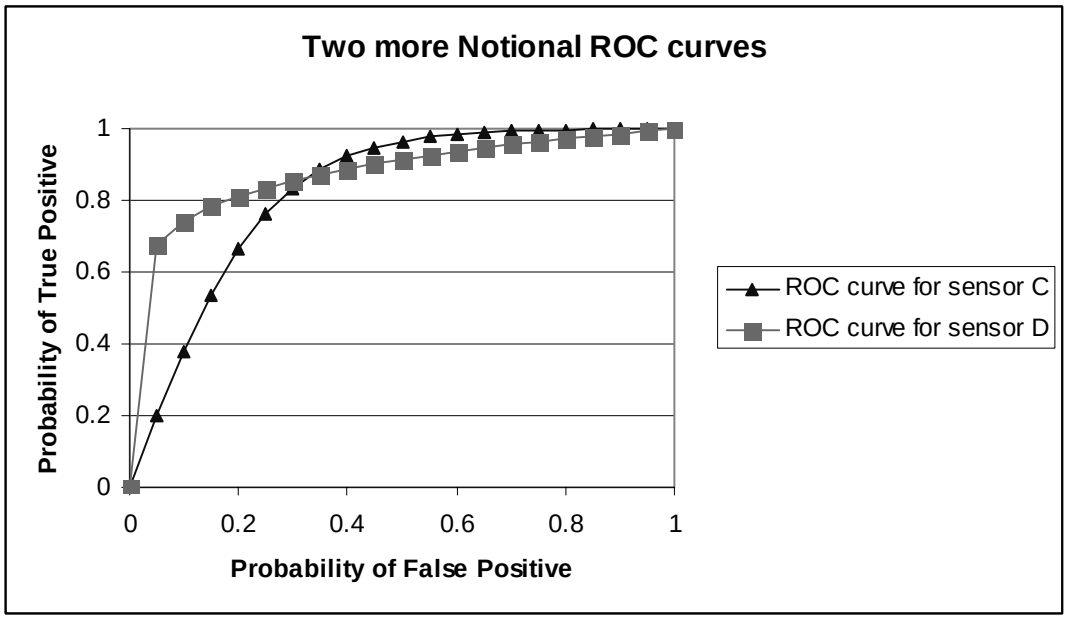

Figure 5-11. Two more notional ROC curves

Figure 5-11 shows these two notional ROC curves. Neither curve is completely dominated by the other.

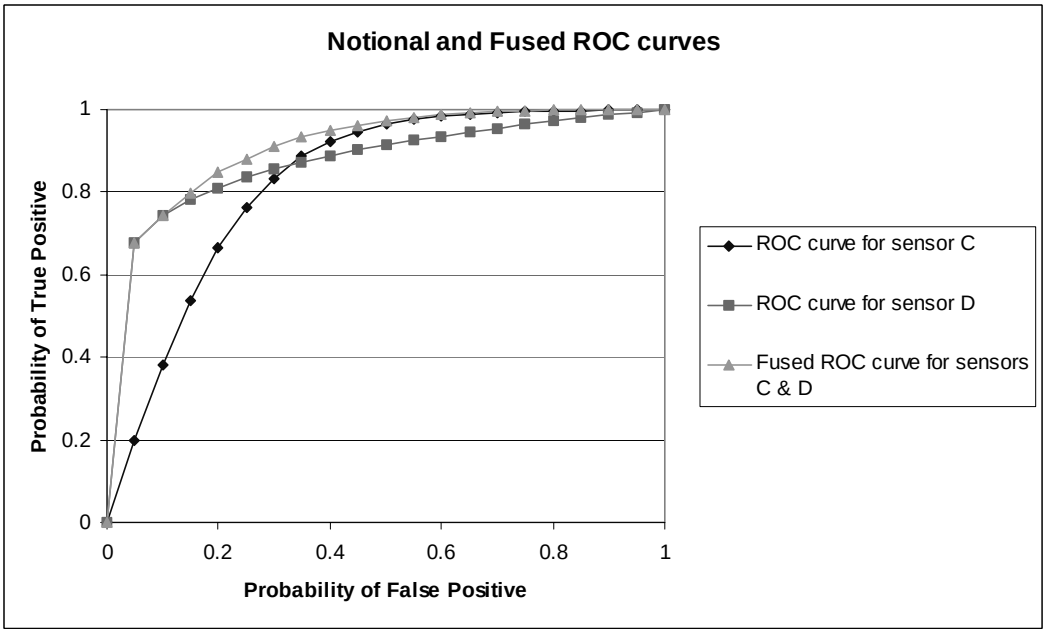

Figure 5-12. Graph of two more notional ROC curves, and the fused curve

Figure 5-12 shows the graph of ROC curves C and D, and their *within* ROC curve.

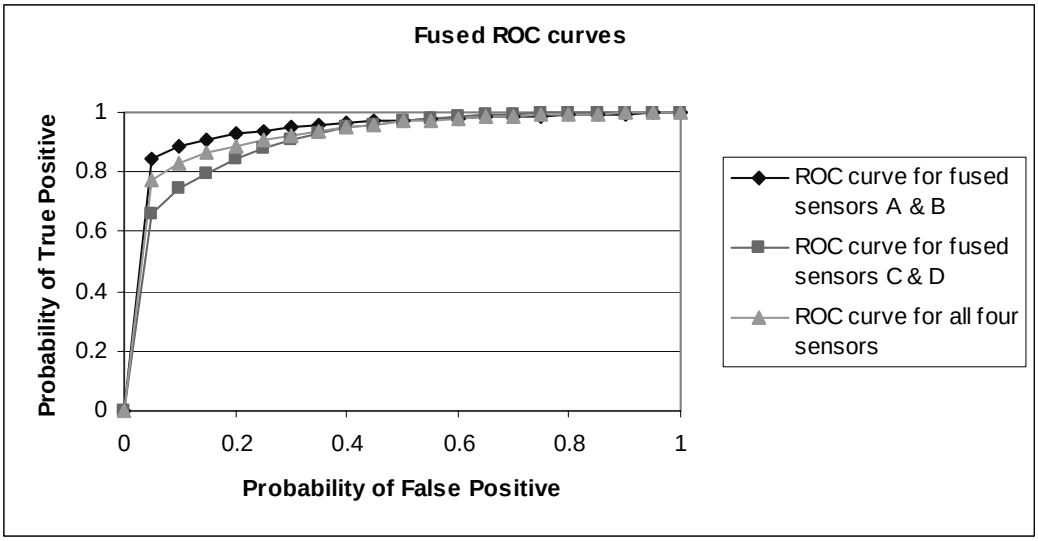

Figure 5-13. The *across* ROC curves for the two classifier pairs, and the *across* ROC curve obtained by fusing all four classifiers using *across* fusion

Figure 5-13 shows the fusion of the two *across* curves into a single *across* ROC curve, using the *across* fusion method. This is now the ROC curve for the component/system. Note that the *across* ROC curve approximately splits the difference between these two curves.

5.3.2 The General Solution Algorithm

This section presents the general solution algorithm for solving these types of problems.

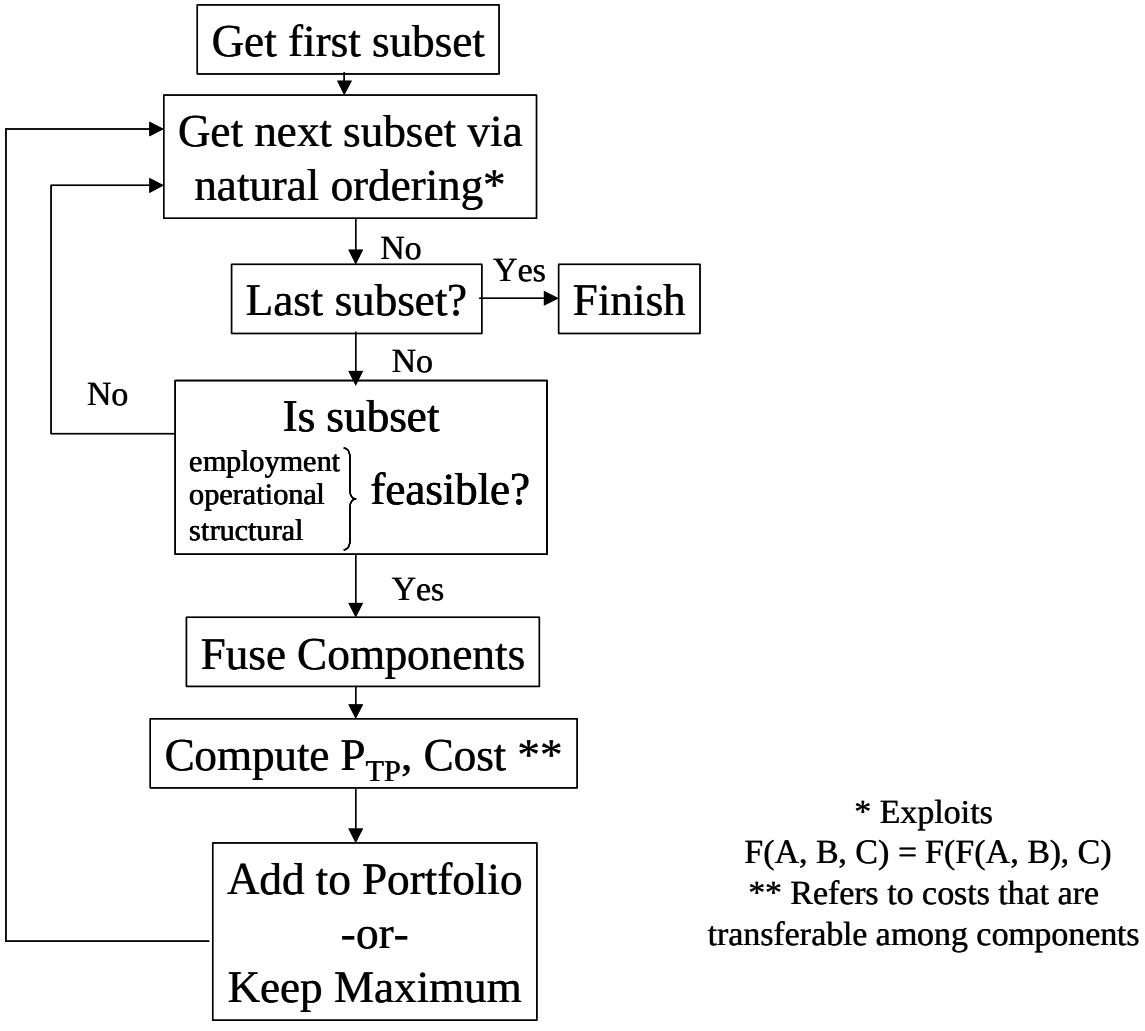

Figure 5-14. Algorithm for problem solution

Figure 5-14 shows the algorithm employed to solve this problem. Each subset for a given component is enumerated. The employment, operational, and structural costs are determined for each subset. If any of these costs exceeds the values specified in the problem constraints, the subset is considered to be infeasible. Infeasible subsets are eliminated from the solution space. The subset fusion (using *within* and *across* techniques) is performed only if a subset is feasible. When the fusion process has been completed, the P_{TP} value is computed and compared to the current maximum P_{TP} value. If the P_{TP} of the current subset exceeds the current maximum, the current subset becomes the new optimal solution. Otherwise, it is discarded and the next subset is checked for feasibility. This process continues until all possible subsets have been considered.

5.3.3 A Two Component Problem

As was demonstrated in section 5.3.1, the *across* fusion method is used to fuse other *across* curves. As previously stated, most systems will typically have more than one component. Figure 4-3 (reproduced below) shows a more complex system.

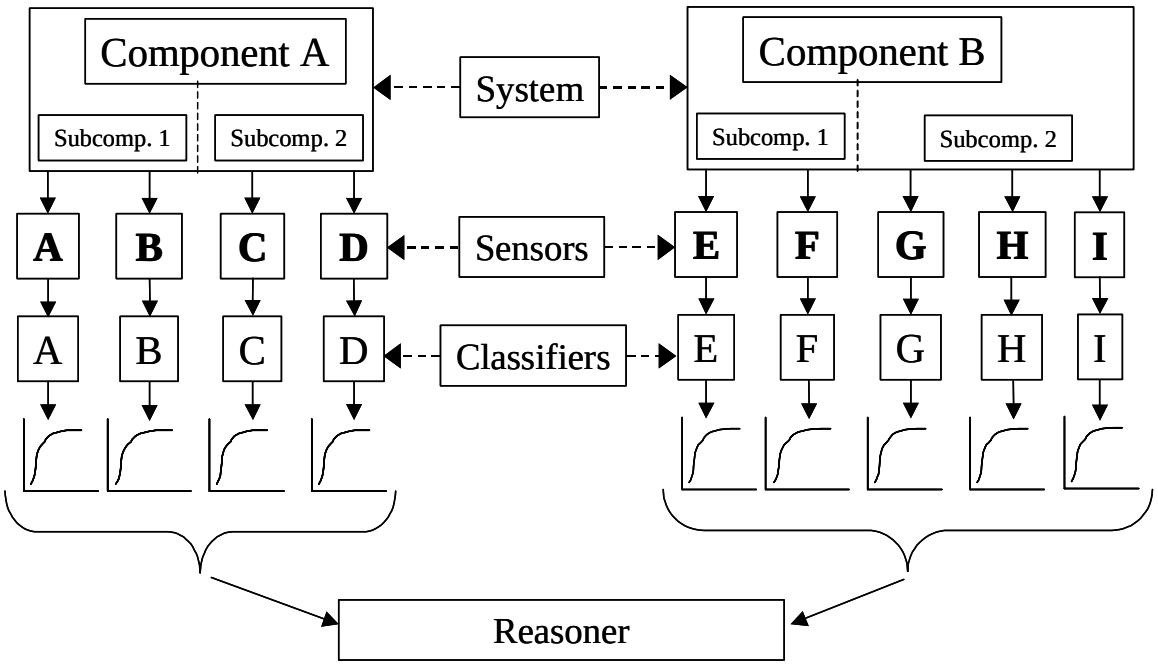

Figure 5-15. Figure 4-3 reproduced for clarity

A mathematical programming framework was presented in chapter 4 to determine the optimal allocation of sensors to subcomponents problem. This is a complex problem because there are an exponential number of subsets that must be considered in light of various structural and operational constraints. Each subset of sensors typically requires multiple ROC curve fusions. These fusions are the most computationally intense calculations encountered in the optimization. Some of the constraints are rapidly evaluated and as such certain sensor subsets are eliminated from consideration. Interestingly, this mathematical programming problem is actually easier to solve given the addition of these easily evaluated structural and operational constraints. Consider the problem posed by Figure 5-15. In this example, it is notionally assumed that one sensor subset for component A and three sensor subsets for component B are not feasible. It is

also assumed that at least one sensor is monitoring each subcomponent, which means the subset of no sensors (the empty set) is excluded from consideration.

Table 5-6. Number of sensor subsets to consider given constraint types

Criteria	Number of sensor combinations
1. Any non-empty sensor combinations	511
2. At least one sensor per component	465
3. At least one sensor per subcomponent	189
4. One sensor per subcomponent and cost feasible	144

Table 5-6 shows the number of sensor ensembles which must be considered given the various constraint types. The number 511 in the first row was obtained by determining the total number of subsets of the nine sensors available for use (512), and subtracting the empty set. The number 465 was computed by determining the total number of subsets of the four sensors available for use on the first component (16), and subtracting the empty set to yield a total of 15. The total number of non-empty subsets on the second component was similarly determined to be 31, and multiplying these two numbers gives 465. The number 189 in the third row was determined using a similar process. The number of non-empty subsets for each subcomponent was determined, and these numbers (3, 3, 3, and 7) were multiplied together to give the number of subsets that have at least one sensor per subcomponent. The number in the fourth row incorporates cost feasibility, so of the nine non-empty subsets for component one that have at least one

sensor per subcomponent, eight are assumed to be cost feasible. Similarly, 18 of the second component subsets are cost feasible, and multiplying these two values gives 144.

Notice that the last row listed in the table, which embodies the mathematical programming approach espoused in chapter 4, corresponds to a 72% reduction in the number of ensembles to be considered. It should be noted that entries 2, 3, and 4 in the table are not consistent with the mathematical programming assumption that each subcomponent requires at least one sensor.

Solving this example will require five additional notional single classifier ROC curves. For simplicity, let the classifiers E through H have the same ROC curves as classifiers A through D. The ROC curves will then be defined as:

$$\text{Classifier A and Classifier E -- } y_1 = x^{0.1},$$

$$\text{Classifier B and Classifier F -- } y_2 = \left(\left(\frac{2}{\pi}\right)\arcsin(x)\right)^{1/6},$$

$$\text{Classifier C and Classifier G -- } y_3 = \tanh(4x),$$

$$\text{and Classifier D and Classifier H -- } y_4 = x^{0.13}.$$

Let the ROC curve for classifier I be given by $y_5 = (1-(x-1)^2)^{0.5}$ (the upper left quadrant of a circle centered at (1, 0)).

