(2)

AFWAL-TR-89-3009

HIERARCHICAL DAMAGE TOLERANT CONTROLLERS FOR SMART STRUCTURES

A.K. Caglayan, S.M. Allen, and S.J. Edwards

Charles River Analytics Inc.
55 Wheeler Street
Cambridge, MA 02138

March 1989

Final Report for Period Jun 88 - Dec 88

FLIGHT DYNAMICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6553

NOTICE

[illegible]

CHRISTOPHER J. MAZUR, Capt, USAF
Fatigue, Fracture & Reliability Gp
Structural Integrity Branch

James L. Rudd

JAMES L. RUDD, Tech Mgr
Fatigue Fracture & Reliability Gp
Structural Integrity Branch

FOR THE COMMANDER

Tony G. Gerardi

TONY G. GERARDI, Chief
Structural Integrity Branch
Structures Division

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-89-3009
6a. NAME OF PERFORMING ORGANIZATION Charles River Analytics Inc. / 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Flight Dynamics Laboratory (AFWAL/FDSEC) Air Force Wright Aeronautical Laboratories
6c. ADDRESS (City, State, and ZIP Code) 55 Wheeler Street Cambridge, MA 02138	7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6553
8a. NAME OF FUNDING/SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-88-C-3212
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO
65502F	3005	40	21

11. TITLE (Include Security Classification)
Hierarchical Damage Tolerant Controllers for Smart Structures

12. PERSONAL AUTHOR(S)
A.K. Caglayan, S.M. Allen, and S.J. Edwards

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Report	FROM 88 JUN TO 88 DEC	89 MAR	

16. SUPPLEMENTARY NOTATION
This is a Small Business Innovative Research Program Phase I report.

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Smart Structures, Damage Tolerance, EXpert Systems, Damage Detection Isolation and Assessment

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
Here, we investigate and define a smart aerospace structure baseline architecture consisting of: sensor arrays for immediate damage detection such as in situ piezoelectrics and fiber optic sensors used in conjunction with other rigid and flexible body conventional sensors; a knowledge based expert system for damage detection, isolation and assessment which capture the domain expert's expertise in diagnosing structural damage based on temporal and spatial damage signatures; an identification system which transforms the structural damage information into a model suitable for the redesign of the active controller on line; a decentralized controller which reconfigures itself on-line based on detected damage conditions and their estimated levels; an expert system implementation which operates real-time based on a knowledge compiler approach using general purpose numeric processors.

In this study, we investigate the desired architectural attributes of a smart structure

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL Capt. Christopher J. Mazur	22b TELEPHONE (Include Area Code) 513/255-6104	22c. OFFICE SYMBOL AFWAL/FIBEC

DD Form 1473, JUN 86 Previous editions are obsolete. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

(#19 continued)

in the context of a smart wing example implemented as a knowledge based expert system using the interpretive shell CLIPS. In particular, we discuss knowledge representation issues, damage detection, isolation and assessment strategies, real-time performance issues, requirements on the expert system development tool imposed by the smart structure attributes, smart structure design methodology, and present implementation details. Under the damage detection, isolation and assessment strategy discussion, we present the limitations of conventional hypothesis test formulations, expert systems based damage diagnosis techniques which alleviate these limitations by exploiting the local spatial and temporal signatures of a damage, and the applicability of artificial neural networks to smart structures.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This report summarizes the research and development results of the SBIR Phase I study entitled "Hierarchical Damage Tolerant Controllers for Smart Structures" supported by U.S. Air Force under Contract No. F33615-88-C-3212. The major aim of this study is the investigation and definition of a baseline architecture for a smart aerospace structure which can detect and isolate structural damage in real-time and provide on-line reconfiguration of the structure's control system under the detected impairment conditions. In particular, we investigate how a smart aerospace structure can be implemented as a real-time knowledge based expert system by addressing issues involved with structural knowledge representation, structural damage detection and isolation strategies and real-time performance in an embedded environment.

1.1 Summary

A smart aerospace structure which can detect and isolate structural damage in real-time and provide on-line reconfiguration of the structure's control system under the detected impairment conditions would significantly boost the reliability, maintainability and performance of current and proposed Air Force flight and space vehicles. A closely related R&D effort in this area is the Air Force Self-Repairing Flight Control System Reliability and Maintainability program supporting the development of reconfiguration strategies for aircraft subjected to actuator failure and surface damage, and maintenance diagnostics for the development of non-real time diagnostic expert systems for flight line maintenance [1]-[4].

What would be the key elements of a smart aerospace structure? First, such a smart structure would have to incorporate new innovative sensors for the fast detection of structural damage. Second, a smart structure would have to include a decentralized controller architecture adaptable for on-line reconfiguration. Third, a smart structure should include a representation of the necessary aerospace knowledge to generate damage hypotheses, reason and reconfigure based on this knowledge representation. Fourth, a smart structure has to reconfigure itself in real-time.

In the sensing area, most global rigid body sensors (e.g., accelerometer), can be used for damage detection examining the global effects of structural damage. Since it is desirable to detect failures at the lowest level of a fault tolerant hierarchy, local sensors would be the critical sensor technology in the development of smart aerospace structures. The sensing technology advancement would require increasing the accuracy of current techniques such as hinge moment sensing for control surfaces and the development of innovative new sensing and actuation arrays such as piezoelectrics and fiber optic sensors built into the structure. In addition to providing damage detection information, some of these sensing arrays can also be used for actuation without the interaction effects common to conventional actuators mounted on structures.

The baseline architecture of a smart aerospace structure has to offer a decentralized solution in order to deal with dispersed sensor and actuator locations, constraints on information transfer, and distribution of computing resources for survivability considerations. We see the specification of a decentralized information structure as one of the major technical problem to be resolved. In such a decentralized baseline architecture, damage detection and active control become decentralized failure detection and control problems. Hence, current techniques on surface damage detection and decentralized large scale system control techniques would be the starting points for the smart aerospace architecture. Finally, the structural damage detection problem is linked to sensor and actuation failure detection since any structural damage would impair nearby sensing and actuation devices as well.

Another critical technical requirement for a smart aerospace structure is the representation and incorporation of aero knowledge into the architecture for use in damage detection and reconfiguration. Here, the issue is the generation of embedded expert system code which can meet the memory size and execution time contraints of embedded processors. Hence, the decentralized damage detection and active control system should be built using artificial intelligence methods so that based on observed symptoms and underlying structural and aerodynamic knowledge, the smart structure can form, test and make decisions about damage hypotheses, transform the structural damage information into a model usable by the active control system, and reconfigure the control law on-line to compensate for the effects of the damage.

Finally, the real-time requirements on damage detection and controller reconfiguration precludes the use of current generation general purpose artificial intelligence computers. Hence, the development of real-time expert systems implemented on current flight and space qualified numeric processors using conventional embedded programming languages (e.g., Ada) becomes a critical issue.

Here, we investigate and define a smart aerospace structure baseline architecture consisting of:

- sensor arrays for immediate damage detection such as in situ piezolectrics and fiber optic sensors used in conjunction with other rigid and flexible body conventional sensors
- a knowledge based expert system for damage detection, isolation and assessment which capture the domain expert's expertise in diagnosing structural damage based on temporal and spatial damage signatures

- an identification system which transforms the structural damage information into a model suitable for the redesign of the active controller on-line
- a decentralized controller which reconfigures itself on-line based on detected damage conditions and their estimated levels
- an expert system implementation which operates real-time based on a knowledge compiler approach using general purpose numeric processors.

In this study, we investigate the desired architectural attributes of a smart structure in the context of a smart wing example implemented as a knowledge based expert system using the interpretive shell CLIPS. In particular, we discuss knowledge representation issues, damage detection, isolation and assessment strategies, real-time performance issues, requirements on the expert system development tool imposed by the smart structure attributes, smart structure design methodology, and present implementation details. Under the damage detection, isolation and assessment strategy discussion, we present the limitations of conventional hypothesis test formulations, expert systems based damage diagnosis techniques which alleviate these limitations by exploiting the local spatial and temporal signatures of a damage, and the applicability of artificial neural networks to smart structures.

1.2 Outline of the Report

The outline of the report is as follows. Chapter 2 contains the formulation of the problem. In particular, we present an overview of the relevant sensing technology, controller structure, damage detection algorithms, artificial intelligence methods and artificial neural networks for smart structures. Moreover, we discuss the issues involved in utilizing these techniques in building smart structures in general terms. In Chapter 3, we present the desired architectural attributes of a smart structure in the context of a smart wing knowledge based expert system. In particular, we discuss the technical issues involved in building smart structures: knowledge representation, damage detection, isolation and assessment strategies, real-time performance in an embedded environment and required expert system development tools. The report ends with Chapter 4 which contains the conclusions and recommendations.

2. FORMULATION OF THE PROBLEM

2.1 Sensors for Smart Structures

In the development of a reconfiguration strategy for aircraft subjected to actuator failure and surface damage, rigid body flight control accelerometer and rate gyros, indicated airspeed, angle of attack and sideslip indicators, and actuator command and surface position sensors have been used for control surface damage detection algorithms [2]. One of the technology gaps identified is the need for accurate hinge moment sensors since the current hinge moment accuracy does not allow the incorporation of these hinge moment sensors into monitoring algorithms [5].

In addition to the conventional structure/sensor architectures, there is an increasing need for "smart materials" in space structures as well as in airframe applications. Such materials would contain sensors and control elements as integral parts of their structures to provide detection, and perhaps prevention or correction, of imminent failure or misalignment. One such possible approach is the in situ formation of piezoelectric thin films on the surface of existing light-weight structural materials such as aluminum alloys [6]-[11]. Piezoelectric material is a natural electromechanical transducer. When an electric field is applied to such a material, a strain is produced in it. Conversely, if the material is stressed mechanically, an electric field is generated.

Polymeric materials with piezoelectric properties such as polyvinylidene fluoride can be deposited as thin films are potential candidates for such control materials. However, polymer materials typically have limited lifetimes in the radiation and vacuum environment of space. In contrast, ceramic piezoelectric materials have much greater stability. However, they are less easily formed into thin films, particularly on existing large structures. Recently, significant advances have been made in using organic precursors such as alkoxides to produce piezoelectric ceramics such as barium titanate.

Fiber optics can also be used by embedding a fault tolerant fiber optic network into the structure and using fault tolerant network techniques to detect and isolate structural damage. In such a fault tolerant network environment, there are several available routing algorithms to detect and isolate the malfunctioning nodes and paths for a given network topology. In addition, it is possible to measure physical variables of interest in structural applications using optic fibers. For instance, measurements of both static and dynamic strain have been reported using optical fibers embedded in composite materials [12].

Other potential sensors for damage detection in smart aerospace structures include computer vision. Here, we see the development of a TV based vision system for the detection of structural damage using, for instance, edge detection algorithms.

2.2 Decentralized Control for Smart Aerospace Structures

The active control of airframe, and flexible space structures in particular, under structural damage will require decentralized control. The decentralization constraint enters into large scale systems because it may be impractical or even impossible to communicate signals from one controller to another, due to for instance, data-bandwidth constraints. Moreover, decentralization may be required by the control designer to achieve reliability and survivability with graceful degradation under failures, imposing a control structure in which control authority is relegated to separate levels. An equally important aspect is the possibility of relief from extreme computational constraints encountered in the design of a centralized controller for the system. Since each control agent is responsible for producing only a subset of the inputs while using restricted information, the overall dynamic system essentially reduces to a collection of smaller subsystems for which controllers can be generated with more computational ease than for a single larger-order system.

Early results in optimal decentralized control theory were of a negative nature. The separation of estimation and control is not optimal for linear decentralized control systems with quadratic cost and Gaussian disturbances. In fact, the optimal control is not necessarily a linear feedback law, as noted by Witsenhausen [13]; in general, the existence of an optimal law is not guaranteed [14] for decentralized systems, and the optimal linear law can be of infinite dimensions, and hence not realizable [15]. Questions as to the importance of "signalling," "second guessing" and the extent of "cross-communication" among decentralized controllers (which do not arise in centralized systems), remain to be answered.

Recently, more successful results have been obtained by extending the standard LQG control and estimation algorithms to decentralized systems with fixed information exchange patterns [16]-[22]. For instance, the solution to the decentralized LQG control problem without a central supervisor has been obtained by Speyer in [21] by hypothesizing a network structure consisting of interconnected nodes where the local filter's estimate at a given node is fused with incoming data from other nodes to compute the globally optimum estimate. The main benefits of this structure are parallel processing and minimal required bandwidth for data transmission. Another class of decentralized estimators results from the application of perturbation techniques such as a structure consisting of a set of locally optimal estimators for the individual subsystems driven by compensatory signals of a global estimator on a higher level to account for the interconnection effects [22].

In summary, decentralized control using fixed information patterns appear to be the most feasible approach for smart aerospace structures. The fact that this formulation does not address the question of what structure should be assumed may be considered a drawback in general decentralized system applications. However, this aspect would not be a problem in aerospace structure applications due to the large known body of knowledge about the desired subsystems.

Since Large Space Structures (LSS) are infinite dimensional distributed parameter systems, their finite order linear system approximations are used for controller designs. These active LSS shape and vibration control methods based on modal truncation of the structural modes suffer due to control and observation spillover effects [23]. Spillover is the coupling between the controller and the unmodelled dynamics within the bandwidth of the controller. The destabilizing effects of spillover are more pronounced at higher modes where the modelling uncertainty is higher for structures with no closed-form solution for the eigenvalue problem. In the last decade, a significant amount of control R&D effort has been expended to address these unique problems associated with the control of large space structures such as decentralized control [24]-[27], adaptive control based on identification techniques [28], and robust reduced order controllers [29].

In contrast, colocated local direct velocity feedback, has been shown to be unconditionally stable under some assumptions about the actuator dynamics [24]. Recently, positive position feedback has been suggested as an alternative to local velocity feedback [12]. This method is also not sensitive to spillover and stable under minimally restrictive conditions. Hence, local velocity/position feedback control algorithms would be ideal for smart structure applications. However, our proposed decentralized smart structure concept would handle arbitrary decentralized information patterns including non-colocated sensors and actuators.

2.3 Damage Detection Algorithms

The analytic redundancy approach to failure detection and isolation (FDI) problems in dynamic systems uses the analytic relationships between various sensor outputs arising from a knowledge of the underlying system dynamics. Analytic redundancy can be either in the form of algebraic redundancy -- the instantaneous relationship between sensor outputs, or dynamic redundancy -- the relationship between the time histories of sensor outputs. The term "analytic redundancy" was coined in the early seventies to differentiate this technique from the traditional hardware redundancy approach in which the outputs of like sensors are compared for failure detection. Analytic redundancy comes about from the common estimation capability of various sensor groups. Sensor FDI algorithms make use of this inherent analytic redundancy by considering different sensor subsets. Hence, the analytic redundancy approach offers the capability of comparing dissimilar instrument outputs for failure detection and thus, allows the design of reliable systems with reduced hardware duplication requirements.

Over the last fifteen years, several failure detection and isolation algorithms applicable to damage detection problems for smart structures have been developed based on dynamic system descriptions. Such analytic redundancy research culminated in the development of aircraft sensor fault tolerant digital flight control systems such as the USAF DIGITAC A-7 and the NASA/LaRC F8-DFBW application [31]-[32], engine sensor failure detection systems such as the NASA/LaRC F100 application [33], strapped down navigation systems with skewed sensor arrays such as the NASA LaRC RSDIMU [34], sensor fault tolerant integrated flight control and navigation systems such as the NASA/LaRC FINDS application [35], and more recently in reconfiguration strategies for aircraft subjected to actuator failure/surface damage for the USAF Self-Repairing Flight Control System Reliability and Maintainability program [36].

General failure detection and isolation methods for dynamic systems can be divided into the following groups: a) parity techniques; b) failure sensitive filters; c) jump parameter formulations; d) multiple model methods; and e) innovations signature analysis. Parity techniques [37] encompass the standard voting techniques for systems with parallel hardware redundancy and their generalizations to systems with algebraic redundancy. Failure sensitive filters exploit output decoupling concepts are used to make failures manifest themselves in a fixed direction within the measurement space [38]. In jump parameter formulations, failures are modeled as jumps in system parameters [39]. Multiple model methods are based on constructing a different model for each failure mode [40]. Innovations signature analysis techniques include performing statistical tests on measurement innovations [41].

Recent work has concentrated on extending the signature analysis methods to nonlinear dynamic systems with modelling uncertainties. For instance, the research by Caglayan and Rahnamai [2] resulted in a hierarchical actuator failure/surface damage detection and isolation system based on the modification of the multiple hypothesis test to account for modelling errors in aircraft dynamics. In this approach, isolation thresholds are introduced in addition to detection thresholds so that the algorithm first declares a detection decision with a partial isolation decision, and a full isolation decision only when a certain distinguishability criterion is met. The performance of this developed actuator failure/surface damage detection and isolation system has been demonstrated using the nonlinear AFTI F16 and Combat Reconfigurable Control Aircraft dynamic simulators.