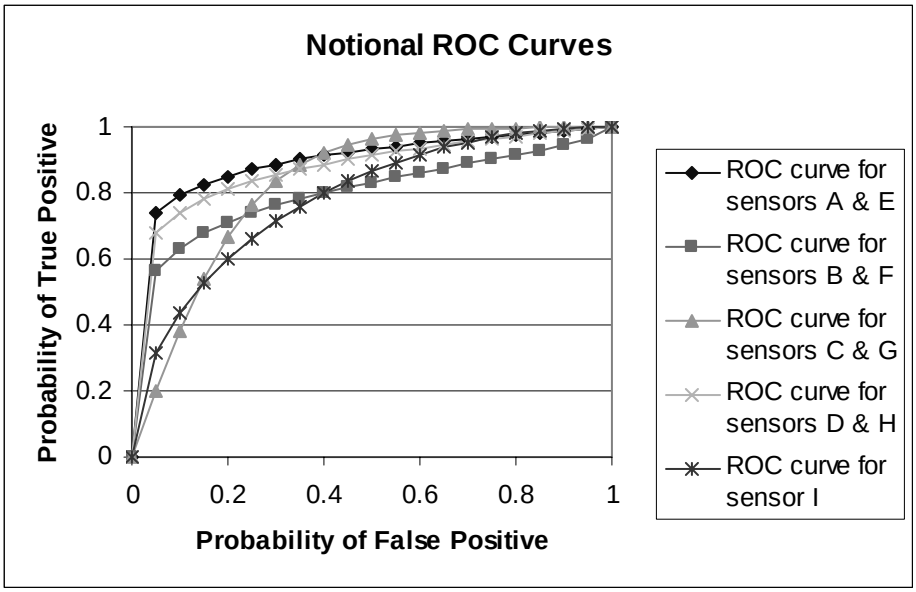

Figure 5-16. Notional ROC curves for all 9 classifiers

Figure 5-16 shows the graphs for all of these notional ROC curves.

The overall sensor budget is divided into a portion for each component. This is because the cost for employing a sensor includes power, weight, space, and other constraints that are not readily transferable to other components. However, some of this cost is the actual monetary cost required to purchase the sensor hardware. Consequently, some portions of the unused budget amounts for a given component could be transferred to other components. This point is addressed in section 5.3.4.

Table 5-7. Sensor costs for the employment cost constraint

Sensors/Component	Per Unit Cost
A, E	45
B, F	30
C, G	25
D, H	35
I	35

Table 5-7 shows the cost for each sensor. The budget for component A is 125 and the budget for component B is 135.. These values incorporate the notional assumption that one sensor combination is infeasible for component A, and three sensor combinations are infeasible for component B. The solution method rapidly determines all the feasible sensor combinations, and then computes the respective ROC curves.

It is of interest to compute the "best" ROC curves (those that possess the largest P_{TP} value among all the ROC curves at a given P_{FP} value) for subsets within the components. It should be noted that in the range $0.0 \leq P_{FP} \leq 0.04$, there are many ensembles which have the same probability of true positive, to four decimal places. However, a unique ensemble is always the "best" ensemble when the probability of false positive value reaches 0.05.

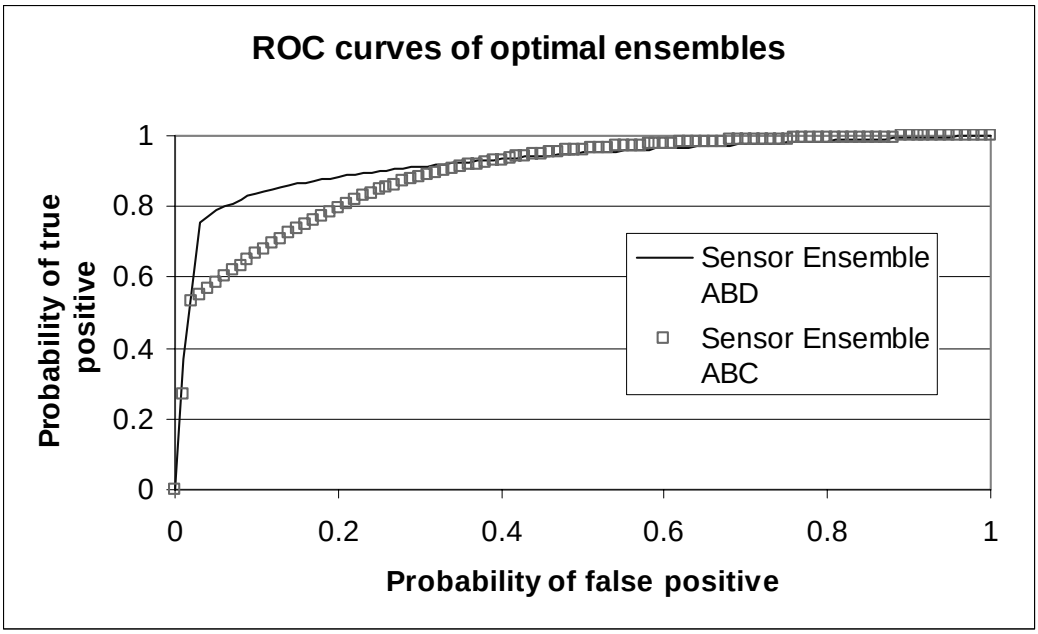

Figure 5-17. The solution for component A

Figure 5-17 shows the solution for component A. Sensor ensemble ABD is the "best" ensemble until the probability of false positive value reaches 0.37, then sensor ensemble ABC is the "best" ensemble. Sensor ensemble ABCD would have been included on this graph if it had been cost feasible.

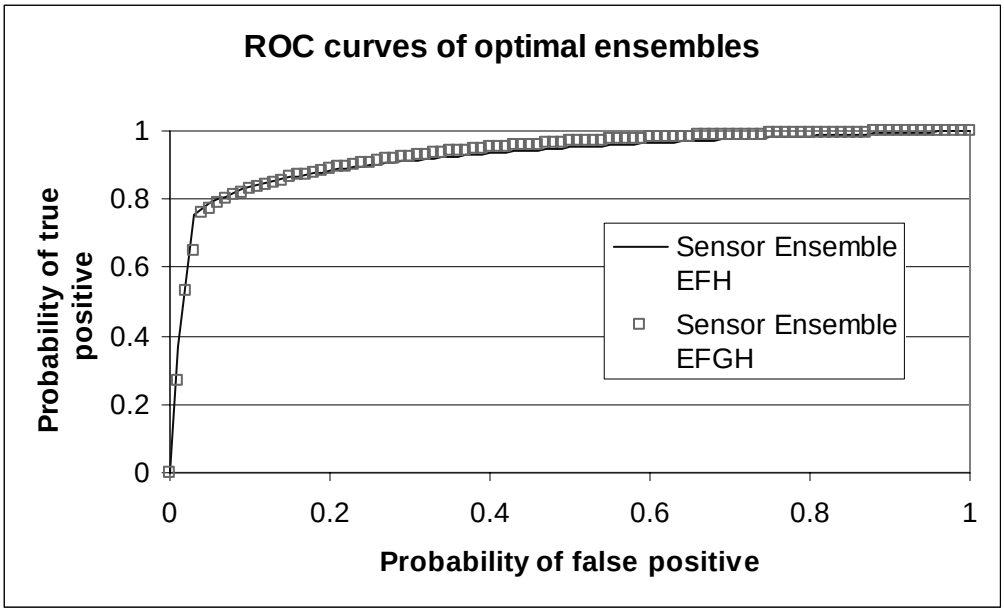

Figure 5-18. The solution for component B

Figure 5-18 shows the optimal ROC curves for component B. Sensor ensemble EFH is the "best" ensemble until the probability of false positive value reaches 0.16, then sensor ensemble EFGH is the "best" ensemble.

The actual solution process is implemented according to the algorithm presented in figure 5.9. Once all the feasible ROC curves have been generated for each component, they are combined using the *across* fusion method. In this example, this means that each of the 8 feasible ROC curves from component A are individually combined with each of the 18 feasible ROC curves from component B. This creates the entire set of feasible ROC curves. Then, for each probability of false positive value, the solution method determines which ROC curve has the best TP value. This usually leads to a collection of a number of ROC curves, as one ROC curve supersedes another as the maximization process

continues. The optimal solution for this problem is given by the four curves shown below.

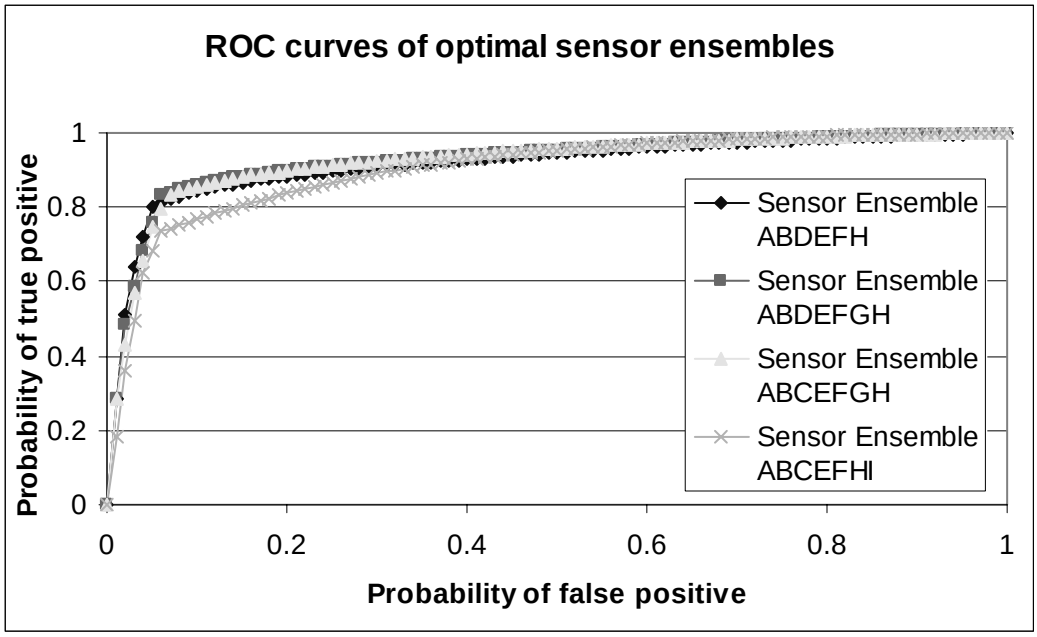

Figure 5-19. The optimal ROC curves for this example

Figure 5-20 shows an enlarged view of the area of the graph where the probability of true positive value is greater than0.8.

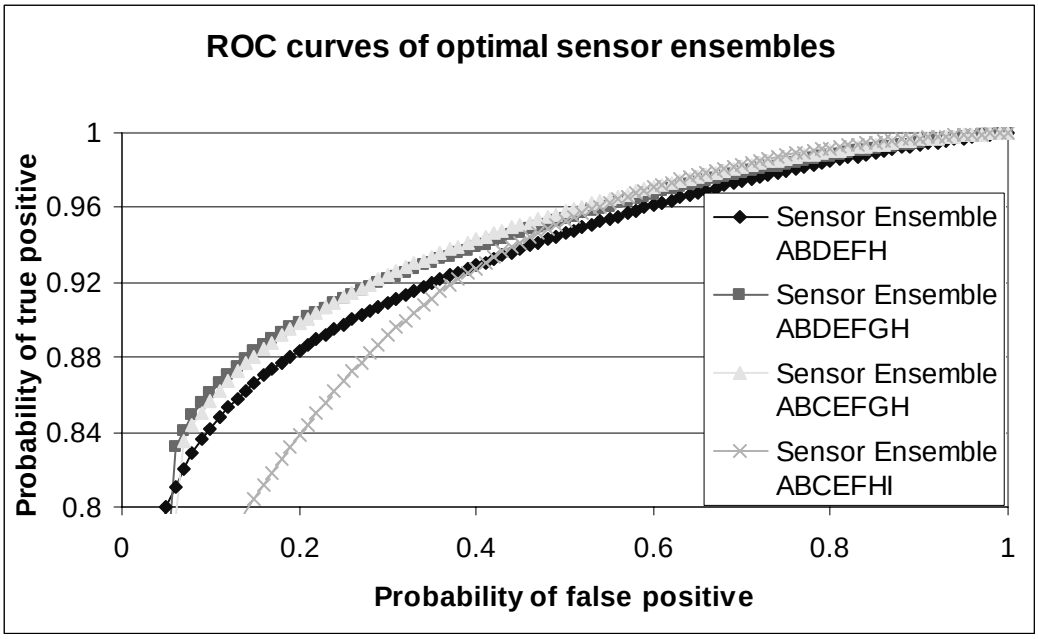

Figure 5-20. A closer view of the optimal ROC curves

Figure 5-19 shows that the sensor ensembles ACDEH, ACDEFH, ACDEFGH, and ABCEFGH are the "best" curves for this notional example. These ensembles will be referred to as sensor ensembles 1-4, respectively, for simplicity. As an example, sensor ensemble 1 (ACDEH) is represented by the diamonds, and dominates the other curves over a small part of the range, from domain values 0.05 through 0.06. At the domain values 0.07 through 0.23, sensor ensemble 2 (ACDEFH) dominates. For the domain values 0.24 through 0.61, sensor ensemble 3 (ACDEFGH) dominates. And sensor ensemble 4 (ABCEFGH) dominates for the remainder of the domain values, 0.62 through 1.00.

Once these best sensor ensembles have been identified, the best overall ensemble is selected. This is done by selecting a maximum allowable value for the probability of

false positive. The ensemble with the largest probability of true positive value at that particular point is chosen as the ensemble to employ on the system. In this example, if the maximum allowable value for the probability of false positive is 0.2, the best sensor ensemble is ACDEFH.

5.3.4 Two Component Problem Excursion

As previously stated, some unused portions of the budget for a particular component are not transferable to other components, in the context of this model. However, excess cost may be transferred between components. This ability to transfer excess cost may require the calculation of new solutions. Assuming the excess budget amounts may be transferred between the two components, the optimal solution changes.

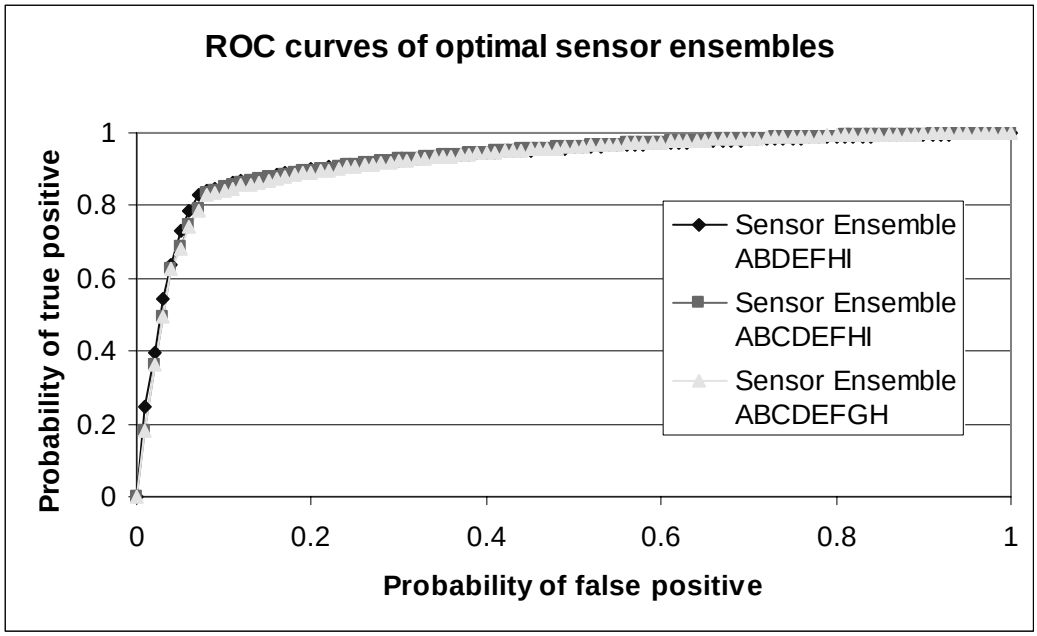

Figure 5-21. The optimal ROC curves if unused budget allocations could be transferred among components

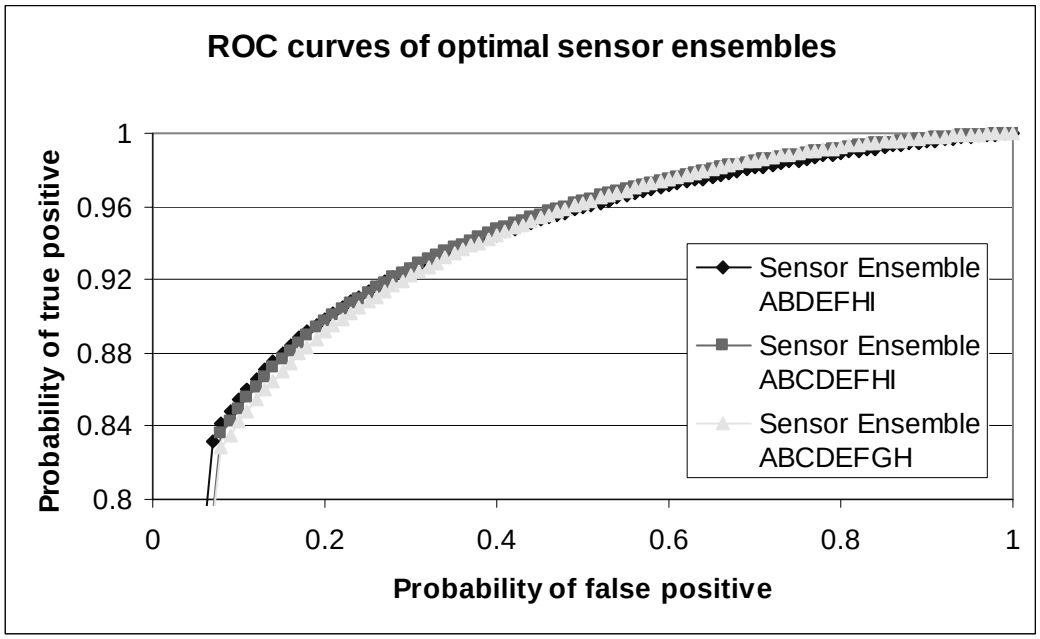

Figure 5-22. A closer view of the optimal ROC curves

Figure 5-21 shows the optimal solution to the problem if the entire budget could be shifted among the two components. The change from sensor ensemble ABDEFHI to sensor ensemble ABCDEFHI occurs at 0.29. The change from sensor ensemble ABCDEFHI to sensor ensemble ABCDEFGH occurs at 0.63. These ROC curves are generally only a few thousandths better than the ROC curves presented in Figure 5-19.