The advantages to using analytic redundancy in reliability improvement will be more pronounced in developing damage detection algorithms for large space structures (LSS). Due to weight limitations, these structures will be very flexible and, therefore, the suppression of the structural vibrations by active control will be necessary for the critical substructures. Furthermore, the proposed functional uses will dictate stringent design specifications for shape control and pointing accuracy. Consequently, the overall reliability design of the control hardware deployed in LSS will be critical in satisfying the vibration suppression, shape control and pointing accuracy specifications. These requirements, along with the weight and volume limitations of the sensors and actuators which can be transported into space, will necessitate the use of analytic redundancy in the design of fault tolerant control systems for LSS. The automated diagnostics capability provided by the analytic redundancy approach will be especially important in unmanned structures such as the large flexible antennas. In addition, the reliability provided by an analytic redundancy based scheme would be indispensable in an application where the accuracy requirement is critical, e.g., the pointing accuracy of a microwave energy transmitter.

The current failure detection and isolation technology developed for the conventional rigid aircraft will not be directly applicable to large space structures. First, these structures have infinite dimensionality since they are distributed parameter systems. In practice, their description will be approximated, through modal truncation, by a large dimensional linear system. Therefore, current methods for low order systems will be unsuitable for LSS due to the increased system order. The second challenge in LSS will be the inadequacy of the model used for the system description, due both to truncation and to a lack of knowledge about the parameters of the truncated model. For instance, the application of the generalized parity technique to a simple flexible beam problem has shown that this approach is very sensitive to system parameter errors [42]. Again, the current methods which may be adequate for low dimensional systems with fairly known parameters, e.g. a conventional rigid body aircraft will prove unsuitable for LSS due to large false alarms caused by the parameter modeling errors [43]. Since on-earth testing for LSS will be extremely limited, these limitations should be incorporated into the design.

2.4 Artificial Intelligence Methods for Smart Aerospace Structures

The recent success of expert systems technology [44], a subfield of Artificial Intelligence, in diagnosis and monitoring problems in certain application domains has initiated similar efforts in onboard flight monitoring and diagnosis as well. Successful expert system applications such as SOPHIE in computer assisted instruction [45], MYCIN in medical diagnosis [46], PROSPECTOR in oil exploration [47], DENDRAL in biology [48] initiated onboard real-time expert system developments such as Experimental Expert Flight Status Monitor (EEFSM) in flight control systems monitoring [49], Faultfinder in onboard aircraft diagnostics [50], and expert systems for self-repairing flight control system maintenance diagnostics [4].

An expert system is a computer program that can perform a task normally requiring the reasoning ability of a human expert [44]. Expert systems are highly specialized according to their application domains. Although any program solving a particular problem may be considered to exhibit expert behavior such as computer programs implementing the damage detection algorithms described in the last section, expert systems are differentiated from other programs according to the manner in which the application domain specific knowledge is structured within a program. In particular, expert system programs partition their knowledge into the following three blocks: Data Base, Rule Base, and Inference Engine.

In other words, the knowledge about the application domain is compartmentalized rather than distributed throughout the program. The Data Base contains the facts about the application domain. The Rule Base contains the set of rules specifying how facts in the Data Base can be combined to generate new facts and form conclusions. The Inference Engine determines the construct of reasoning in the application of the rules. For instance, the diagnostic system MYCIN starts from the symptom facts in order to find the conditions causing the symptom. This manner of reasoning is called "backward chaining." In contrast, "forward chaining" inference starts with the established facts to find a set of consistent conclusions. The partitioning of application domain knowledge in expert systems allow the incremental addition of rules to the Rule Base without major revisions to the program. Moreover, the expert system can explain the reasoning chain by recording the rules as they are applied.

While expert systems have been traditionally built using collections of rules based on empirical associations, interest has grown recently in knowledge-based expert systems which perform reasoning from representations of structure and function knowledge. For instance, an expert system for digital electronic systems troubleshooting is developed by using a structural and behavioral description of digital circuits [51]-[54]. Qualitative process theory [55]-[58] is another approach allowing the representation of causal behavior based on a qualitative representation of numerical knowledge using predicate calculus.

QP theory is a first order predicate calculus defined on objects parameterized by a quantity consisting of two parts: an amount and a derivative, each represented by a sign and magnitude. In Qualitative Process theory, physical systems are described in terms of a collection of objects, their properties, and the relationships among them within the framework of a first c-der predicate calculus.

In applying QP theory to physical dynamic systems such as structural damage detection problems, the bottoms-up approach in getting the qualitative rules from low levels of elemental descriptions can possibly yield erroneous results at higher levels [59]. In contrast, finding qualitative rules at high levels using a complete knowledge of the system via reduced order modelling would not be susceptible to such problems.

For fault diagnosis in digital circuit applications, Davis advocates reasoning from first principles starting with simple hypotheses, keeping track of simplifying assumptions made, and using multiple representations (e.g., both physical and functional representation of a digital circuit) [51-54]. Multiple representation approach is analogous to Rasmussen's hierarchical knowledge representation at several levels of abstraction [60] used in modelling human problem solving strategies for complex systems.

Rasmussen introduces an abstraction hierarchy in modelling human fault diagnostic strategies. This hierarchy is two dimensional. The first is the functional layers of abstraction for the physical system: functional purpose, abstract function, generalized function, physical function, and physical form. The second is the structural layers of abstraction for the physical system: system, subsystem, module, submodule, component. Using a qualitative approximation method based on a simplified version of such a functional hierarchy, a training system for marine engineers of a steam power plant has been developed by Govindaraj [59].

The aerospace flight and space application domains introduce the attribute of dynamics not encountered in early applications of expert systems.

This necessitated the inclusion of a dynamics knowledge into expert systems implementations [62]-[63].

Current commercially available expert system building tools (shells) are not generally applicable to building expert systems for onboard fault diagnosis applications due to the following reasons [64]: 1) the shells are not fast enough; 2) the shells have insufficient facilities for temporal reasoning; 3) the shells are not easily embeddable into conventional high level programming languages and most cannot run on numeric microprocessors used for embedded applications; 4) the shells have insufficient facilities for devoting attention to significant events; 5) the shells are not designed to accept onboard sensor data; 5) the shells have no integration with a real-time clock and do not handle hardware interrupts; and 7) the shells cannot provide guaranteed response times.

As discussed in [65], most interpretive expert system shells spend 90% of their time in matching the current facts against the antecedent of rules in their rule base. Hence, an expert system development approach where the interpretive processing is performed off-line would offer a substantial execution time improvement [67]. Similarly, the execution efficiency is a strong function of the knowledge representation facilities employed in the expert system shell. For instance, an approach based on multiple hierarchical representations of a physical system and using forward chaining would have a linear execution time complexity as compared to a rule based system with forward chaining having exponential time complexity.

For ease of integration into conventional high level programs, programming language of the expert system shell is an important choice. For instance, the choice of a programming language commonly used for embedded application such as Ada or C would be advantageous from an integration viewpoint. Moreover, such an expert system would be easily portable to microprocessors commonly used for embedded applications (e.g., 1750A, 80386, 68020). Moreover, the language constructs for handling real-time issues (tasking, interrupt servicing, exception handling) would be available to such an expert system development tool.

As discussed by Laffey et al [64], there are two formal definitions for real-time expert systems: the expert system is said to exhibit real-time performance if a) it is predictably fast enough for the process being served, or b) if it can provide a response within a given time limit.

For real-time fault diagnosis applications, even if an expert system satisfies both of these premises, it would still not be sufficient for inclusion in an embedded application since the quality of the response would determine its inclusion in an onboard time-critical system. We believe that the following is an appropriate criterion for real-time expert systems for embedded applications: an expert system is said to exhibit real-time *performance if the execution* speed of a standalone compiled version of the expert system for a fixed application is comparable to the speed of a real-time conventional program written to solve the specific application at hand.

The integration of expert systems technology into time-critical applications presents new challenges due to the unique attributes of these applications. For instance, most expert systems have usually been implemented as standalone computer programs that presuppose a high degree of human interaction will be available during the problem solving process. While this approach is quite satisfactory for many naturally interactive applications, immediate human interaction is neither available nor desirable in time-critical smart structure applications. Similarly, the powerful explanation feature of the inference mechanism in expert systems is also neither required nor desirable during on-line execution in these applications.