VI. Summary and Recommendations

6.1 Overview

This dissertation research makes contributions in the emerging field of prognostics. This section summarizes these contributions and presents recommendations for future research.

6.2 Theoretical Contributions

A mathematical programming model was developed to optimally allocate sensors and their respective classifiers among system components. The model includes structural, employment cost, and operational cost constraints, allowing this formulation to be tailored for any given system and budget.

System data fusion methods were developed to allow for the combination of information from the classifiers associated with different sensors. Two different types of fusion methods were employed. The first method, called *within* fusion, uses the characteristics of sensors on a single system component to provide an assessment of that component's functionality, and is developed here. The second method, called *across* fusion, combines *within* fusion measures (and other *across* fusion measures) to ultimately provide an assessment of the system's functionality.

A proof was given demonstrating to show that in the absence of noise for independent sensors, adding sensors of any capability to a given sensor ensemble will improve the

ability of the ensemble to accurately determine the system state. This allows for rapid evaluation of points in the solution space, since if all ensembles of a certain size are feasible, all smaller ensembles will have smaller objective function values and can be eliminated from consideration.

A methodology was developed to assess the relative merit of various fusion rules. There are many different methodologies for combining the information from multiple sensors. The method presented for scoring the different methodologies allows for the selection of the best methodology for fusing sensor information, based on the capabilities of the sensors, the relative importance of avoiding false negatives compared to false positives, and the reliability of the system components under consideration.

A proof was given demonstrating that demonstrate that under the conditions of sensor independence and no system "noise", a "logical or" fusion rule is the best methodology for combining sensor information. It also demonstrates that there is no "best" fusion rule for situations which do not meet the conditions required for this proof.

A similar proof was given to show that under the conditions of sensor independence and no system "noise", a "logical and" fusion rule is the best methodology for combining sensor information. It also demonstrates that there is no best fusion rule for situations which do not meet the conditions required for this proof.

6.3 Applied Contributions

A comprehensive literature review was written summarizing research activities associated with applying the science of prognostics to various military and industrial applications. This review includes descriptions of efforts to develop both system-wide and component-part prognostic systems. It also discusses some of the technical challenges that must be overcome in order to successfully implement a prognostics system.

A Prognostics and Health Management system taxonomy was developed to provide a common frame of reference for discussions about prognostics systems. This taxonomy included the definitions of various types of faults, and the expected outputs from a prognostics system.

Sample problems using the mathematical program and the system data fusion methodology were presented and solved to show the application of this methodology. A notional two-component system was constructed with places for notional sensors to be employed. ROC curves were used to approximate the sensors' classification performance. Notional costs were assigned to each sensor, and a problem solution algorithm was developed to ensure the optimal solution was found, while avoiding unnecessary sensor fusion computations.

6.4 Areas for Future Research

This methodology could be employed to perform prognostics functions on real world systems. Data can be collected from a given system of interest. Once sufficient data has been collected analysis of that data should reveal unique data patterns which correspond to different failure states. An appropriate set of classifiers can then be trained to recognize these unique patterns and provide high confidence diagnoses of system problems. The algorithm for optimum sensor allocation from this research can be employed to appropriately deploy sensors on this system and use these classifiers to provide system prognostics.

The prognostic information from the preceding effort could be used to manage operational systems. Once information about the future health of multiple systems is known, that information can be used to proactively schedule maintenance actions, assess population health, determine future mission/production capability rates, and adjust future mission/production schedules. These capabilities have been collectively described as an Autonomic Logistics System (ALS). A possible research effort would involve actually designing an ALS which performed these functions.

System damage generation and prediction mechanisms could be developed. Real system damage data streams are hard to find. The goal of system maintenance is to prevent damage from occurring. Additionally, allowing a system to be destroyed to capture the actual failure data can be prohibitively expensive. In virtually all cases, modeling catastrophic failure paths must be accomplished via analytical models or simulation as

opposed to actual data. The development of a damage generation model would allow for the simulation of catastrophic damage processes for a wide variety of systems. This would provide data for a prognostics system to recognize the early symptoms of catastrophic damage, and allow for preventative action to terminate system operations before the catastrophic failure occurred.

The sensor fusion methodology presented in this dissertation can be applied to other sensor fusion problems. These problems include Automatic Target Recognition, Combat Identification, Battle Damage Assessment, and related battlefield issues. All these issues require a high degree of confidence in the answer, and consequently employ a number of different data streams to ensure the answer provided is as accurate as possible. The fusion methodology presented in this work could be used to combine the different data streams to provide the accurate answer required.

Appendix A. Methodology Comparison

The following table [95] briefly describes and compares 18 different methods, including variations on neural network and fuzzy logic implementations.

The column headings on each page list the different techniques, and the row headings on each page describe a characteristic of interest associated with each technique. The row headings begin with "Nature of the required data" and "Nature of the system". The first heading describes the kind and amount of data required for the particular technique to be useful. The next heading describes the kind of system for which the technique would be most effective. The next row headings are "Time required to generate a solution" and "'Cost' of the solution (in a relative sense)". The "Time" heading provides an assessment of the time required to develop an appropriate solution. The "Cost" heading provides a relative idea of how much time and how many resources would be expended to develop a good solution, compared to other the other techniques. Next are the "Reliability (robustness) of the solution" and "Stability of the solution" headings. The "Reliability" heading describes how accurately model results reflect the true nature of the system. The "Stability" heading describes the technique's consistency over time. The last heading "Changes required if something new is introduced to the underlying system" describes what changes must be made to the model if the underlying system changes. This row provides an idea of how easy or hard it is to maintain an appropriate model using a particular technique. Nearly all systems will be changed (through maintenance, upgrades, etc.) from their original configuration during their operational lifetime, and the

model in use must adapt to these changes to continue to provide accurate system diagnoses. Some modeling techniques are inherently more flexible than others, and this row indicates which techniques are more flexible.

Table 3-1. Summary of Diagnostic/Prognostic Methods [95]

Technique/ Problem Parameters	Fuzzy Logic ("reasoning")	Neural nets ("associative")	Genetic Algorithms (optimization)	Dempster-Schafer (evidential theory)
Nature of the required data	Maybe Incomplete, low-fidelity, small amounts	Lots of data, high fidelity, need to cover dynamic range of system, need large separation between data classes	"Large" solution population, data can be missing, incomplete or discontinuous	Incomplete, low-fidelity, small amounts conflicting
Nature of the system	Non-linear, highly complex (use other techniques if system is "linear", has lots of data)	Non-linear, highly complex (use other techniques if system is "linear")	Non-linear, highly complex (use other techniques if system is "linear")	Missing or conflicting information-need to combine information
Time required to generate a solution	Moderate/Very long if experts' opinions must be gathered	Short, moderate, or long training time depending on size of net	Very long	Short to moderate
"Cost" of the solution (in a relative sense)	Moderate	Moderate to large	Large	Moderate
Reliability (robustness) of the solution	Optimal—depending on initial expert opinions	Optimal, guaranteed to exist (finding it another matter)	Optimal solution not guaranteed to exist/be found	Optimal—depending on initial expert opinions
Stability of the solution	Depends on inherent "disagreement" among the "experts"	Very stable for data on which the network has been trained, unpredictable otherwise	Stable for the initial problem	Depends on the accuracy of the prior probabilities
Changes required if something new is introduced to the underlying system	Requires updating but easy to update	Net requires more training	Requires re-computation of the solution	Requires re-computation of the solution

Technique/ Problem Parameters	Feature Selection/ Extraction	Sensor/ Knowledge/ Information/ Fusion	Rule-Based Expert systems	Fuzzy Clustering/ Classifying
Nature of the required data	Lots of data, high fidelity, needs to cover dynamic range of system	More is better, can handle incomplete, low-fidelity, small amounts	Incomplete, low-fidelity, small amounts	Moderate amount, hi-fidelity, large separation between data classes
Nature of the system	Doesn't matter	Doesn't matter	Doesn't matter	Doesn't matter
Time required to generate a solution	Depends on selection/ development method chosen	Depends on selection/ development method chosen	Long for development, short to run	Moderate
"Cost" of the solution (in a relative sense)	Depends on selection/ development method chosen	Depends on selection/ development method chosen	Large if development must be done, small otherwise	Moderate
Reliability (robustness) of the solution	Depends on initial data	Depends on initial data	Optimal—depending on initial expert opinions	Depends on location and number of clusters
Stability of the solution	Depends on selection/ development method chosen	Depends on selection/ development method chosen	Depends on the accuracy of the heuristics	Depends on location and number of clusters
Changes required if something new is introduced to the underlying system	Process must be repeated	Process must be repeated	Requires re-computation of the solution	Requires re-computation of the solution

Technique/ Problem Parameters	Least Squares Fit	Kalman Filtering	Simulation	Fuzzy wavelet analysis
Nature of the required data	Need clear definition of independent, dependent variables, lots of data	Need accurate system model. "Noise" associated with data must be Gaussian white, must have "confidence" (variance) associated with each data point	Good insight on system functions—math models used to represent system must be accurate	Hi-fidelity, quantity not as important
Nature of the system	Independent variables must be independent, system must be linear with few non-linearities	Linear (non-linear models exist but not covered in class)	Can be of any kind	Non-linear, highly complex (use other techniques if system is "linear")
Time required to generate a solution	Short	Moderate	Depends directly on number of system functions	Long if knowledge base must be created, else moderate
"Cost" of the solution (in a relative sense)	Small	Moderate	Depends directly on time	Moderate
Reliability (robustness) of the solution	Only over the range where data was collected	Optimal for a linear system	Depends on accuracy of math model	Very reliable
Stability of the solution	Very stable	Filter "adapts" to new data-compare to some baseline	Very stable	Very stable
Changes required if something new is introduced to the underlying system	Recomputation required	Only if baseline changes, then change comparison baseline	Math model functions must be altered	Knowledge base must be updated—feature set must be re-validated

Technique/ Problem Parameters	Statistical Change Detection (SCD)	State-Based Feature Recognition	Case-Based Reasoning	Dynamic Neural Nets
Nature of the required data	Accurate data collection, need to know "defect frequencies"	Accurate pattern representation, state machines for each failure mode	Hi-fidelity, sufficient to describe the event	Moderate amount of high-fidelity data
Nature of the system	Can be of any kind producing frequency information	Signal data	Can be of any kind	Can be of any kind
Time required to generate a solution	Moderate	Long if failure modes need to be identified	Short if case library exists/Very long if case library needs to be built	Long if fuzzy sets need to be built
"Cost" of the solution (in a relative sense)	Moderate	Small to Moderate	Large if library needs to be built, small otherwise	Large
Reliability (robustness) of the solution	Optimal change detection point	Very reliable	Reliable, not optimal—has difficulty with novel events	Optimal
Stability of the solution	May be affected by noise, other frequencies not of interest	Very stable	Very stable	Very stable
Changes required if something new is introduced to the underlying system	Ensure frequency set of interest is still correct	Modify appropriate state machines	None—new events will be added to the library as they occur	New rule sets must be generated and WNN must be trained further

Technique/ Problem Parameters	ARMA/ARIMA	Weibull Modeling
Nature of the required data	"Noise" associated with data must be Gaussian white, data collected is evenly spaced and consecutive in time	Actual failure data, hi-fidelity, as much as possible
Nature of the system	Linear	Failure events should follow a Weibull distribution, otherwise this technique is useless
Time required to generate a solution	Moderate	Moderate/Very long if failure data must be collected
"Cost" of the solution (in a relative sense)	Moderate	Moderate
Reliability (robustness) of the solution	Reliable	Somewhat reliable—generated solution will never be correct, but may be "close enough"
Stability of the solution	Stable	Somewhat stable
Changes required if something new is introduced to the underlying system	Baseline operation series must be updated	New failure data must be collected and the curve re-generated

Appendix B. Sensor Ensemble Accuracy

If the assumption is made that each sensor in an ensemble has a positive probability of detecting a problem (a positive value for P_{TP}), then adding such a sensor to an ensemble only increases the value of P_{TP} for the ensemble (ignoring any system noise contribution). The sensors are also assumed to be independent. This assertion is formalized in the following theorem.

First, given a set X of sensors, define the maximum probability of obtaining a true positive by max $P_{TP}(x)$.

Theorem 1: *Let* $T \in \mathbb{A}_n$, $S \in \mathbb{A}_m$, *where* $n < m$, *and* $T \subset S$. *Then* $max_{TP}(T) < max_{TP}(S)$.

Proof:

Since there are n sensors in T, the probability of not detecting a true fault with this sensor suite is

$$P_{nodetect}(T) = \prod_{i=1}^{n} (1 - P_{TP}(S_i)) \qquad (4a\text{-}1)$$

Hence, the probability of detecting any problem is given by

$$\max P_{TP}(T) = 1 - \prod_{i=1}^{n} (1 - P_{TP}(S_i)) \qquad (4a\text{-}2)$$

This expression is the "logical or" fusion rule—if any one of the sensors detects a true fault, the fault is defined to be detected.

Consider a set S containing m= n + k sensors, where $k \in Z^+$. $T = \{s_1, s_2, \ldots, s_n\}$, and $S = \{s_1, s_2, \ldots, s_n, s_{n+1}, \ldots, s_m\}$. Clearly, $T \subset S$. The probability of not detecting a true fault with this sensor suite is

$$P_{nodetect}(S) = \prod_{i=1}^{n+k} (1 - P_{TP}(S_i)) \quad (4a\text{-}3)$$

Hence, the probability of detecting a true fault is

$$\max P_{TP}(S) = 1 - \prod_{i=1}^{n+k} (1 - P_{TP}(S_i)) \quad (4a\text{-}4)$$

Note that equations (4a-1) and (4a-3) have the same first n terms. Notice also that each term in each equation is strictly less than 1. If the terms in common between the expansions in each equation are removed, then

$$\omega = \prod_{i=n+1}^{n+k} (1 - P_{TP}(S_i)) \quad (4a\text{-}5)$$

Since each term in the expansion in equation (4a-5) is less than 1, it is clear that $\omega < 1$. If both sides of equation (4a-5) are multiplied by $\prod_{i=1}^{n} (1 - P_{TP}(S_i))$, the equation becomes

$$\prod_{i=1}^{n} (1 - P_{TP}(S_i)) > \prod_{i=1}^{n+k} (1 - P_{TP}(S_i)) \quad (4a\text{-}6)$$

Multiplying both sides of equation (4a-6) by –1 and then adding 1 to each side yields

$$1 - \prod_{i=1}^{n} (1 - P_{TP}(S_i)) < 1 - \prod_{i=1}^{n+k} (1 - P_{TP}(S_i)) \quad (4a\text{-}7)$$

But the left-hand side of (4a-7) is (4a-2) by definition, and the right-hand side of (4a-7) is (4a-4) by definition, so replacement yields

$$\max P_{TP}(T) < \max P_{TP}(S) \quad (4a\text{-}8)$$

the desired result.

It should be noted that even if a sensor is completely dependent with respect to another sensor in the ensemble, although it will not add to the accuracy of the sensor ensemble, it will add to the ensemble's reliability.

This theorem implies that it is possible to reduce the size of the solution space. The first step is to determine the set of cost-feasible sensors. Each ensemble size is searched for cost feasibility, beginning with ensembles containing only one sensor (cardinality 1). If the entire group of sensor ensembles of a particular size (cardinality n) is cost feasible, the process is repeated on the next ensemble set (cardinality n + 1). If all elements of this next ensemble set (cardinality n + 1) are cost feasible, the previous set (cardinality n) is discarded from the solution space since this larger ensemble set will have a higher value of P_{TP} for any sensor combination, by the previous theorem. However, once a cost infeasible solution is found in a set of cardinality k , all sensor ensembles of cardinality k - 1 and greater are retained for further examination (except for cost infeasible ensembles). All sets of cardinality k – 2 and lower are eliminated from consideration. This reduces the solution space by $\sum_{j=1}^{k-2}\binom{M}{j}$ possible solutions, where M is the total number of sensors available.

Alternatively, the search for cost feasibility could begin at the ensemble containing all the sensors (cardinality M). The search would terminate when all ensembles of a particular size are found to be cost feasible. If this particular size is k – 1 (as above), then all ensembles of cardinality k – 2 and below are eliminated from consideration.

Appendix C. Application of Fusion Rules to the Model

5.A1 Application of Fusion Rules to the Model (Optimality considerations)

This model uses a logical or rule to declare a system failure: if either or both of two classifiers on a subcomponent indicates a failure, the reasoner concludes a failure has occurred and reports a failed condition on the system. Both fusion techniques introduced in the previous section use a logical or rule to combine the ROC curves associated with each classifier to produce a new ROC curve. This section addresses whether or not a "logical or" fusion rule may be considered optimal.

The model used for this assessment is the one Oxley and Bauer [63] used to develop the *across* fusion methodology (see Figure 5-11). If the two systems' *a priori* failure rates are equal ($p_f = q_f$), and the two classifiers' failure and nominal detection capabilities are equal ($P_{TP}(A_\theta) = P_{TN}(A_\theta) = P_{TP}(B_\phi) = P_{TN}(B_\phi)$), then the "logical or" rule is the best fusion rule. If there is even a slight inequality in one of these probabilities, then it is possible to set the values for the other pair of variables so that a fusion rule other than "logical or" is the best fusion rule. However, in the general case, "logical or" is the best fusion rule. The appendix provides a general description of the values of these parameters showing where the transition from "logical or" to a different fusion rule occurs.

This appendix also presents a scoring rule for determining which fusion rule is best. This scoring rule adds the P_{TP} result and the P_{TN}, or $(1 - P_{FP})$, result obtained from a particular

fusion rule, given a set of values for the six parameters specified above. More formally, the equation is

$$\text{Fusion rule score} = w_1 P_{TP} + w_2(1 - P_{FP}) \qquad (5a\text{-}1)$$

where w_1 and w_2 are weights which can be manipulated to reflect the importance of each quantity. Note that $w_1, w_2 \in [0, 1]$ and $w_1 + w_2 = 1$. The relative importance of each of these terms depends on the system for which the prognostic system is being designed.

Once the scoring rule is developed, all eight parameters (p_f, q_f, $P_{TP}(A_\theta)$, $P_{TN}(A_\theta)$, $P_{TP}(B_\phi)$, $P_{TN}(B_\phi)$, w_1, and w_2) are analyzed to determine the optimal fusion rule based on the scoring rule, and where the optimal fusion rule changes, based on varying values of these parameters. As previously stated, the "logical or" fusion rule is the best in most cases. Other fusion rules only become the best fusion rule if the *a priori* probability of failures are relatively high, or the classifier's accuracy is not very good, or one term of the scoring rule is weighted much more heavily than the other term. All of these conditions interact to some extent. The rest of this appendix provides the development and analysis of these ideas.

The system model is developed as before. Certain aspects will be repeated here for clarity and further development. Figure 5-8 (reproduced below) is again the basis for this discussion.

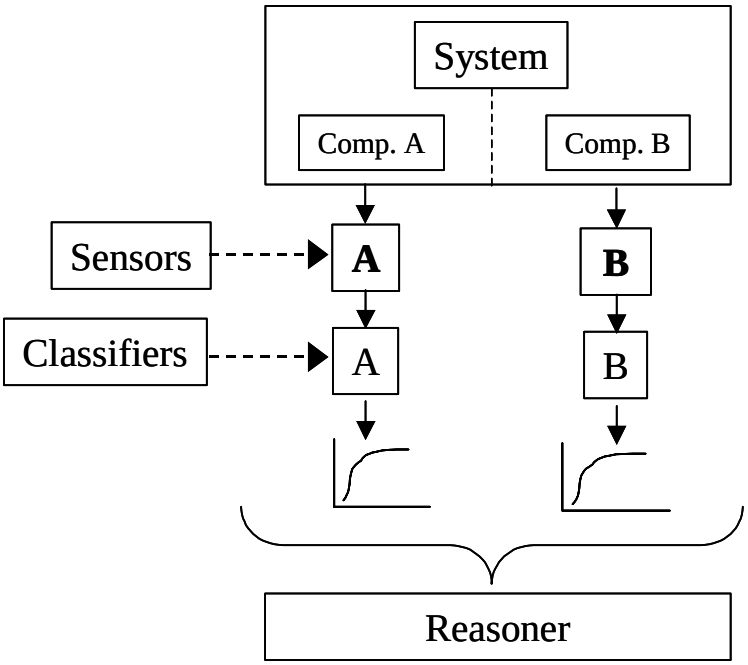

Figure 5a-1. Figure 5-8 reproduced for clarity

The conditional probabilities associated with this model are assigned a notional value as indicated below. The terms "high" and "low" refer to a notional relative probability value for the given condition. The variables "x" and "y", respectively, correspond to those probability values.

P(classifier declares failure|actual failure) = P_{TP}^{A}, P_{TP}^{B} = high = x

P(classifier declares failure|actual nominal) = P_{FP}^{A}, P_{FP}^{B} = low = y

P(classifier declares nominal|actual nominal) = P_{TN}^{A}, P_{TN}^{B} = high = x

P(classifier declares nominal|actual failure) = P_{FN}^{A}, P_{FN}^{B} = low = y

These probability values are assumed to be equal to each other (within the high and low categories) for the sake of the discussion that follows. The joint probability table is reproduced below for clarity. The cells are numbered for ease of reference.

Table 5a-1. Joint probability values [63]

Classifier Reports (A, B) True State	F, F	F, N	N, F	N, N
F, F	1. $P_{TP}^{A}\ P_{TP}^{B}\ p_f q_f$	2. $P_{TP}^{A}\ P_{FN}^{B}\ p_f q_f$	3. $P_{FN}^{A}\ P_{TP}^{B}\ p_f q_f$	4. $P_{FN}^{A}\ P_{FN}^{B}\ p_f q_f$
F, N	5. $P_{TP}^{A}\ P_{FP}^{B}\ p_f q_n$	6. $P_{TP}^{A}\ P_{TN}^{B}\ p_f q_n$	7. $P_{FN}^{A}\ P_{FP}^{B}\ p_f q_n$	8. $P_{FN}^{A}\ P_{TN}^{B}\ p_f q_n$
N, F	9. $P_{FP}^{A}\ P_{TP}^{B}\ p_n q_f$	10. $P_{FP}^{A}\ P_{FN}^{B}\ p_n q_f$	11. $P_{TN}^{A}\ P_{TP}^{B}\ p_n q_f$	12. $P_{TN}^{A}\ P_{FN}^{B}\ p_n q_f$
N, N	13. $P_{FP}^{A}\ P_{FP}^{B}\ p_n q_n$	14. $P_{FP}^{A}\ P_{TN}^{B}\ p_n q_n$	15. $P_{TN}^{A}\ P_{FP}^{B}\ p_n q_n$	16. $P_{TN}^{A}\ P_{TN}^{B}\ p_n q_n$

Again, Table 5a-1 summarizes these joint probabilities as a series of disjoint events. The failure on component A is reflected with the *a priori* probability p_f, and the nominal condition on component B is reflected with the *a priori* probability q_n.

Replacing the this table's contents with the qualitative values of "high" (x) and "low" (y) as previously defined in the table yields an assessment of which combinations of classifier readings and actual data streams would have relatively large likelihoods. Note that $p_n = q_n = x$ and $p_f = q_f = y$.

Table 5a-2. **Table of relative likelihoods**

Classifier Reports (A, B) True State	F, F	F, N	N, F	N, N
F, F	1. x^2y^2	2. xy^3	3. xy^3	4. y^4
F, N	5. x^2y^2	6. $\mathbf{x^3y}$	7. xy^3	8. x^2y^2
N, F	9. x^2y^2	10. xy^3	11. $\mathbf{x^3y}$	12. x^2y^2
N, N	13. x^2y^2	14. $\mathbf{x^3y}$	15. $\mathbf{x^3y}$	16. $\mathbf{x^4}$

Table 5a-2 summarizes the relative likelihoods of these 16 disjoint events. The cell entries in **bold (cells 6, 11, 14, 15, and 16)** indicate a cell with a relatively high likelihood. The cells 1, 6, 11, and 16 (on the main diagonal) indicate an accurate assessment of performance. The cells 4, 7, 10, and 13 (on the anti-diagonal) indicate an inaccurate assessment of performance from both systems. All the other cells have one performance report right and one performance report wrong. This table provides a notional idea of which events are more likely than others.

As can be seen from the table, there are four combinations of readings from the two classifiers:

1. F, F 2. F, N 3. N, F 4. N, N

These combinations of readings can be thought of as four rules for declaring a system failure. If a "logical and" fusion method is chosen, then a system failure would be declared only if the situation described by rule one occurred. This will be referred to specifically as "applying rule one", and more generally as "applying a fusion rule". If a

"logical or" fusion method is chosen, then a system failure would be declared if rules one, two, and three were applied. Since there are four rules, there are fifteen different combinations of rule sets (including the two previously presented) to consider. The results are presented in the following table.

Table 5a-3. Summary of probability values for different fusion rules

Probability Measure Fusion Rule	Cells used to declare a failure (Cells with an actual failure are 1-12)	P_{TP} True Positive (intersection with cells 1-12)	P_{FP} False Positive (intersection with cells 13-16)
1 (logical and)	1, 5, 9, 13	$\frac{3x^2y}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{y^2}{(x + y)^2}$
2	2, 6, 10, 14	$\frac{x^3 + 2xy^2}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{xy}{(x + y)^2}$
3	3, 7, 11, 15	$\frac{x^3 + 2xy^2}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{xy}{(x + y)^2}$
4	4, 8, 12, 16	$\frac{2x^2y + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2}{(x + y)^2}$
1, 2	1, 2, 5, 6, 9, 10, 13, 14	$\frac{x^3 + 3x^2y + 2xy^2}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{xy + y^2}{(x + y)^2}$
1, 3	1, 3, 5, 7, 9, 11, 13, 15	$\frac{x^3 + 3x^2y + 2xy^2}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{xy + y^2}{(x + y)^2}$
1, 4	1, 4, 5, 8, 9, 12, 13, 16	$\frac{5x^2y + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2 + y^2}{(x + y)^2}$
2, 3	2, 3, 6, 7, 10, 11, 14, 15	$\frac{2x^3 + 4xy^2}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{2xy}{(x + y)^2}$
2, 4	2, 4, 6, 8, 10, 12, 14, 16	$\frac{x^3 + 2x^2y + 2xy^2 + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2 + xy}{(x + y)^2}$
3, 4	3, 4, 7, 8, 11, 12, 15, 16	$\frac{x^3 + 2x^2y + 2xy^2 + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2 + xy}{(x + y)^2}$
1, 2, 3 (logical or)	1-3, 5-7, 9-11, 13-15	$\frac{2x^3 + 3x^2y + 4xy^2}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{2xy + y^2}{(x + y)^2}$
1, 2, 4	1-2, 4-6, 8-10, 12-14, 16	$\frac{x^3 + 5x^2y + 2xy^2 + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2 + xy + y^2}{(x + y)^2}$
1, 3, 4	1, 3-5, 7-9, 11-13, 15-16	$\frac{x^3 + 5x^2y + 2xy^2 + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2 + xy + y^2}{(x + y)^2}$
2, 3, 4	2-4, 6-8, 10-12, 14-16	$\frac{2x^3 + 2x^2y + 4xy^2 + y^3}{(2x^3 + 5x^2y + 4xy^2 + y^3)}$	$\frac{x^2 + 2xy}{(x + y)^2}$
1, 2, 3, 4	1-16	1	1

Since it is hard to see from Table 5a-3 which rules have high and low probability values, the following section provides an example with specific values.

5.5 Scoring Rule

Table 5a-4. Fusion rule probability values for a specific case

Probability Measure	PTP (overlap with cells 1-12)	PFP (cells 13-16 over all in formul.)	Score:
Fusion Rule			
1	0.960888119	0.0001	1.960788119
2	0.019408911	0.0099	1.009508911
3	0.019408911	0.0099	1.009508911
4	0.000294059	0.9801	0.020194059
1,2	0.98029703	0.01	1.97029703
1,3	0.98029703	0.01	1.97029703
1,4	0.961182178	0.9802	0.980982178
2,3	0.038817822	0.0198	1.019017822
2,4	0.01970297	0.99	0.02970297
3,4	0.01970297	0.99	0.02970297
1,2,3	0.999705941	0.0199	1.979805941
1,2,4	0.980591089	0.9901	0.990491089
1,3,4	0.980591089	0.9901	0.990491089
2,3,4	0.039111881	0.9999	0.039211881
1,2,3,4	1	1	1

Table 5a-4 shows the values that would be obtained if the following substitutions were made: $p_f = q_f = P_{FP} = P_{FN} = .01$, $P_{TP} = P_{TN} = .99$.

In this table, there is also a column titled "Score". Determining the "best" fusion rule is done initially by selecting the fusion rule which provides the highest P_{TP} and the lowest P_{FP} (highest P_{TN}). The formula to determine the fusion rule "score" is:

$$\text{Fusion rule score} = \{P_{TP} + (1 - P_{FP})\} \text{ or } \{P_{TP} + P_{TN}\} \qquad (5a\text{-}2)$$

The scoring rule was selected to maximize the benefit obtained from a particular fusion rule combination. In this context, the best results from the reasoner are true negatives and true positives. The best fusion rule combination is defined to be the one that provides the highest probability of true positive and the highest probability of true negatives (alternatively, the smallest probability of false positive). The fusion rule that has the highest score for the selected values of p_f, q_f, P_{FP}, P_{FN}, P_{TP}, and P_{TN} is the "logical or" fusion rule. These six parameters are used to develop the notion of an "optimal fusion rule" in the following section.

5.A3 Optimal Fusion Rule Analysis

This result leads to the question of which rule, if any, is optimal, given the set of six inputs p_f, q_f, P_{FP}, P_{FN}, P_{TP}, and P_{TN}. (It should be noted that P_{TP} and P_{TN} determine the values of P_{FP} and P_{FN}.) To answer this question, the following assumptions are made. The classifiers are assumed to be independent of each other. The *a priori* component probability of failure values p_f and q_f are assumed to be equal. The P_{FP} and P_{FN} values are assumed to be equal for each classifier, as are the P_{TP} and P_{TN} values. Additionally, the P_{FP} and P_{FN} values are assumed to be equal to 1 - P_{TP}. The following graph shows which is the best fusion rule, given the preceding assumptions.

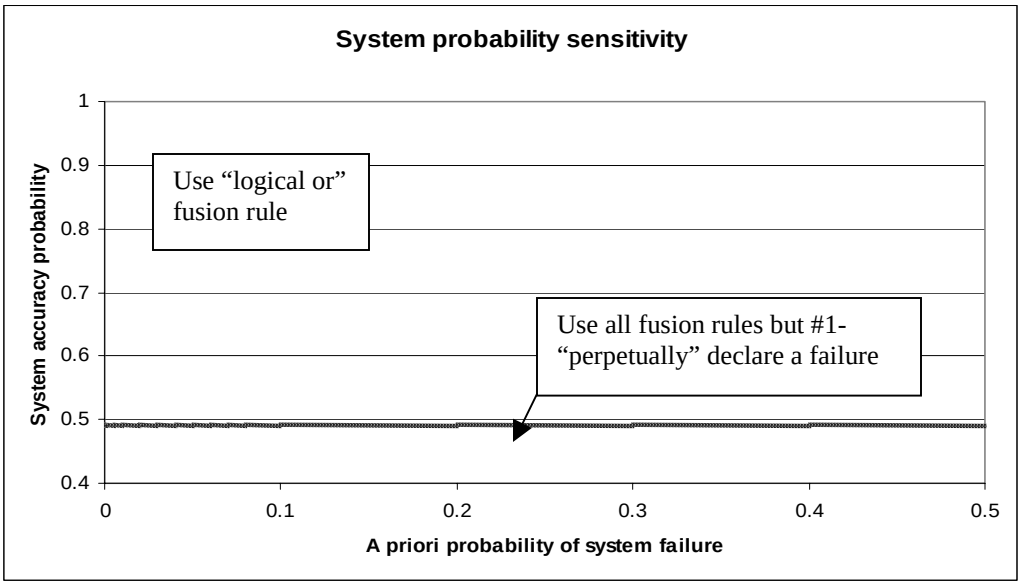

Figure 5a-2. Where the decision rule changes based values of p_f and q_f (x-axis) and values of P_{TP} and P_{TN} (y-axis)

Figure 5a-2 shows a graph of system accuracy vs. the *a priori* probability of system failure. Points on the graph that fall above the line indicate the "logical or" rule for declaring failures should be used for a given system having those characteristics. Points that fall below the line indicate all rules except number 1 should be used to declare a failure. That is, a system should be declared operational only if both classifiers indicate a system failure. This makes sense because the *a priori* probabilities of the classifiers being correct is less than 0.5, and hence the opposite of what the classifiers are reading will be correct more often than the actual readings. As an example, if the two components are expected to fail 10% of the time, and the system correctly reports errors with 80% or better accuracy, the "logical or" fusion rule should be used to make decisions.