2.5 Neural Networks for Smart Structures

Neural networks [68] represent nonalgorithmic class of information processing for using massively parallel distributed processing architectures. Stimulated by the efforts directed at understanding the interconnection of neurons in the human brain allowing the storage, retrieval, and processing of complex data, research over the last 25 years in artificial neural systems has produced solutions to complex problems in visual pattern recognition, combinatorial search, and adaptive signal processing.

The most common artificial neural net structure is a network of processing elements (neurons) connected with each other through interconnects (information links). Each processing element can have multiple inputs and only one output. The input/output relationship is described by a first-order differential equation. Specifically, a weighted sum of the nonlinear transformations of the multiple inputs along with a nonlinear transformation of the current neuron's state are the driving functions of this first-order differential equation. The weighting coefficients also satisfy a first-order differential equation driven by a nonlinear transformation of multiple inputs and the weighting coefficients associated with the input links. These two first-order differential equations (also called the update and learning rules) and a specific network topology [69] provide a complete neural network specification.

Artificial neural networks produce a nearest-neighbor classifier. Since the weighting coefficients change in an unpredictable manner, the global stability of the neural network description is an important consideration. The strongest theoretical result to date is due to Cohen and Grossberg [70] who have shown that the neural net converges to one of the finite set of equilibrium points corresponding to local minima of the energy function under certain restrictive conditions (e.g., symmetric, positive weighting coefficients). Another advantage of neural nets is that the convergence to the answer is independent of the number of local minima in the energy function, thus comparing favorably to other general search techniques. Although the global stability and convergence results have not been extended to the case for nonsymmetric weighting coefficients, several successful heuristic applications with nonsymmetric weighting coefficients have been reported [69], [70], [72].

Conceptually, artificial neural networks are applicable to smart structures for learning spatiotemporal stress patterns associated with an undamaged structure for use in detecting structural damage. The relationship between artificial neural network approach to spatiotemporal pattern recognition and conventional signal processing structures (i.e., finite impulse filters, correlation detectors, etc.) has been discussed by Myers [73]. Used in this context for smart structures, neural networks would be the nonalgorithmic counterpart of spatiotemporal filters. It is also conceptually possible to train a neural network representation of a structure with a multitude of damaged structure scenarios isolating a specific damage using the neural net as a classifier. The relative advantages and disadvantages of this approach are discussed on the next section.

The recent interest in artificial neural networks is due to the availability of fast, relatively inexpensive computers made possible by advances in VLSI design for realizing neural network structures. Given that the neurons in the human brain process information in milliseconds while outperforming current serial supercomputers with a processing rate in nanoseconds, there is considerable interest in the new generation neurocomputers and computing environments.

3. DESIRED ARCHITECTURAL ATTRIBUTES OF A SMART STRUCTURE

In this chapter, we discuss the desirable attributes of a baseline architecture for a smart structure example involving damage detection and isolation of structural damage for an airplane wing. The objective of the smart wing is to detect and isolate structural damage on a wing and to assess the impact of the identified damage. Here, we discuss the various issues involved in implementing such a knowledge based expert system and in incorporating neural networks into such an intelligent system.

3.1 An Example - Smart Wing

The smart wing expert system example has been implemented as a rule-based expert system using CLIPS (Giarratano 1987). CLIPS - C Language Integrated Production System - is a tool for the development of rule-based expert systems. CLIPS provides a powerful rule syntax and an inference engine based on the Rete match algorithm [66]. We have selected CLIPS for this example since it is written in C, embeddable to other programs written in different languages (C, Ada, FORTRAN), and portable across various hardware platforms.

Figure 3.1 shows the actual and the idealized multi-cell box beam wing structures used in simulating and designing the example expert system. As seen from 3.1, the idealized wing has a symmetric and taperless cross section. Since the main objective of the example is the analysis of knowledge representation, damage detection and isolation strategies and real-time performance issues, the model complexity has been kept to a minimum by making the following assumptions. The wing has no twist. Flanges (longitudinal stiffeners) carry only axial loads whereas wing skin and webs carry only shear loads. Furthermore, the direct stress over the flange cross section and the shear stress over the web skin cross section are assumed to be uniform.

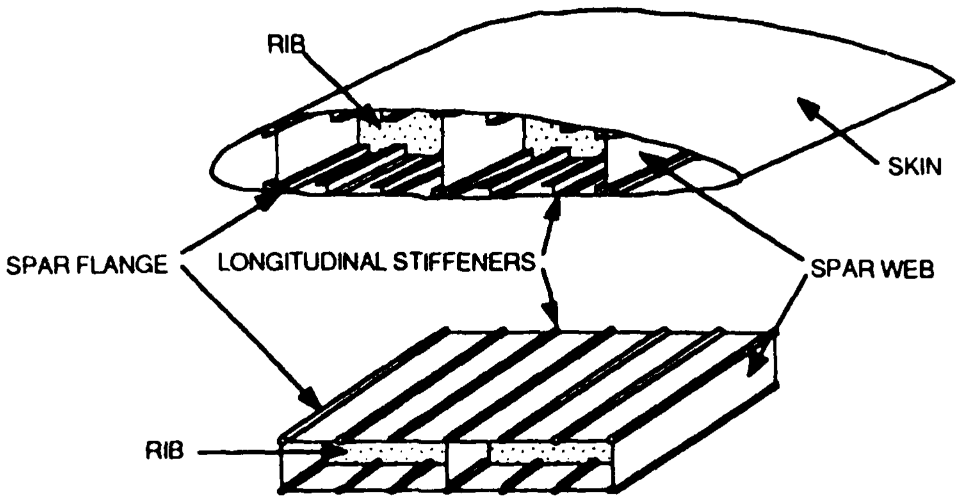

Figure 3.1: Actual and Idealized Wing Structure

Figure 3.2 shows a typical wing geometry. As seen from the figure, a strain sensor is embedded onto each subsection of a flange between the ribs. This sensor array geometry is selected because of its simplicity. In a real application, the sensor array geometry would be selected after going through a structural analysis in order to cover the critical areas (e.g., inboard leading edge area, trailing edge surfaces, etc.). The type of sensing technology (e.g., discrete vs. continuous strain sensing) used would also impact the selected geometry. Figure 3.2 also shows the location of the hydraulic line. A smart structure would also be expected to reason about the damage interior to the wing such as hydraulic lines, fuel tanks based on the observed and assessed external damage.