Although perpetually declaring a failure may result in the best fusion rule score, it would not result in productive operation of the equipment. This fusion rule ignores all data from sensors and their associated system, making it pointless to install them. The perpetual failure rule contains rule 4. Rule 4 states that if both classifiers declare a normal reading, then a system failure is declared. This doesn't make much sense. Declaring a perpetual failure states that regardless of the classifier readings, a failure is declared. This makes even less sense. In effect, all fusion rules containing rule 4 make no sense, and would not be followed in practice.

If these eight rules for declaring a failure are dropped, then the remaining seven rules are all the combinations of rules 1 (F, F), 2 (F, N), and 3 (N, F). Of these seven combinations, the remaining one that would not be followed in practice would be the combination of rules 2 and 3. This rule states that a failure is declared if one system or the other declares a failure, but no failure is declared if both systems declare a failure. Again, this is not realistic, and this rule would not be followed in practice.

The remaining rule combinations which will be used to further develop the notion of an optimal rule are:

Table 5a-5. Practical fusion rule combinations

Rule Combination	Rules used
"Logical and"	1
Single sensor	2
Single sensor	3
Single sensor plus "Logical and"	1,2
Single sensor plus "Logical and"	1,3
"Logical or"	1,2,3

Table 5a-5 shows the six rule combinations that will be used for all further analysis in this section.

The next issue is weighting different parts of the scoring rule. The new equation is:

$$\text{Fusion rule score} = w_1 P_{TP} + w_2(1 - P_{FP}) \qquad \text{(5a-3)}$$

Recall that $w_1, w_2 \in [0, 1]$ and $w_1 + w_2 = 1$. The weights w_1 and w_2 are set to appropriate values depending on which capability is more important. As an example, inspectors on an assembly line may need to ensure that absolutely no defective parts get through. In probability terms, this means that false positives (claiming a defect exists when it actually doesn't) are less important than false negatives (passing a defective part through as a functional part). Consequently, the value for w_2 would be set much higher than for w_1 in this application. Conversely, it may be more important to ensure that a defect really does exist if there is time pressure to produce the product, and/or defective products don't cost

much if they are mistakenly sent through. In that case, the value for w_1 would be set much higher than for w_2.

It is of interest to examine which fusion rule is best if $w_1 \neq w_2$. For the following discussion, only the ratio w_2/w_1 is considered. P_{TP} is defined to be a function of pf, qf, w_2/w_1 , and R, where $R \in \{(1); (2); (3); (1, 2); (1, 3); (1, 2, 3)\}$ (the six different fusion rules). Let $p_f = q_f = \rho \in [0, 0.6]$, and recall that $w_1, w_2 \in [0,1]$. Let $(w_2/w_1) = r$ $(w_1 \neq 0)$. Then let

$$P_{TP}^*(\rho, r) \equiv \underset{\text{Max } R \in \mathbf{R}}{P_{TP}(\rho, r, R)}$$

where

$$\{(\rho, r) \in [0,1] \times [1, \infty) | P_{TP}(\rho, r, R) \geq P_{TP}^*(\rho, r)\}$$

If the weight w_1 is larger than the value of w_2, then the "logical or" fusion rule is always the best, regardless of the difference in the weights, provided P_{TP}^A and P_{TP}^B is at least 0.5. (If the values for these probabilities fall below 0.5, then rules 2 and 3 tie for the best rule. These results are independent of the prior probabilities of failure.) However, if w_2 was set higher, then the fusion rule would change, based on other system parameters. If the prior probability of system failure was varied, the weight at which the decision rule changed also varied, as shown in the graph below.

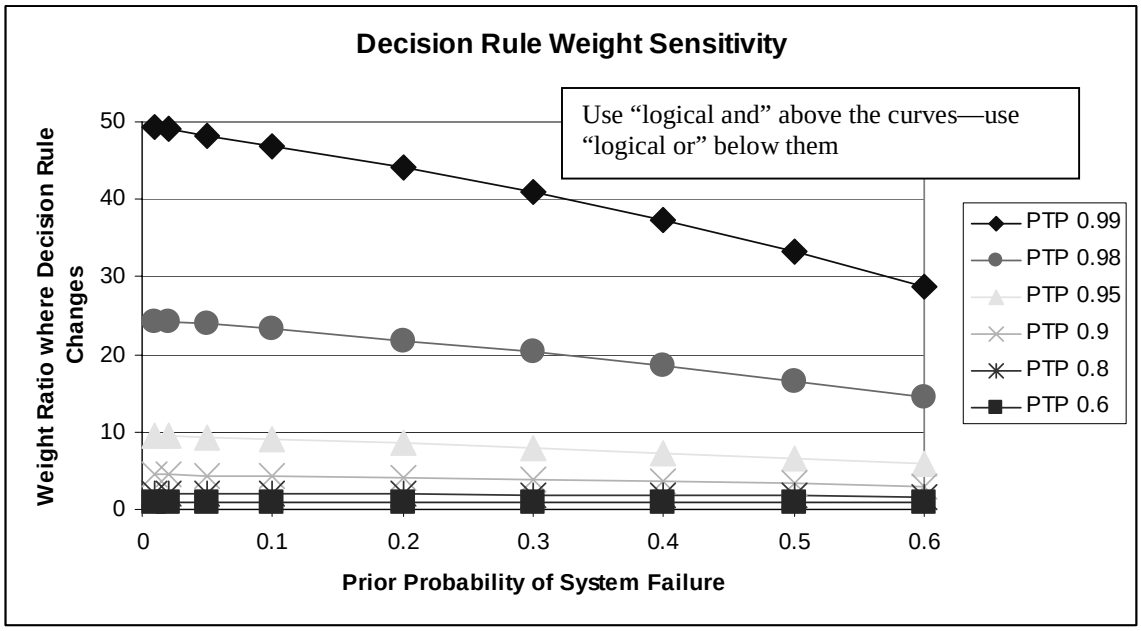

Figure 5a-3. Where the decision rule changes based values of p_f and q_f (x-axis) and weights applied to the scoring rule (y-axis). The values of P_{TP} and P_{TN} are held constant at differing values, as shown in the legend.

Figure 5a-3 shows the ratio of w_2 to w_1 that causes a change in the best decision rule, for the given values of P_{TP} and P_{TN} (recall that $P_{TP} = P_{TN}$). The best decision rule under each curve is the "logical or" decision rule. Above each curve, the best decision rule is the "logical and". As an example, consider the top curve, where $P_{TP} = 0.99$. If the prior probability of system failure is 0.1, then the ratio w_2/w_1 must be at least 45 before the best decision rule changes from "logical or" to "logical and". If the value of P_{TP} becomes 0.6 (the bottom curve) and the prior probability of system failure remains constant, then the ratio drops to 1 before the best decision rule changes. For all practical purposes, the "logical or" fusion rule is the best decision rule for all "realistic" values of P_{TP} and the ratio w_2/w_1.

In each case, the best decision rule was either “logical or” or “logical and”. No other decision rule obtained the best score. The best decision rule also changed when the ratio of the prior probabilities of failure that changed (the weights on the scoring rule were set equal).

It is also of interest to examine which fusion rule is best if $p_f \neq q_f$. Again, only the ratio w_2/w_1 is considered. P_{TP} is still defined to be a function of pf, qf, w_2/w_1 , and R, where R $\in$ {(1); (2); (3); (1, 2); (1, 3); (1, 2, 3)} (the six different fusion rules). Let max (p_f, q_f) = r $\in$ [0, 0.6], and recall that w_1, $w_2 \in$ [0,1]. Let w_2/w_1 = r ($w_1 \neq 0$). Then let

$$P_{TP}^*(\rho, r) \equiv \underset{\text{Max } R \in \mathbf{R}}{P_{TP}(\rho, r, R)}$$

where

$$\{(\rho, r) \in [0,1] \times [1, \infty) | P_{TP}(\rho, r, R) \geq P_{TP}^*(\rho, r)\}$$

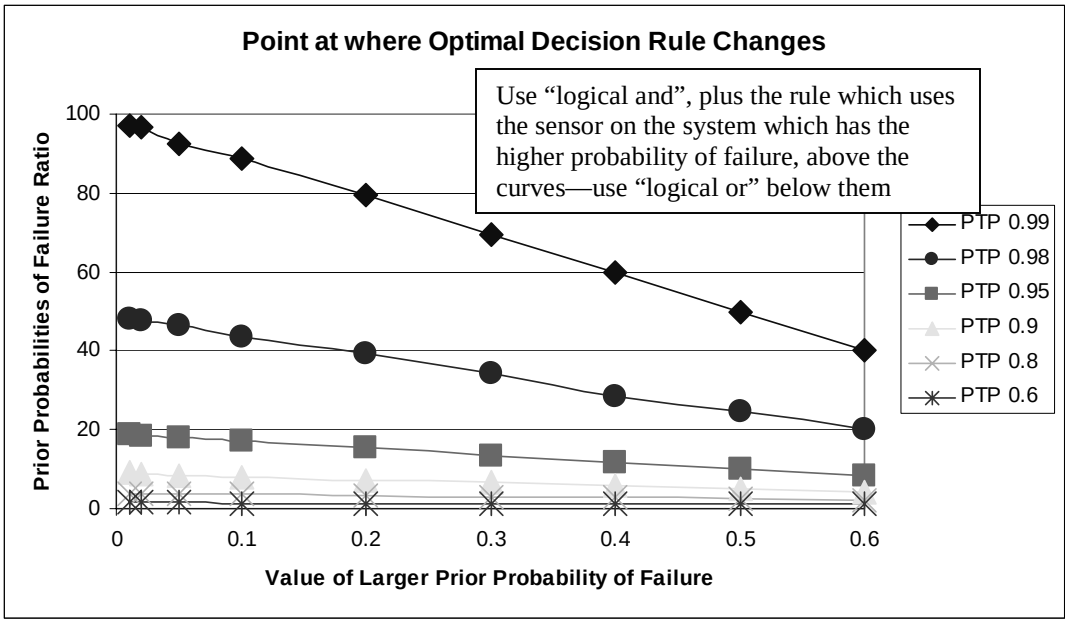

Figure 5a-4. Where the decision rule changes based on max $\{p_f, q_f\}$ (x-axis) and the ratio of p_f to q_f (y-axis). The values of P_{TP} and P_{TN} are held constant at differing values, as shown in the legend.

Figure 5a-4 shows where the decision rule changes based on the prior probabilities of system failure and the probabilities of detection. The decision rule changes between only "logical or" and a two-rule combination. The two rules are "logical and", and using the classifier on the system with the larger probability of failure. The other classifier is ignored except for the "logical and" rule. If the prior probabilities of system failure are low, and the probability of a true positive is high, then the ratio of the larger probability of system failure to the smaller probability of system failure is also high. Specifically, if PTP = 0.99 and the value of the larger probability of failure is 0.1, then the ratio of this larger probability of failure to the smaller probability of failure is about 90 before the decision rule changes from "logical or" to the two-rule combination. Provided the

expected failure rates of the two components are not vastly different, “logical or” is still the best decision rule.

5.A4 Proof that a Logical OR Fusion rule is the Best For a Logical OR Failure Model

THEOREM: Assume there are two components, each with an equal probability of failure less than 0.5. Assume there are two sensors, one for each component, each with an equal probability of (accurately) detecting a failure greater then 0.5 $P_{TP}(A_\theta) = P_{TN}(A_\theta) = P_{TP}(B_\phi) = P_{TN}(B_\phi)$. Then the “logical or” fusion rule provides the best score (Score = $P_{TN} + P_{TP}$) among all six useful fusion rules (1, 2, 3, 1 and 2, 1 and 3, 1 and 2 and 3—see Table 5-12.). (Note that this result does not hold if either or both of the sets of probabilities is not strictly equal.)

PROOF:

Let $0 < \varepsilon < 0.5$.

Assume each component’s probability of failure is $(0.5 - \varepsilon)$.

Assume each sensor’s probability of accurate detection is $(0.5 + \varepsilon)$.

The approach used is to compute the score for each distinct case. Note that the score for rule 2 will be the same as that for rule 3 (the formulas in the table are exactly the same). Similarly, the score for rule combination 1 and 2 will be the same for rule combination 1 and 3. This leaves four distinct cases.

We have $x = (0.5 + \varepsilon)$ and $y = (0.5 - \varepsilon)$. Using the preceding table, the denominator of the P_{TP} expression ($2x^3 + 5x^2y + 4xy^2 + y^3$) reduces to $(1.5 + \varepsilon)$ with these substitutions, and is the same for all cases. The denominator of the P_{TN} (or $1 - P_{FP}$) expression, $(x + y)^2$, simplifies to 1. The P_{TN} results were therefore multiplied by $(1.5 + \varepsilon)$ so both the P_{TP} results and the P_{TN} results were additive. The results that are shown for P_{TN} below are after this multiplication, without showing the $(1.5 + \varepsilon)$ in the denominator.

Case 1: Rule 1. ("logical and") (from the formulas in the table)
$\text{num}(P_{TP}) = 0.375 + 0.75\varepsilon - 1.5\varepsilon^2 - 3\varepsilon^3$.
$\text{num}(P_{TN}) = 0.75 + 2\varepsilon + \varepsilon^2$.
$\text{num}(\text{Score}) = 1.125 + 2.75\varepsilon - 0.5\varepsilon^2 - 3\varepsilon^3$.

Case 2: Rule 2/Rule 3.
$\text{num}(P_{TP}) = 0.375 + 0.25\varepsilon + 0.5\varepsilon^2 + 3\varepsilon^3$.
$\text{num}(P_{TN}) = 0.75 + \varepsilon^2$.
$\text{num}(\text{Score}) = 1.5 + \varepsilon + 2\varepsilon^2 + 4\varepsilon^3$.

Case 3: Rules 1 and 2/Rules 1 and 3.
$\text{num}(P_{TP}) = 0.75 + \varepsilon - \varepsilon^2$.
$\text{num}(P_{TN}) = 0.75 + 2\varepsilon + \varepsilon^2$.
$\text{num}(\text{Score}) = 1.5 + 3\varepsilon$.

Case 4: Rules 1 and 2 and 3 ("logical or")
$\text{num}(P_{TP}) = 1.125 + 1.25\varepsilon - 0.5\varepsilon^2 + 3\varepsilon^3$.
$\text{num}(P_{TN}) = 0.375 + 1.75\varepsilon + 2.5\varepsilon^2 + \varepsilon^3$.
$\text{num}(\text{Score}) = 1.5 + 3\varepsilon + 2\varepsilon^2 + 4\varepsilon^3$.

Clearly, case 4 has the highest score of all the cases. Furthermore, the cases are ordered from lowest score to highest score. The only place this is not obvious is for cases 2 and 3. The difference between the two cases (case 3 minus case 2) is $2\varepsilon - 2\varepsilon^2 - 4\varepsilon^3$.

The claim is

$$2\varepsilon > 2\varepsilon^2 + 4\varepsilon^3, \text{ for all } 0 < \varepsilon < 0.5 \qquad \text{(5a-4)}$$

or equivalently

$$\varepsilon + 2\varepsilon^2 < 1 \qquad (5a\text{-}5)$$

or

$$\varepsilon(1 + 2\varepsilon) < 1. \qquad (5a\text{-}6)$$

Notice that

$$1 < 1 + 2\varepsilon < 2. \qquad (5a\text{-}7)$$

Multiplication by ε yields

$$\varepsilon < \varepsilon(1 + 2\varepsilon) < 2\varepsilon < 1, \qquad (5a\text{-}8)$$

which shows the desired result

$$\varepsilon(1 + 2\varepsilon) < 1. \qquad (5a\text{-}9)$$

This is obviously true for $0 < \varepsilon < 0.5$. Hence case 3 has a larger score than case 2, and the cases are arranged in increasing score order.

5.A5 Proof that a Logical AND Fusion rule is the Best For a Logical AND Failure Model

The result from the preceding section suggests that a "logical and" failure rule would be optimal for a "logical and" failure model.

The implicit assumption in a "logical and" failure model is that a system component (subcomponent, etc.) is functional until every part in the component has failed. This means that not every part is critical to system operation. This assumption contradicts the general formulation of the system model presented in this paper, where every part of the component is considered to be critical to system operation. However, there are system

components that are designed to be redundant. These components have many subcomponents which all perform the same operation. If some subcomponents fail, the remaining subcomponents will continue to perform the operation that is critical to system functionality. In the extreme case, if all the subcomponents fail except one, that single remaining subcomponent can still perform the component's function. Since some components of a system may be designed to be redundant, it seems worthwhile to determine which fusion rule is best (if there is a "best" rule) for those components which have a redundant functional design.

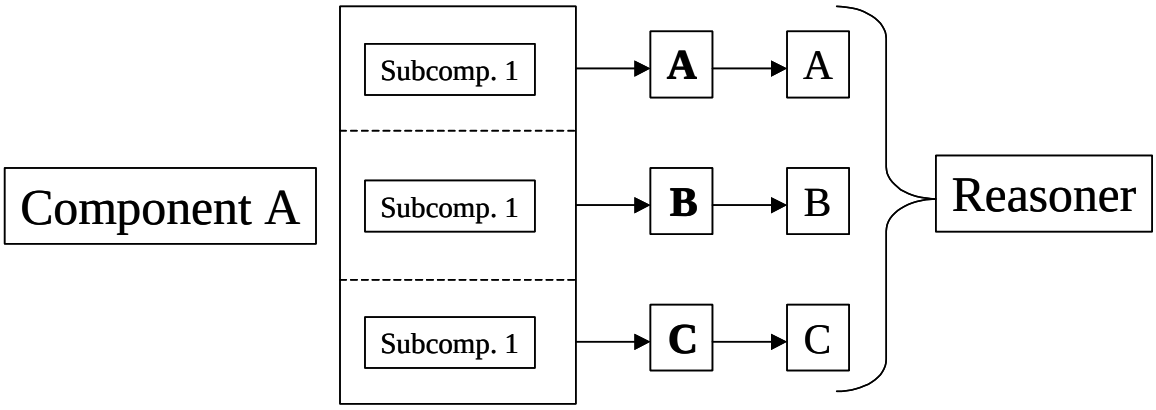

Figure 5a-5. A notional component designed to have redundant functionality

Figure 5a-5 shows a system component designed to have redundant functionality. Each subcomponent has the same number to indicate identical functionality. This component would not be considered to have failed until all three subcomponents fail.

Table 5a-6. **Table 5a-2 reproduced for ease of reference**

Classifier Reports (A, B) True State	F, F	F, N	N, F	N, N
F, F	1. x^2y^2	2. xy^3	3. xy^3	4. y^4
F, N	5. x^2y^2	6. $\mathbf{x^3y}$	7. xy^3	8. x^2y^2
N, F	9. x^2y^2	10. xy^3	11. $\mathbf{x^3y}$	12. x^2y^2
N, N	13. x^2y^2	14. $\mathbf{x^3y}$	15. $\mathbf{x^3y}$	16. $\mathbf{x^4}$

Table 5a-6 shows the likelihood of the occurrence of a particular event, and is reproduced here as an aid for Table 5a-7.

Table 5a-7. Summary of probability values for different fusion rules

Probability Measure Fusion Rule	Cells used to declare a failure (Cells with an actual failure are 1-4)	P_{TP} True Positive (intersection with cells 1-4)	P_{FP} False Positive (intersection with cells 5-16)
1 (logical and)	1, 5, 9, 13	$\frac{x^2}{(x+y)^2}$	$\frac{3xy^2}{(x^3+4x^2y+5xy^2+2y^3)}$
2	2, 6, 10, 14	$\frac{xy}{(x+y)^2}$	$\frac{2x^2y+y^3}{(x^3+4x^2y+5xy^2+2y^3)}$
3	3, 7, 11, 15	$\frac{xy}{(x+y)^2}$	$\frac{2x^2y+y^3}{(x^3+4x^2y+5xy^2+2y^3)}$
4	4, 8, 12, 16	$\frac{y^2}{(x+y)^2}$	$\frac{x^3+2xy^2}{(x^3+4x^2y+5xy^2+2y^3)}$
1, 2	1, 2, 5, 6, 9, 10, 13, 14	$\frac{x^2+xy}{(x+y)^2}$	$\frac{2x^2y+3xy^2+y^3}{(x^3+4x^2y+5xy^2+2y^3)}$
1, 3	1, 3, 5, 7, 9, 11, 13, 15	$\frac{x^2+xy}{(x+y)^2}$	$\frac{2x^2y+3xy^2+y^3}{(x^3+4x^2y+5xy^2+2y^3)}$
1, 4	1, 4, 5, 8, 9, 12, 13, 16	$\frac{x^2+y^2}{(x+y)^2}$	$\frac{x^3+5x^2y}{(x^3+4x^2y+5xy^2+2y^3)}$
2, 3	2, 3, 6, 7, 10, 11, 14, 15	$\frac{2xy}{(x+y)^2}$	$\frac{4xy^2+2y^3}{(x^3+4x^2y+5xy^2+2y^3)}$

2, 4	2, 4, 6, 8, 10, 12, 14, 16	$\frac{xy + y^2}{(x + y)^2}$	$\frac{x^3 + 2x^2y + 2xy^2 + y^3}{(x^3 + 4x^2y + 5xy^2 + 2y^3)}$
3, 4	3, 4, 7, 8, 11, 12, 15, 16	$\frac{xy + y^2}{(x + y)^2}$	$\frac{x^3 + 2x^2y + 2xy^2 + y^3}{(x^3 + 4x^2y + 5xy^2 + 2y^3)}$
1, 2, 3 (logical or)	1-3, 5-7, 9-11, 13-15	$\frac{x^2 + 2xy}{(x + y)^2}$	$\frac{4x^2y + 3xy^2 + 2y^3}{(x^3 + 4x^2y + 5xy^2 + 2y^3)}$
1, 2, 4	1-2, 4-6, 8-10, 12-14, 16	$\frac{x^2 + xy + y^2}{(x + y)^2}$	$\frac{x^3 + 2x^2y + 5xy^2 + y^3}{(x^3 + 4x^2y + 5xy^2 + 2y^3)}$
1, 3, 4	1, 3-5, 7-9, 11-13, 15-16	$\frac{x^2 + xy + y^2}{(x + y)^2}$	$\frac{x^3 + 2x^2y + 5xy^2 + y^3}{(x^3 + 4x^2y + 5xy^2 + 2y^3)}$
2, 3, 4	2-4, 6-8, 10-12, 14-16	$\frac{2xy + y^2}{(x + y)^2}$	$\frac{x^3 + 4x^2y + 2xy^2 + 2y^3}{(x^3 + 4x^2y + 5xy^2 + 2y^3)}$
1, 2, 3, 4	1-16	1	1

Table 5a-7 shows the P_{TP} and P_{FP} values for each of the 15 different fusion rules. Since it is hard to see from Table 5a-7 which rules have high and low probability values, the following table provides an example with specific values.

Table 5a-8. Fusion rule probability values for a specific case

Probability Measure	PTP (overlap with cells 1-4)	PFP (overlap with cells 5-16)	Score:
Fusion Rule			
1	0.9801	0.009850754	1.970249246
2	0.0099	0.487686935	0.522213065
3	0.0099	0.487686935	0.522213065
4	0.0001	0.014775377	0.985324623
1,2	0.99	0.497537688	1.492462312
1,3	0.99	0.497537688	1.492462312
1,4	0.9802	0.024626131	1.955573869
2,3	0.0198	0.975373869	0.044426131
2,4	0.01	0.502462312	0.507537688
3,4	0.01	0.502462312	0.507537688
1,2,3	0.9999	0.985224623	1.014675377
1,2,4	0.9901	0.512313065	1.477786935
1,3,4	0.9901	0.512313065	1.477786935
2,3,4	0.0199	0.990149246	0.029750754
1,2,3,4	1	1	1

Table 5a-8 shows the values that would be obtained if the following substitutions were made: $p_f = q_f = P_{FP} = P_{FN} = .01$, $P_{TP} = P_{TN} = .99$. Note that the "logical and" rule provides the highest fusion rule score.

The theorem and proof are analogous to the preceding section.

THEOREM: Assume there are two components, each with an equal probability of failure less than 0.5. Assume there are two sensors, one for each component, each with an equal probability of (accurately) detecting a failure greater then 0.5 $P_{TP}(A_\theta) = P_{TN}(A_\theta) = P_{TP}(B_\phi) = P_{TN}(B_\phi)$. Then the "logical and" fusion rule provides the best score (Score = $P_{TN} + P_{TP}$) among all six useful fusion rules (1, 2, 3, 1 and 2, 1 and 3, 1 and 2 and 3). (Note that this result does not hold if either or both of the sets of probabilities is not strictly equal.)

PROOF:

Let $0 < \varepsilon < 0.5$.

Assume each component's probability of failure is $(0.5 - \varepsilon)$.

Assume each sensor's probability of accurate detection is $(0.5 + \varepsilon)$.

The approach used is to compute the score for each distinct case. Note that the score for rule 2 will be the same as that for rule 3 (the formulas in the table are exactly the same). Similarly, the score for rule combination 1 and 2 will be the same for rule combination 1 and 3. This leaves four distinct cases.

We have $x = (0.5 + \varepsilon)$ and $y = (0.5 - \varepsilon)$. Using the preceding table, the denominator of the P_{TN} (or $1 - P_{FP}$) expression ($x^3 + 4x^2y + 5xy^2 + y^3$) reduces to $(1.5 - \varepsilon)$ with these substitutions, and is the same for all entries in the table. The denominator of the P_{TP} expression, $(x + y)^2$, simplifies to 1. The P_{TP} results were therefore multiplied by $(1.5 - \varepsilon)$ so both the P_{TP} results and the P_{TN} results were additive. The results that are shown for P_{TP} below are after this multiplication, without showing the $(1.5 - \varepsilon)$ in the denominator.

Case 1: Rule 1. ("logical and") (from the formulas in the table)

$num(P_{TP}) = 0.375 + 1.25\varepsilon + 0.5\varepsilon^2 - \varepsilon^3$.
$num(P_{TN}) = 0.625 + 0.75\varepsilon + 1.5\varepsilon^2 - 3\varepsilon^3$.
$num(Score) = 1 + 2\varepsilon + 2\varepsilon^2 - 4\varepsilon^3$.

Case 2: Rule 2/Rule 3.

$num(P_{TP}) = 0.375 - 0.25\varepsilon - 1.5\varepsilon^2 + \varepsilon^3$.
$num(P_{TN}) = 0.375 + 0.25\varepsilon - 0.5\varepsilon^2 + 3\varepsilon^3$.
$num(Score) = .75 - 2\varepsilon^2 + 4\varepsilon^3$.

Case 3: Rules 1 and 2/Rules 1 and 3.

$num(P_{TP}) = 0.75 + \varepsilon - \varepsilon^2$.
$num(P_{TN}) = 0.25 + \varepsilon + \varepsilon^2$.
$num(Score) = 1 + 2\varepsilon$.

Case 4: Rules 1 and 2 and 3 ("logical or")

$num(P_{TP}) = 1.125 - 0.75\varepsilon - 2.5\varepsilon^2 + \varepsilon^3$.
$num(P_{TN}) = -0.675 + 1.25\varepsilon + 0.5\varepsilon^2 + 3\varepsilon^3$.
$num(Score) = 0.5 + 0.5\varepsilon - 2\varepsilon^2 + 4\varepsilon^3$.

Clearly, the "logical and" fusion rule has the highest score among these four cases (note that $2\varepsilon^2 > 4\varepsilon^3$ because 2 is always greater than 4ε when $\varepsilon < 0.5$). Not surprisingly, the "logical or" rule has the lowest score. This result indicates that the "logical and" fusion rule should be used to assess the health of components which have redundant functionality.