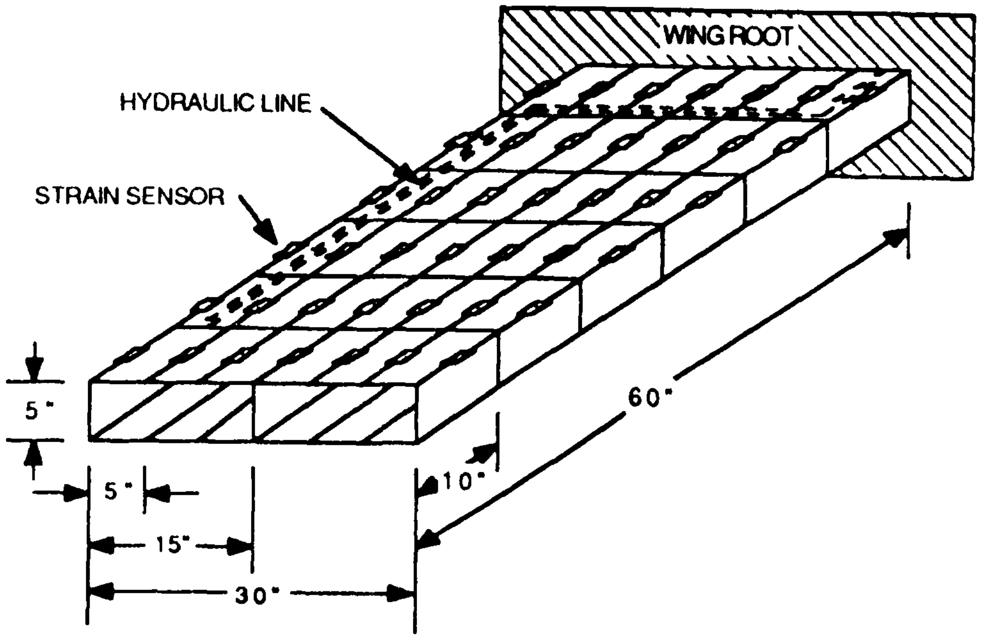

Figure 3.2: Wing Geometry

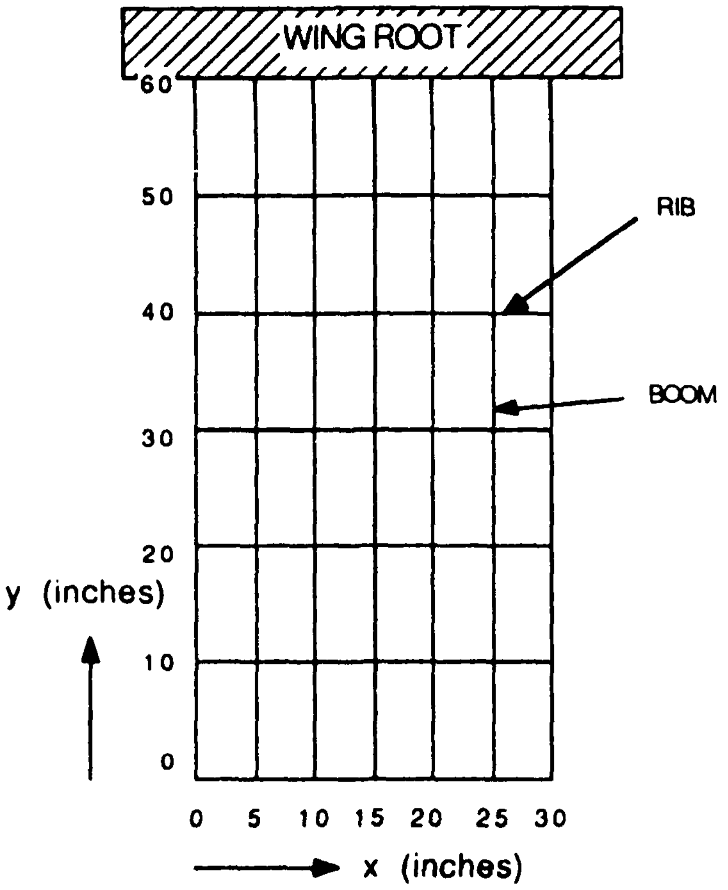

Figure 3.3: Wing Coordinate System

Table 3.1 shows the CLIPS fact defining the wing geometry. Data must be in the fact-list in order for rules to have access to them. The deffacts statement must be followed by a name, in this case "wing-geometry." The first four facts describe the position of the wing in the cartesian coordinate system used in this example (Figure 3.3). The next fact specifies the total number of booms (14) in the wing and cross-sectional area of each boom (0.75) in square inches. The next fact states that the wing is five inches thick, has a chord length of 30 inches, and a span of 60 inches. The next fact, (delta-booms 5), describes the distance between the booms in inches. The fact, (delta-measure 10 starting-at 5), describes the distance between the strain measurement sensors and the starting location. That is, the sensors are located on each boom, ten inches apart starting at five inches from the wing tip. The idealized wing is a box beam with two cells. This is described by the fact, (cell number 2). The fact, (webs thickness 0.05), specifies the thickness of the webs and skin of the wing. Finally, the last fact describes the location of the hydraulic line. The hydraulic line has three main sections and this fact has three sets of coordinates to identify the location of each section.

Table 3.1: CLIPS Fact Defining Wing Geometry

```
(deffacts wing-geometry
  (max x 30)                           ;max x location (inches)
  (max y 60)                           ;max y location (inches)
  (min x 0)                            ;min x location (inches)
  (min y 0)                            ;min y location (inches)
  (booms number 14 area 0.75) ;total number of booms, x-sec area [sq.in.]
  (wing thickness 5 chord 30 span 60) ;in inches
  (delta-booms 5)                      ;distance between booms
  (delta-measure 10 starting-at 5)     ;location of measuring devices
  (cells number 2)                     ;number of cells in box beam
  (webs  thickness 0.05)               ;in inches

  ; Below is the starting and ending locations of the hydraulic line.
  (hydraulic-line x start  2.5 end  2.5  y start 10.0 end 52.5
                  x start  2.5 end 27.5  y start 52.5 end 52.5
                  x start 27.5 end 27.5  y start 52.5 end 60.0))
```

Table 3.2 shows the CLIPS rule asserting the generation of wing booms (longitudinal stiffeners) into the fact list. The rule named "generate-objects-booms" is used to generate the facts called "booms,"

```
(booms top    x ?x-location y ?y-location meas 0.0 old est 0.0 old),
```

and place them in the fact-list. "Booms" are multi-field facts that contain information describing the current status of the boom (longitudinal stiffener, flange) at each sensor location. The first field identifies the surface that a boom is on, either top or bottom. The next four fields represent the x and y sensor location being considered. The next three fields, meas 0.0 old, describe the strain sensor measurement value and the status of that measurement. Since this rule is used for the initial generation of these facts, the value of the sensor measurement is set to 0.0 and the status is set to old. When the simulation is run, a rule will replace the 0.0 with a calculated measured value and change the status to updated. Likewise, est 0.0 old, describes the estimate of the sensor measurement and the status of the estimate.

The first pattern on the left hand side of "generate-objects-booms," (generate-booms), is used to control when this rule will fire. Before this pattern will be satisfied an initialization rule must fire to place (generate-booms) in the fact list.

There are a total of 84 sensors located on the booms of the wing. A "booms" must be generated for each one and placed in the fact-list. This is accomplished by incrementing the x location until the maximum value is reached and then incrementing the y location until its maximum value is reached and generating a "booms" at each location. The next three patterns,

```
?counter-x <- (x-counter ?x-location)
(wing thickness ? chord ?wing-chord span ?wing-span)
(test (<= ?x-location ?wing-chord))
```

check the x location to see if it is less than or equal to the chord of the wing. If it is not, then this rule does not fire and the rule that increments the y location. If it is, then these patterns are satisfied and the next three patterns are considered:

```
(y-counter ?y-location)
(delta-measure ? starting-at ?start)
(test (<= ?y-location (- ?wing-span ?start)))
```

These patterns check the y location to see if it is less than or equal to the wing span. If it is not then all of the "booms" have been generated and the next rule on the agenda will fire. If it is then these patterns are satisfied and this rule will fire and generate a new "booms" fact at ?x-location ?y-location. The last pattern,

```
(delta-booms ?distance-between-booms),
```

must also be in the fact-list before this rule will fire. However, since this fact was part of the deffacts statement named "wing-geometry" it is known to be in the fact list. When this rule fires the four actions on the right hand side of the rule are taken:

```
(retract ?counter-x)
(assert (booms top    x ?x-location y ?y-location meas 0.0 old est 0.0 old))
(assert (booms bottom x ?x-location y ?y-location meas 0.0 old est 0.0 old))
(assert (x-counter =(+ ?x-location ?distance-between-booms)))).
```

The first action taken is to remove the fact representing the current x location from the fact-list. This is accomplished with the statement,

```
(retract ?counter-x).
```

In order to retract a fact it must first be associated with a variable on the left hand side of the rule. The second pattern in this rule,

```
?counter-x <- (x-counter ?x-location),
```

associates the variable ?counter-x with the fact (x-counter ?x-location). The second and third actions taken are to generate the object boom and assert them onto the fact-list. The fourth and final action is to increment the x location counter by the ?distance-between-booms and assert it onto the fact-list.

Table 3.2: CLIPS Rule for Generating Booms

```
;This rule asserts the objects 'booms' into the fact list
;
(defrule generate-objects-booms
  (generate-booms)

  ?counter-x <- (x-counter ?x-location)
  (wing thickness ? chord ?wing-chord span ?wing-span)
  (test (<= ?x-location ?wing-chord))

  (y-counter ?y-location)
  (delta-measure ? starting-at ?start)
  (test (<= ?y-location (- ?wing-span ?start)))

  (delta-booms ?distance-between-booms)
=>
  (retract ?counter-x)
  (assert (booms top    x ?x-location y ?y-location meas 0.0 old est 0.0 old))
  (assert (booms bottom x ?x-location y ?y-location meas 0.0 old est 0.0 old))
  (assert (x-counter =(+ ?x-location ?distance-between-booms))))
```

Figure 3.4 shows one possible damage simulation scenario considered in this study: two holes of arbitrary size on the top and bottom of the wing at two possibly different locations. Figure 3.5 shows the circularity assumption made in simulating and diagnosing the damages. The objective of the smart wing expert system is to detect and isolate the damage, to find out the impaired structural elements, and to assess the impact of the damage on residual strength. The smart wing expert system performs its reasoning based on its knowledge of the wing structure and the strain sensor array and concentrated external load measurements. In the damage diagnosis section, we outline reasoning methods based on the computing residuals between the actual strain measurements and the strain estimates computed using the external load measurement and structural knowledge and other methods which do not require the measurement of any external load.