Appendix D. Computer Code

```
function [subset] = subsetgen()

global subset tot N

% This program lists the natural lexicographic order of all subsets  for a given number of sensors, up to 9 total

% Input number of sensors, total count, and storage matrix

N=3;
tot=2^N-1;
subset=zeros(tot,2); % First column is index, second is subset

% Initialize counts
a=0;
b=0;
c=0;
d=0;
e=0;
f=0;
g=0;
h=0;
k=0;

t=0; % Used as sensor subset index

% Subsets of size 1

for a=1:N
   t=t+1;
   subset(t,1)=t;
   subset(t,2)=a;
end

% Subsets of size 2

for a=1:N-1
   for b=2:N
      if b>a
         t=t+1;
         input=10*a+b;
         subset(t,1)=t;
         subset(t,2)=input;
      end
   end
end

% Subsets of size 3

for a=1:N-2
   for b=2:N-1
```

```
      for c=3:N
         if b>a
            if c>b
               t=t+1;
               input=100*a+10*b+c;
               subset(t,1)=t;
               subset(t,2)=input;
            end
         end
      end
   end
end

% Subsets of size 4

for a=1:N-3
   for b=2:N-2
      for c=3:N-1
         for d=4:N
            if b>a
               if c>b
                  if d>c
                     t=t+1;
                     input=1000*a+100*b+10*c+d;
                     subset(t,1)=t;
                     subset(t,2)=input;
                  end
               end
            end
         end
      end
   end
end

% Subsets of size 5

for a=1:N-4
   for b=2:N-3
      for c=3:N-2
         for d=4:N-1
            for e=5:N
               if b>a
                  if c>b
                     if d>c
                        if e>d
                           t=t+1;
                           input=10000*a+1000*b+100*c+10*d+e;
                           subset(t,1)=t;
                           subset(t,2)=input;
                        end
                     end
                  end
               end
            end
```

```
        end
      end
   end
end

% Subsets of size 6

for a=1:N-5
   for b=2:N-4
      for c=3:N-3
         for d=4:N-2
            for e=5:N-1
               for f=6:N
                  if b>a
                     if c>b
                        if d>c
                           if e>d
                              if f>e
                                 t=t+1;
                                 input=100000*a+10000*b+1000*c+100*d+10*e+f;
                                 subset(t,1)=t;
                                 subset(t,2)=input;
                              end
                           end
                        end
                     end
                  end
               end
            end
         end
      end
   end
end

% Subsets of size 7

for a=1:N-6
   for b=2:N-5
      for c=3:N-4
         for d=4:N-3
            for e=5:N-2
               for f=6:N-1
                  for g=7:N
                     if b>a
                        if c>b
                           if d>c
                              if e>d
                                 if f>e
                                    if g>f
                                       t=t+1;
                                       input=1000000*a+100000*b+10000*c+1000*d+100*e+10*f+g;
                                       subset(t,1)=t;
                                       subset(t,2)=input;
                                    end
```

```
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end

% Subsets of size 8

for a=1:N-7
   for b=2:N-6
      for c=3:N-5
         for d=4:N-4
            for e=5:N-3
               for f=6:N-2
                  for g=7:N-1
                     for h=8:N
                        if b>a
                           if c>b
                              if d>c
                                 if e>d
                                    if f>e
                                       if g>f
                                          if h>g
                                             t=t+1;

input=10000000*a+1000000*b+100000*c+10000*d+1000*e+100*f+10*g+h;
                                             subset(t,1)=t;
                                             subset(t,2)=input;
                                          end
                                       end
                                    end
                                 end
                              end
                           end
                        end
                     end
                  end
               end
            end
         end
      end
   end
end

% Subsets of size 9

for a=1:N-8
```

```
   for b=2:N-7
      for c=3:N-6
         for d=4:N-5
            for e=5:N-4
               for f=6:N-3
                  for g=7:N-2
                     for h=8:N-1
                        for k=9:N
                           if b>a
                              if c>b
                                 if d>c
                                    if e>d
                                       if f>e
                                          if g>f
                                             if h>g
                                                if k>h
                                                   t=t+1;

input=100000000*a+10000000*b+1000000*c+100000*d+10000*e+1000*f+100*g+10*h+k;
                                                   subset(t,1)=t;
                                                   subset(t,2)=input;
                                                end
                                             end
                                          end
                                       end
                                    end
                                 end
                              end
                           end
                        end
                     end
                  end
               end
            end
         end
      end
   end
end
```

```
function [fD] = combinet(rocA,rocB)

% This program combines 2 ROC curves using within fusion

global M N x roc rocA rocB I J K Q alpha beta gam temp
global fA fB fC fD fBQ

% ROC curve computation

xFP=2*x-x.^2;
rocTP=zeros(1,N);
for I=1:N
   rocTP(I)=rocA(I)+rocB(I)-rocA(I)*rocB(I);
end
fD=rocTP;
fA=rocA;
fB=rocB;

fD=interp1(xFP,rocTP,x);

figure
plot(x,fA,'red.');
hold on
plot(x,fB,'blue.');
hold on
plot(x,fD,'green.');
hold off


function [fC] = combine(rocA,rocB,fBQ)

% This program combines 2 ROC curves using across fusion

global M N x roc rocA rocB I J K Q alpha beta gam temp
global fA fB fC fD fBQ

% ROC curve computation
fA = rocA;
FA = alpha*fA + (1-alpha)*x;
GA = 1 - FA;

fB = rocB;
fBQa = fBQ(K,1:N,1:N);
fBQaa = zeros(N,N);
for I=1:N
   for J=1:N
      fBQaa(I,J)=fBQa(1,I,J);
   end
end
FBQ = beta*fBQaa + (1-beta)*Q;
GBQ = 1 - FBQ;

fC = zeros(1,N);
for I=1:N,
```

```
   minvalue = min(GA(1:I).*GBQ(I,1:I));
   fC(I) = temp - (temp-1)*((I-1)/M) - temp*minvalue;
end
%fC=interp1(?,fCa,x)
figure
plot(x,fA,'red.');
hold on
plot(x,fB,'blue.');
hold on
plot(x,fC,'green.');
hold off
```

```
% This program computes every cost feasible ROC curve combination of sensors on a two component
system.  Each component consists of two subcomponents.  Three of the subcomponents have two sensors,
and one has three.

global M N x roc rocA rocB C I J K Q alpha beta gam temp
global fA fB fC fD fBQ fCout

salloc22 % (get combined curves from other component-
        % 2x2 subcomponent configuration)

M=100;% the number of subintervals
    % used to partition the interval [0,1]

%%% Initialize x coordinates
N = M+1;   % number of points used to plot
x = zeros(1,N);
for I=1:N,
   x(I) = (I-1)/M;
end

% Enter the prior probability alpha
alpha = 0.5;

% Enter the prior probability beta
beta = 0.5;

% Initialize Q
gam = alpha + beta - alpha*beta;
temp = 1/gam;
Q = zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
     if J <= I;
        Q(I,J) = (I-J)/(N-J);
    end
    R = Q(I,1:J);
    %fBQ(I,1:J) = interp1(p,fB,R);
  end
end

roc=zeros(5,N);  % 5 ROC curves
fBQ=zeros(10,N,N);  % 10 different entries

% ROC 1
roc(1,1:N)=(x).^.1;
fBQ(1,1:N,1:N) = (Q).^(.1);

% ROC 2
roc(2,1:N)=((2/pi)*asin(x)).^(1/6);
fBQ(2,1:N,1:N) = ((2/pi)*asin(Q)).^(1/6);

% ROC 3
roc(3,1:N)=tanh(4*x);
fBQ(3,1:N,1:N) = tanh(4*Q);
```

```
% ROC 4
roc4=(x).^.13;
roc(4,1:N)=roc4;
fBQ(4,1:N,1:N) = (Q).^(.13);

% ROC 5
roc5=zeros(1,N);
for p=1:N
   pp=(p/100)-.01;
   roc5(p)=((1-(pp-1)^2)^(.5));
end
roc(5,1:N)=roc5;
fBQ(5,1:N,1:N)=((1-(Q-1).^2).^(.5));

% Plot all five ROC curves

figure
plot(x,roc(1,1:N),'r',x,roc(2,1:N),'y',x,roc(3,1:N),'g',x,roc(4,1:N),'b',x,roc(5,1:N),'k')
legend('ROC curve E', 'ROC curve F', 'ROC curve G', 'ROC curve H','ROC Curve I',4);
xlabel('Probability of False Positive');
ylabel('Probability of True Positive');
title('Individual Sensor ROC Curves');

% Set cost for each curve, and total budget

cost1=45;
cost2=30;
cost3=25;
cost4=35;
cost5=35;
budget=135;

% 3 combinations are not cost feasible

% Determine cost for each combination

cost13=cost1+cost3;
cost14=cost1+cost4;
cost15=cost1+cost5;
cost23=cost2+cost3;
cost24=cost2+cost4;
cost25=cost2+cost5;
cost123=cost1+cost2+cost3;
cost124=cost1+cost2+cost4;
cost125=cost1+cost2+cost5;
cost134=cost1+cost3+cost4;
cost135=cost1+cost3+cost5;
cost145=cost1+cost4+cost5;
cost234=cost2+cost3+cost4;
cost235=cost2+cost3+cost5;
cost245=cost2+cost4+cost5;
cost1234=cost1+cost2+cost3+cost4;
cost1235=cost1+cost2+cost3+cost5;
```

```
cost1245=cost1+cost2+cost4+cost5;
cost1345=cost1+cost3+cost4+cost5;
cost2345=cost2+cost3+cost4+cost5;
cost12345=cost1+cost2+cost3+cost4+cost5;

%Initialize ROC curves

fC13=zeros(1,N);
fC14=zeros(1,N);
fC15=zeros(1,N);
fC23=zeros(1,N);
fC24=zeros(1,N);
fC25=zeros(1,N);
fC123=zeros(1,N);
fC124=zeros(1,N);
fC125=zeros(1,N);
fC134=zeros(1,N);
fC135=zeros(1,N);
fC145=zeros(1,N);
fC234=zeros(1,N);
fC235=zeros(1,N);
fC245=zeros(1,N);
fC1234=zeros(1,N);
fC1235=zeros(1,N);
fC1245=zeros(1,N);
fC1345=zeros(1,N);
fC2345=zeros(1,N);
fC12345=zeros(1,N);

% Run combinations if cost eligible

% Same side

% Combination 12
K=1;
rocA=roc(K,1:N);
K=2;
rocB=roc(K,1:N);
combinet;
fD12=fD;
%xFP=2*x-x.^2;

% Combination 34
K=3;
rocA=roc(K,1:N);
K=4;
rocB=roc(K,1:N);
combinet;
fD34=fD;
%xFP=2*x-x.^2;

% Combination 35
K=3;
rocA=roc(K,1:N);
```

```
K=5;
rocB=roc(K,1:N);
combinet;
fD35=fD;
%xFP=2*x-x.^2;

% Combination 45
K=4;
rocA=roc(K,1:N);
K=5;
rocB=roc(K,1:N);
combinet;
fD45=fD;
%xFP=2*x-x.^2;

%Combination 345
rocA=fD34;
K=5;
rocB=roc(K,1:N);
combinet;
fD345=fD;

% Different sides (2 sensors)

% Combination 13
if cost13 <= budget
K=1;
rocA=roc(K,1:N);
K=3;
rocB=roc(K,1:N);
combine;
fC13=fC;
end

% Combination 14
if cost14 <= budget
K=1;
rocA=roc(K,1:N);
K=4;
rocB=roc(K,1:N);
combine;
fC14=fC;
end

% Combination 15
if cost15 <= budget
K=1;
rocA=roc(K,1:N);
K=5;
rocB=roc(K,1:N);
combine;
fC15=fC;
end
```

```
% Combination 23
if cost23 <= budget
K=2;
rocA=roc(K,1:N);
K=3;
rocB=roc(K,1:N);
combine;
fC23=fC;
end

% Combination 24
if cost24 <= budget
K=2;
rocA=roc(K,1:N);
K=4;
rocB=roc(K,1:N);
combine;
fC24=fC;
end

% Combination 25
if cost25 <= budget
K=2;
rocA=roc(K,1:N);
K=5;
rocB=roc(K,1:N);
combine;
fC25=fC;
end

% Different sides (3 sensors)

% Combination 123
if cost123 <= budget
rocA=fD12;
K=3;
rocB=roc(K,1:N);
combine;
fC123=fC;
end

% Combination 124
if cost124 <= budget
rocA=fD12;
K=4;
rocB=roc(K,1:N);
combine;
fC124=fC;
end

% Combination 125
if cost125 <= budget
rocA=fD12;
K=5;
```

```
rocB=roc(K,1:N);
combine;
fC125=fC;
end

% Combination 134
if cost134 <= budget
rocA=fD34;
K=1;
rocB=roc(K,1:N);
combine;
fC134=fC;
end

% Combination 135
if cost135 <= budget
rocA=fD35;
K=1;
rocB=roc(K,1:N);
combine;
fC135=fC;
end

% Combination 145
if cost145 <= budget
rocA=fD45;
K=1;
rocB=roc(K,1:N);
combine;
fC145=fC;
end

% Combination 234
if cost234 <= budget
rocA=fD34;
K=2;
rocB=roc(K,1:N);
combine;
fC234=fC;
end

% Combination 235
if cost235 <= budget
rocA=fD35;
K=2;
rocB=roc(K,1:N);
combine;
fC235=fC;
end

% Combination 245
if cost245 <= budget
rocA=fD45;
K=2;
```

```
rocB=roc(K,1:N);
combine;
fC245=fC;
end

% Different sides (4 sensors)

%Combination 1234
if cost1234 <= budget
rocA=fD12;
rocB=fD34;
fBQs=zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
     R = Q(I,1:J);
     fBQs(I,1:J) = interp1(x,fD34,R);
  end
end
fBQ(6,1:N,1:N)=fBQs;
K=6;
combine;
fC1234=fC;
end

%Combination 1235
if cost1235 <= budget
rocA=fD12;
rocB=fD35;
fBQs=zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
     R = Q(I,1:J);
     fBQs(I,1:J) = interp1(x,fD35,R);
  end
end
fBQ(7,1:N,1:N)=fBQs;
K=7;
combine;
fC1235=fC;
end

%Combination 1245
if cost1245 <= budget
rocA=fD12;
rocB=fD45;
fBQs=zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
     R = Q(I,1:J);
     fBQs(I,1:J) = interp1(x,fD45,R);
  end
end
fBQ(8,1:N,1:N)=fBQs;
K=8;
```

```
combine;
fC1245=fC;
end

% Combination 1345
if cost1345 <= budget
rocA=fD345;
K=1;
rocB=roc(K,1:N);
combine;
fC1345=fC;
end

% Combination 2345
if cost2345 <= budget
rocA=fD345;
K=2;
rocB=roc(K,1:N);
combine;
fC2345=fC;
end

% Different sides (5 sensors)

if cost12345 <= budget
rocA=fD12;
rocB=fD345;
fBQs=zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
     R = Q(I,1:J);
     fBQs(I,1:J) = interp1(x,fD345,R);
  end
end
fBQ(9,1:N,1:N)=fBQs;
K=9;
combine;
fC12345=fC;
end

% Store results in a single array
fCouta=zeros(21,N);
fCouta(1,1:N)=fC13;
fCouta(2,1:N)=fC14;
fCouta(3,1:N)=fC15;
fCouta(4,1:N)=fC23;
fCouta(5,1:N)=fC24;
fCouta(6,1:N)=fC25;
fCouta(7,1:N)=fC123;
fCouta(8,1:N)=fC124;
fCouta(9,1:N)=fC125;
fCouta(10,1:N)=fC134;
fCouta(11,1:N)=fC135;
fCouta(12,1:N)=fC145;
```

```
fCouta(13,1:N)=fC234;
fCouta(14,1:N)=fC235;
fCouta(15,1:N)=fC245;
fCouta(16,1:N)=fC1234;
fCouta(17,1:N)=fC1235;
fCouta(18,1:N)=fC1245;
fCouta(19,1:N)=fC1345;
fCouta(20,1:N)=fC2345;
fCouta(21,1:N)=fC12345;

% Combine results from both components
C=0;
fCboth=zeros(189,N);   % Change based on configuration
for II=1:21            % Change based on configuration
   for JJ=1:9          % Change based on configuration
      C=C+1;
      if fCout(JJ,50)>0
        rocA=fCout(JJ,1:N);
        rocB=fCouta(II,1:N);
        fBQs=zeros(N);
        for I = 1:N;      %r=(I-1)/M
          for J = 1:M;    %p=(J-1)/M
            R = Q(I,1:J);
            fBQs(I,1:J) = interp1(x,rocB,R);
          end
        end
        fBQ(10,1:N,1:N)=fBQs;
        K=10;
        combine;
        fCboth(C,1:N)=fC;
      end
   end
end

% Determine best curve
fCbotht=fCboth';
for I=1:N
   [maxroc(I),maxind(I)]=max(fCbotht(I,:));
end
figure
plot(maxind);
figure
plot(maxroc);
```

```
function [fCout] = salloc22()

% This program determines the optimal sensor allocation for a particular system component.  The
component is assumed to consist of 2 subcomponents, each with 2 sensors.  Each subcomponent is assumed
to require at least one sensor.

global M N x roc rocA rocB I J K Q alpha beta gam temp
global fA fB fC fD fBQ fCout

M=100;% the number of subintervals used to partition the interval [0,1]

%%% Initialize x coordinates
N = M+1;   % number of points used to plot
x = zeros(1,N);
for I=1:N,
   x(I) = (I-1)/M;
end

% Enter the prior probability alpha
alpha = 0.5;
% Enter the prior probability beta
beta = 0.5;

% Initialize Q
gam = alpha + beta - alpha*beta;
temp = 1/gam;
Q = zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
    if J <= I;
      Q(I,J) = (I-J)/(N-J);
    end
    R = Q(I,1:J);
    %fBQ(I,1:J) = interp1(p,fB,R);
  end
end

roc=zeros(4,N);
fBQ=zeros(5,N,N);

% ROC 1
roc(1,1:N)=(x).^.1;
fBQ(1,1:N,1:N) = (Q).^(.1);

% ROC 2
roc(2,1:N)=((2/pi)*asin(x)).^(1/6);
fBQ(2,1:N,1:N) = ((2/pi)*asin(Q)).^(1/6);

% ROC 3
roc(3,1:N)=tanh(4*x);
fBQ(3,1:N,1:N) = tanh(4*Q);

% ROC 4
roc4=(x).^.13;
```

```
roc(4,1:N)=roc4;
fBQ(4,1:N,1:N) = (Q).^(.13);

% Set cost for each curve, and total budget

cost1=45;
cost2=30;
cost3=25;
cost4=35;
budget=125;

% Determine budget eligibility for each combination

cost13=cost1+cost3;
cost14=cost1+cost4;
cost23=cost2+cost3;
cost24=cost2+cost4;
cost123=cost1+cost2+cost3;
cost124=cost1+cost2+cost4;
cost134=cost1+cost3+cost4;
cost234=cost2+cost3+cost4;
cost1234=cost1+cost2+cost3+cost4;

%Initialize ROC curves

fC13=zeros(1,N);
fC14=zeros(1,N);
fC23=zeros(1,N);
fC24=zeros(1,N);
fC123=zeros(1,N);
fC124=zeros(1,N);
fC134=zeros(1,N);
fC234=zeros(1,N);
fC1234=zeros(1,N);

% Run combinations if cost eligible

% Same side

% Combination 12
K=1;
rocA=roc(K,1:N);
K=2;
rocB=roc(K,1:N);
combinet;
fD12=fD;
%xFP=2*x-x.^2;

% Combination 34
K=3;
rocA=roc(K,1:N);
K=4;
rocB=roc(K,1:N);
combinet;
```

```
fD34=fD;
%xFP=2*x-x.^2;

% Different sides (2 sensors)
% Combination 13
if cost13 <= budget
K=1;
rocA=roc(K,1:N);
K=3;
rocB=roc(K,1:N);
combine;
fC13=fC;
end

% Combination 14
if cost14 <= budget
K=1;
rocA=roc(K,1:N);
K=4;
rocB=roc(K,1:N);
combine;
fC14=fC;
end

% Combination 23
if cost23 <= budget
K=2;
rocA=roc(K,1:N);
K=3;
rocB=roc(K,1:N);
combine;
fC23=fC;
end

% Combination 24
if cost24 <= budget
K=2;
rocA=roc(K,1:N);
K=4;
rocB=roc(K,1:N);
combine;
fC24=fC;
end

% Different sides (3 sensors)
% Combination 123
if cost123 <= budget
rocA=fD12;
K=3;
rocB=roc(K,1:N);
combine;
fC123=fC;
end
```

```
% Combination 124
if cost124 <= budget
rocA=fD12;
K=4;
rocB=roc(K,1:N);
combine;
fC124=fC;
end

% Combination 134
if cost124 <= budget
rocA=fD34;
K=1;
rocB=roc(K,1:N);
combine;
fC134=fC;
end

% Combination 234
if cost234 <= budget
rocA=fD34;
K=2;
rocB=roc(K,1:N);
combine;
fC234=fC;
end

% Different sides (4 sensors)
%Combination 1234
if cost1234 <= budget
rocA=fD12;
rocB=fD34;
fBQs=zeros(N);
for I = 1:N;      %r=(I-1)/M
  for J =1:M;    %p=(J-1)/M
     R = Q(I,1:J);
     fBQs(I,1:J) = interp1(x,fD34,R);
  end
end
fBQ(5,1:N,1:N)=fBQs;
K=5;
combine;
fC1234=fC;
end
fCout=zeros(9,N);
fCout(1,1:N)=fC13;
fCout(2,1:N)=fC14;
fCout(3,1:N)=fC23;
fCout(4,1:N)=fC24;
fCout(5,1:N)=fC123;
fCout(6,1:N)=fC124;
fCout(7,1:N)=fC134;
fCout(8,1:N)=fC234;
fCout(9,1:N)=fC1234;
```

Bibliography

1. Aamodt, A. and Plaza, E. "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches." AI Communications, vol. 7, no. 1, pp 39-59, 1994.

2. Albus, J. "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC), Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control, pp 220-227, 1975.

3. Alkon, Blackwell, Barbour, Werness, and Vogl. "Intelligent Robots and Computer Vision: Fifth in a Series." Proceedings of SPIE – The International Society for Optical Engineering, vol. 726, pp 552-557, 1986.

4. Allison Engine Company. "AE 3007 High Reliability Engine Control Program – Development/Demonstration of Reliability Enhanced Control Technologies to Improve Aircraft Dispatch Reliability." NASA3 – 27394 Task Order 008, April 1998.

5. Alsing S. "The Evaluation of Competing Classifiers". Ph D Dissertation, Air Force Institute of Technology, March 2000.

6. Anonymous. Webpage at http://www.fmeca.com/.

7. Anonymous. Webpage at http://www.libinst.com/cepst.htm.

8. Anonymous. Webpage at http://www.mtain.com/relia/relfmeca.htm

9. Atlas, L., Bloor, G., Brotherton, T., Howard, L., Jaw, L., Kacprzynski, G., Karsai, G., Mackey, R., Mesick, J., Reuter, R., and Roemer, M. "An Evolvable Tri-Reasoner IVHM System". The Boeing Company, 1999.