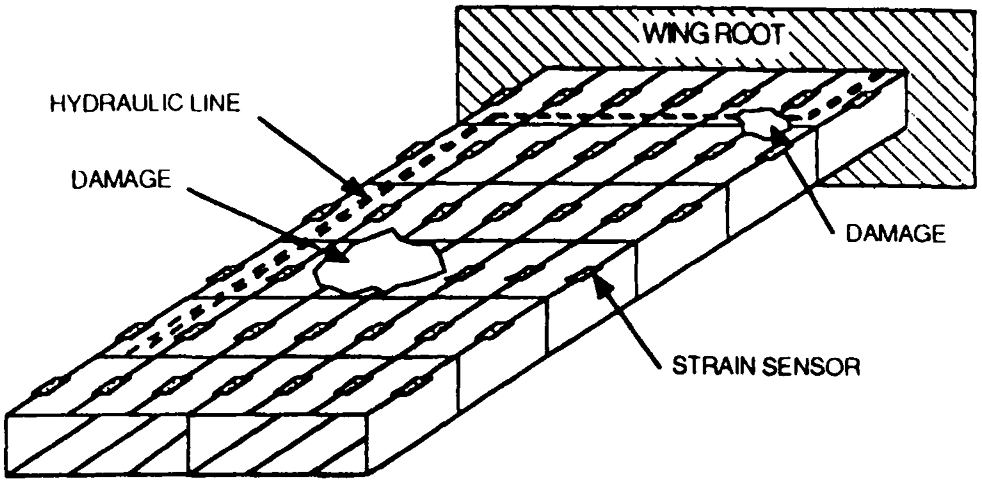

Figure 3.4: Wing Damage Scenario

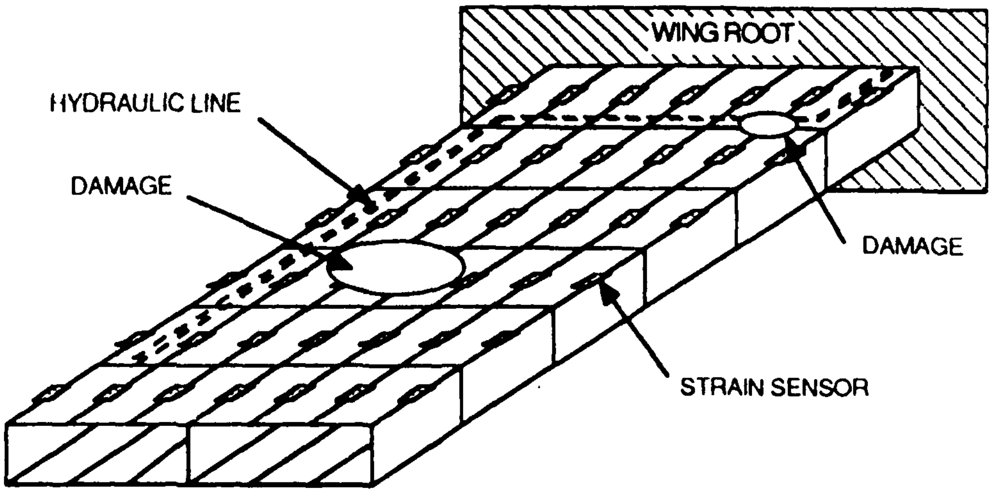

Figure 3.5: Wing Damage Model

3.2 Knowledge Representation Issues

Procedural information - commonly used to represent sequences of actions and algorithms - is usually realized as mostly linear arrangements of instructions. Traditional computer languages (Ada, C, FORTRAN) are examples of knowledge representation tools using primarily procedural information strategies. These languages work well for those problems that can be treated in an easily reducible, one step at a time, sequential approach.

Topological information - commonly used to represent the relationships among multiple entities in a fixed but arbitrary arrangement - is usually realized as a graph structure composed of a set of nodes with arcs forming the various interconnections of the nodes. Traditional representations of topological knowledge include both graphical approaches and computer generated network data structures. Graphic hardcopy diagrams can be easily made, but are not easily represented by traditional computer languages. It is possible to dynamically produce data structures to represent topological information, but the design and development needed for this task is often time consuming.

As typified by the simple smart wing example, smart structures require a hybrid knowledge representation allowing both structural declarative knowledge and sequential procedural knowledge. For instance, in the smart wing example, the description of the physical interconnection between the wing rib, flange and web, longitudinal stiffener, skin elements and the location of sensing arrays on these structural members, and the relative location of other hardware housed inside the structure (e.g., hydraulic lines, electrical wires, fuel tanks, etc.) requires a topological knowledge representation capability. Such a symbolic representation is ideally suited for an expert system implementation. In contrast, an estimation algorithm for predicting the strain at a given location performed at each sampling interval requires a procedural knowledge representation facility. Ideally, a real-time expert system shell should support both of these knowledge representation facilities.

Another desired knowledge representation construct is the hierarchical representation of a physical system at several levels of abstraction. Such a structural knowledge representation facility is usually available in hybrid expert system shells which allow object definitions with inheritance relationships. In this example, such a hierarchical representation of a smart wing can take the following form: At the highest level, the wing can be described by wing sections in between the ribs. At this level, only representative sensors can, for example, need to be monitored. At the next lower level of hierarchy, wing subsection can be further decomposed into spar web and flanges, booms, etc.

The hierarchical representation of a wing is not restricted to the physically identifiable partitioning of the wing structure described above. For example, at a higher level over the interconnected physical element structure, the wing may be described by a set of modal dynamics. Such a modal representation may be used to detect the presence of a damage using conventional rigid body sensors and would initiate reasoning based on the interconnected element description at the lower level to isolate the damage using sensor arrays to a given structural element. Similarly, a level below the interconnected physical element description, a finite element description of the wing elements can be used to further assess the magnitude of a given detected and isolated damage condition.

Apart from the evidence of similar diagnosis strategies employed by humans, such a hierarchical representation would enable a faster reasoning mechanism than a flat description where all elemental dynamic objects have to be tested at each iteration. Moreover, in such an inference tree, a failure declaration at a higher level may deemed to be a false alarm at a lower level based on a more accurate physical system model. Ideally, a real-time expert system shell for smart structures should support hierarchical knowledge representation at various levels of abstraction so that both top-down diagnosis, bottoms up simulation or hybrid failure diagnosis strategies can be employed.

3.3 Damage Detection, Isolation and Assessment Strategies

There are two major elements in assessing the impact of a damage on the wing. The first is the determination of residual strength which would impact the structural maneuver limits after a damage. The second is the determination of aerodynamic effects which would impact the control reconfiguration strategy after a damage. Under residual strength effects, it is important to determine not only the current damage but also be able to reason about future likely effects such as delamination based on the current damage effects. Under aerodynamic effects, it is important to determine the control effectiveness of surfaces after a damage based on the detected structural damage. These assessment requirements necessitate the determination of the size of the hole caused by the damage and the stress redistribution in the wing after a damage.

For designing the damage detection and isolation strategy, an appropriate set of procedural rules can be initially selected from the wealth of algorithms developed for fault diagnosis in dynamic systems [74]-[77]. These algorithms have been developed for detecting sensor, actuator and component failures in dynamic systems. In the smart wing context, we will assume that the strain sensor hardware redundancy is such that any strain sensor failure can be attributed to structural damage. We now discuss the applicability of these algorithms to smart structures.

3.3.1 Hypothesis Test Formulations

Denoting the strain measurements by y_{ij} where i denotes the location across a rib and j denotes the location across a flange, the external load by f, the postulated damage circles by A_ℓ and considering only a static case for simplicity, the multiple hypothesis test formulation to the damage detection and isolation problem at hand would require the computation of measurement residual r_{ij}:

$$r_{ij} = y_{ij} - E[y_{ij} | \text{all } y_{mn} \text{ not in } A_\ell \text{ and } f] \qquad (3.1)$$

for all strain measurements y_{ij} for each damage hypothesis A_ℓ. Here, E denotes the expected value of y_{ij} given all measurements not affected by the postulated damage area A_ℓ. The computation of the estimate would involve using a global spatial filter based on the discretized partial differential equation describing the wing stress distribution. Considering that the number of strain measurements in this example is 2x6x7 = 84 and the number of postulated damage circles would substantially exceed 84 (considering cases when more than one sensor would be involved in a damage and the likelihood of different top and bottom damage locations), it is clear that the brute force approach of multiple hypothesis test to the problem at hand would result in a combinatorial explosion.

As a second approach, consider the application of the global surface damage detection and isolation algorithm used in the CRCA reconfiguration strategy. Such an approach would require the computation of

$$r_{ij} = y_{ij} - E[y_{ij} | \text{all } y_{ij}, \text{ f and no damage}] \qquad (3.2)$$

for each y_{ij} in the sensor array. That is, the measurement residual is the difference between the actual measurement and the estimate of the measurement using all measurements under the no damage hypothesis. Here, the tradeoff is between modelling uncertainty and damage signature. For a given y_{ij}, the more the estimate is based on that measurement y_{ij}, the less the effects of modelling errors would be on the residuals. On the other hand, the less the estimate is based on that measurement y_{ij}, the more the damage signature would be on the residuals. After a compromise filter design is selected, then the residuals would be processed to filter out the damage signature under each postulated damage scenario. Although this approach would have less computational burden that the multiple hypothesis testing formulation, it still would require a considerable amount of computational throughput. Although these formulations do not result in a practical solution, they do provide a formal framework to study the damage diagnosis problem. We discuss the most important one, namely, the damage signature issue next.

3.3.2 Damage Signature

Since a wing is a distributed parameter system (described by a partial differential equation), it exemplifies the two types of damage signature induced by a structural damage: spatial and temporal. The spatial signature is the characteristic of the stress distribution over the structure after a damage. For example, a strain peak associated with a hole in a wing is such an example. In the previous section, the relative magnitudes of r_{ij} in (3.2) would constitute such a diagnosis information.

The temporal signature is the stress characteristics of a damage over time. For instance, a natural damage would result in a slow crack whereas battle damage would result in a fast crack. In the context of the previous section, modifying (3.2) via

$$r_{ij}(k) = y_{ij}(k) - E\,[y_{ij}(k)\,|\,\text{all } y_{ij}(k-1),\ f(k-1) \text{ and no damage}] \quad (3.3)$$

would show a temporal strain peak over time associated with the onset of a damage (where k denotes the time index).

As discussed in the previous section, the brute force application of system theoretic damage detection algorithms do not result in a practical solution. What is required in an expert system solution capturing the problem solving skill of an expert with expertise in the structure and damage detection, isolation and assessment strategies. Such an approach is discussed next.

3.3.3 Expert System Based Damage Diagnosis

In the smart wing example, we have considered a number of strategies in detecting and isolating a damage from the observed temporal and spatial signatures of a damage. For the first strategy for each strain measurement, we compute

$$\hat{y}_{ij} = E[y_{ij}\,|\,f \text{ and no damage}] \quad (3.4)$$

that is, an estimate of the measurement based on the current external load value and the knowledge about the wing structure. This is the procedural rule performed at each sampling instant using a loop construct. In general, this estimate can be computed using either open-loop or closed-loop filtering based on the known properties of the wing (e.g., model description, finite element description, etc.)

The expert system rule base captures a domain expert's damage model. For instance, referring to figure 3.2, here is one such possible domain expertise formulation. The longitudinal stiffeners outboard of the hole can not carry axial loads. Strain sensors on damaged flanges measure approximately zero outboard of the hole. The sensors on undamaged booms located outboard the damage senses a uniformly redistributed load after a damage. The sensors inboard side of the damage would show no change due to a damage. Under these simplified set of rules, the following damage detection and isolation rules given in Table 3.3 would then apply:

Table 3.3: Qualitative Rules for Damage Detection and Isolation

---

If a strain sensor is not functioning, then the flange section containing the sensor has been damaged.

If a strain sensor measurement is significantly greater than its estimate, then the flange section containing that sensor is carrying additional load due to a damage on the same rib or further inboard.

If a strain sensor measurement is significantly less than its estimate, then the flange containing that sensor has been damaged further inboard.

The CLIPS implementation of the damage detection rule is given in Table 3.4. As the name implies the rule "detect-the-damage" incorporates the damage detection strategy used in the Smart Wing example. This rule determines which booms have been damaged.

The first statement in this rule,

```
(declare (salience -10))
```

is used to control when this rule can fire. All activated rules with a higher salience will fire first. When a rule does not have its salience declared than it is given a salience of 0 by default. This rule was given a salience of -10 to insure that all the estimates of the sensor measurements have been calculated before checking for damage.

The next statement in this rule,

```
?sensor <- (bad-sensors ?number-of-bad-sensors-on-the-wing),
```

binds the fact (bad-sensors ?number-of-bad-sensors-on-the-wing) to the variable ?sensor so that this fact can be retracted from the fact-list when this rule fires. This fact is used to keep track of the total number of sensors affected by the damage. Each time a sensor is determined to be affected by damage this fact is removed from the fact-list (retracted), the new total is calculated, and a new "bad-sensors" fact is asserted onto the fact-list. Retracting old facts from the fact-list makes the program run more efficiently. A fact will usually be retracted if it is going to be reasserted by the same rule.

The fact in the next pattern of the rule,

```
?boom <- (booms ?surface x ?x-location y ?y-location
                         meas ?measured-value  updated
                         est  ?estimated-value updated)
```

is bound to the variable ?boom since it will be retracted if this rule fires. This fact contains the measured and estimated values of strain at a given (x,y) location. The actual values are bound to the variables ?measured-value and ?estimated-value. These values are used in determining if damage has occurred. Another value used in determining damage is the value in the third pattern of this ruie,

```
(damage-detection-threshold ?threshold).
```

The last two statements on the left hand side of this rule are the actual conditions that must be satisfied to detect damage on a specific boom. The first condition,

```
(test (> (abs (- ?measured-value ?estimated-value)) ?threshold)),
```

is used to determine if the difference between the measured and estimated values for strain are greater than the detection threshold. Satisfaction of this condition alone indicates that damage has occurred, but the next condition,

```
  (test (< (abs ?measured-value) (abs ?estimated-value)))
```

is needed to determine which boom(s) has actually been damaged. A damaged boom can no longer carry its original load. Therefore, the measured strain of a damaged boom at a given location will be less than the estimated strain at the same location. The second condition is satisfied when this situation is true.

There are four actions on the right hand side of the rule "detect-the-damage". The first two action,

```
  (retract ?sensor)
  (retract ?boom)
```

retract the facts that were bound to the variables ?sensor and ?boom on the left hand side of the rule. These facts are retracted because they are being replaced in the final two action of this rule,

```
  (assert (booms ?surface x ?x-location y ?y-location
                              meas ?measured-value old
                              est ?estimated-value damaged))
  (assert (bad-sensors =(+ ?number-of-bad-sensors-on-the-wing 1)))).
```

A new "booms" fact is asserted onto the fact-list. It has changed the status of the measured value to 'old' to indicate that this measurement has been used in the damage detection rule. The status of the estimated value has been changed to reflect that this boom at the current location has been damaged. The value associated with the "bad-sensors" fact is incremented by one in order to update the total number of sensors affected by the damage.

Table 3.4: CLIPS Rule for Damage Detection

```
;This rule is used to detect damage.  It calculates the difference
;between the measured values of stress and the estimated values.
;When the difference is greater than a specified threshold damage
;has occurred. If the above condition is satisfied and the magnitude of
;the measured value is less than the magnitude of the estimated value
;then the boom under consideration has been damaged.
;
(defrule detect-the-damage
  (declare (salience -10))

  ?sensor <- (bad-sensors ?number-of-bad-sensors-on-the-wing)

  ?boom <- (booms ?surface x ?x-location y ?y-location
                          meas ?measured-value  updated
                          est  ?estimated-value updated)
  (damage-detection-threshold ?threshold)

  (test (> (abs (- ?measured-value ?estimated-value)) ?threshold))
  (test (< (abs ?measured-value) (abs ?estimated-value)))
=>
  (retract ?sensor)
  (retract ?boom)

  (assert (booms ?surface x ?x-location y ?y-location
                         meas ?measured-value old
                         est ?estimated-value damaged))
  (assert (bad-sensors =(+ ?number-of-bad-sensors-on-the-wing 1))))
```

The CLIPS implementation of the damage isolation rule is given in Table 3.5. The rule "detect-the-damage" determines which booms have been damaged, but it does not determine where on the booms the damage occurred. The rule "isolate-the-damage" is used to determine the y location of the damage on a particular boom. This is done by locating the furthest inboard sensor on a damaged boom that has been affected by the damage.

Since damage can not be isolated until it has been detected the rule "isolate-the-damage" was given a salience of -20. This salience is less than that of "detect-the-damage" so "isolate-the-damage" will not fire until all activations of "detect-the-damage" have fired.