10. Azzam, H. and Brindley, J. "Structural Health Monitoring." MJA Dynamics Briefing. Hampshire, UK. 15 Jul 97.

11. Bearse, T. "Deriving a Diagnostic Inference Model from a Test Strategy." 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

12. Ben-Bassat, M., Beniaminy, I., and Joseph, D. "Can Model-Based and Case-Based Expert Systems Operate Together?" 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

13. Ben-Basset, M., Beniaminy, I., and Joseph, D. "Different Approaches to Fault Isolation Support Software." 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

14. Biasizzo, A., Zuzek, A., and Novak, F. "Sequential Diagnostics Toll." 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

15. Bishop, C. "Neural Networks for Pattern Recognition." Oxford University Press, 1998.

16. Blemel, K. "Dynamic Autonomous Test Systems for Prognostic Health Management." Joint Strike Fighter Program Office paper. 10 Nov 98.

17. Bogert, B., Healy, M., and Tukey, J. "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum, and Saphe Cracking." Proceedings of the Symposium on Time Series Analysis, pp. 209-243, 1963.

18. Borden, A. "The Impact of Advanced Computer Systems on Avionics Reliability." Computers and Avionics. pp S7-S21.

19. Borky, J., Lachenmaier, R., Messing, J., and Frink, A. "Architectures for Next Generation Military Avionics Systems." 1998 IEEE Aerospace Conference Proceedings, pp 265-81, vol. 1.

20. Bursch, P., Meisner, J., and Winegar, K. "A PC Based Expert Diagnostic Tool." Honeywell, Inc. paper. 1988.

21. Cardona, R. "Aircraft Prognostics and Advanced Diagnostics Project: Problem Identification Task Group Study and Recommendations". Air Force Research Laboratory Logistics Readiness Branch, 1999.

22. Chu, S. "Using a Neural Network and a Statistical Classifier for Aircraft Fault Diagnostics." Human Resources Directorate, Logistics Research Division, WPAFB. Aug 1996.

23. Conway, J. and Sloane, N. "Lexicographic Codes: Error-Correcting Codes from Game Theory". IEEE Trans. Information Theory, pp 337-348, 1986.

24. Cover, T. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition." IEEE Transactions on Neural Networks, vol. 2, no. 2. 1965.

25. Dasarathy, B. Webpage at http://www.inforfusion.org/mission.htm. 1999.

26. De Montauk, J.P. "On-Board Maintenance Aids." Airbus Industrie. (private communication)

27. Dean, J. "Integrated Diagnostics: Confusion and Solutions." SA-ALC/LDAE paper. Undated.

28. Dill, H. "Pass/Fail Limits-The Key to Effective Diagnostic Tests." 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

29. Dussault, H., Clothier, R., and Ferrell, B. "Integrated Diagnostics During Design." Rome Air Development Center paper. Undated draft. (private communication)

30. Eilbert, R. and Christensen, R. "Contrivedness: The Boundary Between Pattern Recognition and Numerology." Pattern Recognition, vol. 15, no. 3. 1982.

31. Federici, D., Bisgambiglia, P., and Santucci, J. "VHDL Behavioral Fault Simulator: Experiments o (sic) the ITC'99 Benchmarks". 1999 IEEE International Workshop on System Test and Diagnosis.

32. Feldkamp, L., and Marko, K. "Model Based Controls and Diagnostics." General Electric, Ford Motor Company Final Program Review. Dearborn MI. 7 Jan 98.

33. Ferlez, R. and Lang, D. "Gear Tooth Fault Detection and Tracking Using the Wavelet Transform." Proceedings of the Conference on Prognosis of Residual Life of Machinery and Structures; 52nd Meeting of the Machine Failure Prevention Technology Society; Virginia Beach VA. 1998. pp 451-460.

34. Furnival, G. and Wilson, R. "Regressions by Leaps and Bounds". Technometrics, vol. 16, no. 4, pp 499-511, Nov 1974.

35. Girosi, F. and Poggio, T. "Networks and the best approximation property." Biological Cybernetics, pp 169-176, 1990.

36. Greitzer, F., Stahlman, E., Ferryman, T., Wilson, B., Kangas, L., and Sisk, D. "Development of a Framework for Predicting Life of Mechanical Systems: Life Extension Analysis and Prognostics (LEAP)". Pacific Northwest Laboratories paper presented at the International Society of Logistics (SOLE) 1999 Symposium, Las Vegas NV, August 30 – September 2, 1999.

37. Hadden, G., Bergstrom, P., Bennett, B., Vachtsevanos, G., and Van Dyke, J. "Machinery Diagnostics and Prognostics/Condition Based Maintenance: A Progress Report". Proceedings of the 53rd Meeting of the Society for Machinery Failure Prevention Technology, pp 367-378, 1999.

38. Hansen L. and Salamon P. "Neural network ensembles". IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp.993 - 1001, 1990.

39. Hartman E., Keeler J., and Kowalski J. "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations." Neural Computations, vol. 2, no. 2, pp 210-215, 1990.

40. Hornik, K., Stinchcombe, M., and White, H. "Universal Approximation of Unknown Mapping and its Derivatives Using Multilayer Feedforward Neural Networks". Proceedings of the Second Joint Technology Workshop on Neural Networks and Fuzzy Logic, NASA, pp 62-76, 1990.

41. Hyafil, L. and Rivest, R. "Constructing Optimal Binary Decision Trees is NP-complete." Information Processing Letters, vol. 5, no. 1, pp 15-7. 1976.

42. John, B. "Fuzzy Inferencing Systems – Problems and Some Solutions." Working paper available at http://www.cms.dmu.ac.uk/People/rij/newrep/newrep.html. Dec 1995.

43. Kangas, L., Greitzer, F., and Illi, O. "TEDANN: Turbine Engine Diagnostic Artificial Neural Network". US Army Ordnance Center and School, Aberden Proving Ground, Maryland. Presented at the Advanced Information Systems and Technology Conference 28-30 March 1994.

44. Keller, K., Holland, J., Bartz, D., and Swearingen, K. "An Advanced Onboard Diagnostic System for Vehicle Management." 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

45. Klassen, M., Pao, Y., and Chen, V. "Characteristics of the Functional Link Net: A Higher Order Delta Rule Net." IEEE International Conference on Neural Nets, vol. 1, pp 507-513, 1988.

46. Krogh A and Vedelsby J. "Neural network ensembles, Cross Validation, and Active Learning". Advances in Neural Information Processing Systems 7, pages 231-238, 1995.

47. Lane, S., Flax, M., Handelman, D., and Gelfand, J. "Multi-layer Perceptrons with B-spline Receptive Field Functions." Advances in Neural Information Processing Systems 3, pp 684-692, 1991.

48. Lee, C. "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II." IEEE Transactions on Systems, Man and Cybernetics, vol. 20(2), pp 419-435, 1990.

49. Lo, J., Van Dyck, R., Garga, A., and Hall, D. "Fault Prediction in Transmissions Using Wavelet Analysis." Proceedings of the Conference on Prognosis of Residual Life of Machinery and Structures; 52nd Meeting of the Machine Failure Prevention Technology Society; Virginia Beach VA. 1998. pp 441-450.

50. Logan, K., Galie, T., and Savage, C. "A Practical Application of Probabilistic Neural Networks to Machinery Failure Prevention." Proceedings of the 53rd Meeting of the Society for Machinery Failure Protection Technology, pp. 223-236, 2000.

51. Logan, K., Inozu, B., and Roy, P. "Shipboard Learning of Diesel Engine Operating Characteristics". Interim Report from the Gulf Coast Region Maritime Technology Center. 2001.

52. MacDonald, D. "Reducing the Maintenance Cost of the Joint Strike Fighter through Condition-Based Maintenance." SRI International briefing. 1998. (private communication)

53. Malley, M. "A Methodology for Simulating the Joint Strike Fighter's (JSF) Prognostics and Health Management System." Air Force Institute of Technology GOR/ENS/01M-11 Master's Thesis, March 2001.

54. Marko, K., Feldkamp, L., and Puskorius, G. "Automotive Diagnostics Using Trainable Classifiers: Statistical Testing and Paradigm Selection," Proceedings of the International Joint Conference on Neural Networks, 1990.

55. Marko, K., James, J., Dosdall, J., and Murphy, J. "Automotive Control System Diagnostics Using Neural Nets for Rapid Pattern Classification of Large Data Sets". Proceedings of the International Joint Conference on Neural Networks, 1989.

56. McDuff, R. and Simpson, P. "Using Neural Networks for F-16 Fault Diagnostics." Contract Extension Final Report, Electronics Division, General Dynamics. Oct 1988.

57. Meisner, J., Bursch, P., and Funk, H. "Evolution of a Maintenance Diagnostic System." Honeywell, Inc. paper. 1990.

58. Mizumoto, M. and Tanaka, K. "Some Properties of Fuzzy Sets of Type 2". Information and Control, vol. 31, pp 312--340, 1976.

59. Moody, J. and Darken, C. "Fast Learning in Networks of Locally-Tuned Processing Units." Neural Computation, vol. 1, pp 281-294, 1989.

60. National Science Foundation Workshop on "Signal Processing for Manufacturing and Machine Monitoring". Draft 1.2 of Final Report. Mar 96.

61. Opitz, D. and Maclin, R. "Popular Ensemble Methods: An Empirical Study". Journal of Artificial Intelligence Research 11 (1999), pp. 169-198. August 1999.

62. Oppenheim, V. and Schafer, R. "Discrete-Time Signal Processing." Prentice Hall, 1989.

63. Oxley, M. and Bauer, K. "Classifier Fusion for Improved System Performance." Working Paper No. 02-02, AFIT Department of Operational Sciences, 2002.

64. Pattipati, K. and Alexandridis, M. "Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis." IEEE Transactions on Systems, Man, and Cybernetics.

65. Powell, M. "Radial Basis Functions for Multivariable Interpolation: A Review." Algorithms for Approximation, pp 143-167, 1987.

66. Rabiner, L. and Schafer, R. "Digital Processing of Speech Signals", Prentice Hall, 1978. vol. 20, no. 4. 1990.

67. Rao, V. Webpage at http://avalon.epm.ornl.gov/~nrao/history.html.

68. Rebulanan, R. "Simulation of the Joint Strike Fighter's Autonomic Logistics System Using the JAVA Programming Language." Air Force Institute of Technology GOR/ENS/00M-19 Master's Thesis, March 2000.

69. Renals, S. and Rohwer, R. "Phoneme Classification Experiments Using Radial Basis Functions." International Joint Conference on Neural Networks, vol. 1, pp 461-467, 1989.

70. Rimey, R., Gouin, P., Scofield, C., and Reilly, D. "Intelligent Robots and Computer Vision: Fifth in a Series". Proceedings of SPIE – The International Society for Optical Engineering, vol. 726, pp 552-7, 1986.

71. Rolfe, R., Brown, H., Savage, H., Scalia, A., and Simpson, W. "Activities and Results of the 1996 Joint Service Integrated Diagnostics Workshop." Institute for Defense Analysis, December 1996.

72. Ross, B. "Open Systems Approach Integrated Diagnostics Demonstration (OSAIDD) Study Program Final Report." DoD Automatic Test Systems Executive Agent, Dec 98.

73. Rousseau, M. and Logan, K. "Machine Learning of Diesel Engine Operating Characteristics." Gulf Coast Region Maritime Technology Center, University of New Orleans. July 1997.

74. Rummelhart, D., Hinton, G., and Williams, R. "Learning Internal Representation by Error Propogation". Parallel Distributed Processing: Exploration in the Microstructure of Cognition, vol. I, 1986.

75. Schaefer, C. and Haas, D. "A Simulation Model for Determining the Impact of Health and Usage Monitoring Systems (HUMS) on Helicopter Maintenance and Logistics Operations." 58th Annual Forum of the American Helicopter Society, 2002.

76. Scheuren, W. "Joint Strike Fighter: Prognostics and Health Management". Internet homepage, http://www.jast.mil/html/body_phm.htm.

77. Scheuren, W. "Joint Strike Fighter: Prognostics and Health Management". Briefing presented at the 1998 Technology Showcase – Joint Oil Analysis Program, and available at http://www.jast.mil/html/body_phm.htm.

78. Scheuren, W. "Safety & the Military Aircraft Joint Strike Fighter Prognostics & Health Management." DARPA. 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit. Cleveland OH, 13-15 July 98.

79. Shahdad, et. al. "VHSIC Hardware Description Language." IEEE Computers vol. 18, pp. 94-103, Feb 1985.

80. Sheppard, J. "Introducing Information Flow Models from Case Data." 1998 IEEE International Workshop on System Test and Diagnosis. 7-9 April 1998.

81. Sheppard, J. and Kaufman, M. "IEEE Test and Diagnostics Standards." Available at http://www.dtic.mil/ndia/systems/Kaufman.pdf.

82. Smith, G., Schroeder, J., Faas, P., and Williams, G. "Prognostics-The Cornerstone of the Autonomic Logistic System." 55th Annual Forum of the American Helicopter Society, Montreal, Quebec, Canada. 25-7 May 99.

83. Smith, G., Schroeder, J., Navarro, S., and Halderman, D. "Development of a Prognostics & Health Management Capability for the Joint Strike Fighter." IEEE Systems Readiness Technology Conference, Anaheim CA. 22-25 Sep 97.

84. Specht, D., "A General Regression Neural Network." IEEE Transactions on Neural Networks, vol. 2, no. 6, pp 568-576, 1991.

85. Specht, D.F. "Probabilistic Neural Networks". Neural Networks, vol. 3, pp 109-118, 1990.

86. Specht, D.F. "Probabilistic Neural Networks for Classification, Mapping, or Associative Memory." Proceedings, IEEE International Conference on Neural Networks, vol.1, pp 525-532, 1988.

87. Stoll, F. and Vincent, P. "Advanced Diagnostics." USAF Research Laboratory, Final Report for Mar 99 - Dec 99. Wright-Patterson AFB OH. Dec 99.

88. Su, L., Nolan, M., DeMare, G., and Carey, D. "Prognostics Framework." U.S. Army Test, Measurement, and Diagnostic Activity, Advanced Technology Office paper. 1999.

89. Takagi, H. and I. Hayashi. "NN-Driven Fuzzy Reasoning." Approximate Reasoning, vol. 5, pp 191-212, 1991.

90. Theodoridis, S. and Koutroumbas, K. "Pattern Recognition." Academic Press, 1999.

91. Thesen, A. and Beringer, D. "Goodness-of-fit in the User-Computer Interface: A Hierarchical Control Framework Related to Friendliness." IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-16, no. 1. Jan/Feb 1986.

92. Trachten, A. "Lexicographic Codes". Located at http://ipsit.bu.edu/phdthesis_html/node11.html.

93. US Army Research Laboratory. "Turbine Engine Diagnostics: Diagnostics Intelligent Tutoring System – M1A1 AGT 1500 Turbine Engine." Software Package CD Version 1.5.

94. Vachtsevanos, G., Allgood, G., Goebel, K., and Hadden, G. "Fault Diagnostics/Prognostics for Machine Health Maintenance Short Course", Georgia Institute of Technology, 11-14 Sep 01.

95. Vachtsevanos, G. and Clutz, T. "Fault Diagnostics/Prognostics for Machine Health Maintenance Short Course", Georgia Institute of Technology, Atlanta GA, 11-14 Sep 01.

96. Widyantoro, D., Yen, J., Wall, J., Hanratty, T., Dumer, J., Hammel, R., and Helfman, R. "A Radial Basis Function Networks Approach in Turbine Engine Prognosis." Proceedings of the 54th Meeting of the Society for Machinery Failure Prevention Technology (MFPT'54): Improving Productivity through Applications of Condition Monitoring, pp. 95-104, 1-4 May 2000, Virginia Beach VA.

97. Zadeh, L. "Fuzzy Logic and its Application to Approximate Reasoning." Information Processing, vol. 74, pp 591-594, 1974.

98. Zadeh, L. "Fuzzy Sets". Information and Control, vol. 8, pp 338-353, 1965.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
02-04-2003	**Doctoral Dissertation**	Jun 2001 – Mar 2003

4. TITLE AND SUBTITLE

A FRAMEWORK FOR PROGNOSTICS REASONING

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

CLUTZ, THOMAS C., MAJ, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640, WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/DS/ENS/03-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM, Attn: MAJ JUAN VASQUEZ
801 North Randolph Street, Room 732 DSN: 428-8431
Arlington VA 22203-1977 e-mail: Juan.Vasquez@afosr.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The use of system data to make predictions about the future system state, commonly known as prognostics, is a rapidly developing field. Prognostics seeks to build on current diagnostic equipment capabilities for its predictive capability. Many military systems, including the Joint Strike Fighter (JSF), are planning to include on-board prognostics systems to enhance system supportability and affordability. Current research efforts supporting these developments tend to focus on developing a prognostic tool for one specific system component. This dissertation research presents a comprehensive literature review of these developing research efforts. It also develops presents a mathematical model for the optimum allocation of prognostics sensors and their associated classifiers on a given system and all of its components. The model assumptions about system criticality are consistent with current industrial philosophies. This research also develops methodologies for combining sensor classifiers to allow for the selection of the best sensor ensemble.

15. SUBJECT TERMS
Operations Research, Data Fusion, Non-linear Programming

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**Dr. Kenneth W. Bauer, Jr.**
U	**U**	**U**	**UU**	287	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4328; e-mail: Kenneth.Bauer@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18