The first pattern on the left hand side of "isolate-the-damage,

```
  (booms ?surface x ?x-location-of-sensor y ?y-location-of-sensor
                    meas ?measured-value old
                    est ?estimated-value damaged),
```

will be matched by any of the "booms" facts in the fact-list that had their status changed to 'damaged' by "detect-the-damage". The next pattern,

```
  ?great <- (greatest-y ?surface ?x-location-of-sensor
              ?y-location-of-the-furthest-inboard-sensor damaged|undamaged),
```

will be matched by any "greatest-y" facts that have the same ?x-location-of-sensor as the previous "booms" fact. The "greatest-y" fact keeps track of the y location of the furthest inboard sensor on a damaged boom that has been affected by the damage. The condition,

```
  (test (< ?y-location-of-the-furthest-inboard-sensor
           ?y-location-of-sensor)),
```

is used to determine if the current "greatest-y" fact needs to be updated. This condition is satisfied when the y location of an affected sensor on a damaged boom is greater than the current "greatest-y" y location for that boom.

When the condition is satisfied "isolate-the-damage" fires and these two action take place:

```
  (retract ?great)
  (assert (greatest-y ?surface ?x-location-of-sensor
                               ?y-location-of-sensor damaged)))
```

The first action removes the old "greatest-y" fact from the fact-list and the second action places a new "greatest-y" fact on the fact-list. The new "greatest-y" now has the updated y location.

Table 3.5: CLIPS Rule for Damage Isolation

```
;This rule isolates the damage on each failed boom to the most inboard
;y location where damage has been detected.
;
(defrule isolate-the-damage
  (declare (salience -20))

  (booms ?surface x ?x-location-of-sensor y ?y-location-of-sensor
                  meas ?measured-value old
                  est ?estimated-value damaged)

  ?great <- (greatest-y ?surface ?x-location-of-sensor
             ?y-location-of-the-furthest-inboard-sensor damaged|undamaged)

  (test (< ?y-location-of-the-furthest-inboard-sensor
           ?y-location-of-sensor))
=>
  (retract ?great)
  (assert (greatest-y ?surface ?x-location-of-sensor
                                  ?y-location-of-sensor damaged)))
```

The initial attempt at the estimate damage diagnosis strategy has certain shortcomings. First, the procedural computation for every strain sensor measurement is a time-consuming task. Second, the strategy depends on knowing the external load -- a hard computational problem especially in a maneuvering dynamic environment. Both of these deficiencies can be alleviated by further investigating the damage diagnosis skill of an expert and incorporating into the design. Here, we outline some possible solutions.

One such possible solution is to introduce temporal local filtering in computing the strain sensor measurement estimates. A simple solution, which might suffice under slowly varying (compared to the underlying measurement sampling rate) external force conditions is replacing (3.4) by:

$$\hat{y}_{ij}(k) = E[y_{ij}(k)\,|\,y_{ij}(k-1)] \cong y_{ij}(k-1) \qquad (3.5)$$

The assumption above states that the estimate of a strain measurement is equal to the previous measurement under no damage conditions. Therefore, any discrepancy between the current actual and estimated measurements would indicate the presence of a structural damage which would be isolated using the same set of rules in Table 3.1. Using the strain estimate provided by (3.5) would eliminate both the necessity of measuring external load and computing the strain estimate globally. Other approaches based on closed-loop temporal filtering based on the local structural constraints or adaptive techniques based on the local temporal stress gradient can also be employed instead of the approximation in (3.5).

We have replaced the strain estimate with its previous measurement so that the temporal damage signature can be exploited to detect and isolate on damage. There is also a spatial signature of a structural damage. It is also possible to develop rules exploiting the spatial signature of a damage in detecting and isolating a damage. Table 3.6 gives such a possible set of rules.

Table 3.6: Damage Diagnosis Rules Based on Spatial Signature

If a strain sensor is not functioning, then the flange section containing the sensor has been damaged.

If there is a malfunctioning strain sensor, then the damage is bounded by the two functioning sensors on each of the malfunctioning sensor located at the same rib cross section.

If a strain sensor is significantly less than its neighboring sensors on the same rib cross section, then the flange containing that sensor has been damaged further inboard.

The qualitative rules in Table 3.6 can be quantified by introducing the following estimate for a strain measurement:

$$\hat{y}_{ij} = E[y_{ij} | y_{i,j-1}, y_{i,j+1} \text{ and no damage}] \tag{3.6}$$

Eq. (3.6) requires a local spatial filter around every strain measurement by using the two adjacent strain measurements on each side. Approximations based on the local spatial gradient can also be used. Such a neighborhood definition would result in a maximum damage signature for damage holes covering at most a single sensor. Other definitions of neighborhood are also possible. For example, for a given strain measurement, y_{ij}, using the sensors on the four corners of a square defined by $\{y_{i-1,\ j}, y_{i,\ j+1}, y_{i+1,\ j}, y_{i,\ j-1}\}$ around the strain measurement y_{ij} would be another suitable local neighborhood definition.

Finally, combined local temporal and spatial filtering is also another possible approach for generating damage detection information. Here again, approximations based on the empirical computation of the local temporal and spatial gradients may suffice for damage detection purposes. For instance, the local filtering would be defined over a three-dimensional cube including the time dimension in addition to the two dimensions defining sensor array position on the structure. As before, the maximum damage signature would be generated by excluding all sensors affected by the damage in the cube. For example, considering damage holes covering a single strain sensor, the strain prediction given by

$$\hat{y}_{ij}(k) = E[y_{ij}(k) | y_{i,j-1}(k-1), y_{i,j}(k-1), y_{i,j+1}(k-1), y_{i,j-1}(k), y_{i,j+1}(k)] \tag{3.7}$$

would yield such a signature.

3.3.4 Applicability of Artificial Neural Networks

As discussed in the previous chapter, neural network architectures are appropriate for applications requiring massive parallel distributed processing. If the sensor arrays used in a smart structure are based on a continuous sensing mechanism, then any discrete spatial sampling of a sensor array would induce a large data array suitable for such a parallel processing. Mapping each strain sensor onto a processing element in a neural network and defining the interconnects between the processing elements based on the local neighborhoods defined in the last section, such a neural network can be trained with undamaged wing data to find the optimal weighting coefficients under no damage conditions.

For example, considering the neighborhood defined by the strain measurements $\{y_{i-1,j}, y_{i,j+1}, y_{i+1,j}, y_{i,j-1}\}$ around strain measurement y_{ij} for the static case using the most elementary model of a neuron (i.e., linear weighted average), we would have the following representation for $\hat{y}_{ij}$

$$\begin{aligned}\hat{y}_{ij} &= w_{ij}(i-1,j)y_{i-1,j} + w_{ij}(i,j+1)\, y_{i,j+1} \\ &\quad + w_{ij}(i+1,j)y_{i+1,j} + w_{ij}(i,j-1)\, y_{i,j-1} \qquad (3.8)\end{aligned}$$

The training of the neural network with a set of simulated wing (undamaged) stress data would then yield the set of weighting coefficients $w_{ij}(\cdot,\cdot)$ for each strain sensor measurement.

Hence, neural network based prediction of strain measurements is a nonalgorithmic counterpart of the local prediction filters discussed in the previous section. The similarities between neural networks for analyzing spatiotemporal input patterns and algorithmic signal processing structures (Finite Impulse Response filters, etc.) are noted in [73]. In contrast to the nonalgorithmic approach inherent in the neural network formulations, the model based expert system would incorporate the expert's knowledge (local smoothness constraints on strain (e.g., discretized form of a known partial differential equation representation), material properties, (e.g., density, joining fabrication, etc.) in solving for the weighting coefficients.

Once the strain prediction model is found by either a knowledge based expert system or a neural network approach, then there relations can be used to detect a structural damage. As for isolating and assessing damage, the knowledge based expert system seems to be currently a more feasible solution since current control law designs cannot make use of the neural net representation of a structure directly. Therefore, the damage information has to be translated into area loss, decrease in residual strength type information. Although it seems to be at least conceptually possible to train the neural net with a multitude of damaged wing data in order to create a damage signature dictionary, it is not clear to see the efficiency of such an implementation over a knowledge based expert system approach.

Summarizing, artificial neural networks can play a complementary role in building smart structures based on a knowledge based expert system approach. The research questions that would be crucial on applying neural nets to smart structures are the following: How does the selection of training data (e.g., various external loads, dynamically varying external load, etc.) affect the resulting weighting coefficients? How can the adaptive learning mechanism differentiate between normal dynamic behavior of an undamaged structure and behavior associated with a damaged structure? How can the desire of maximum damage signature attribute be incorporated into the selection of the interconnection structure?

3.4 <u>Real-Time Performance in an Embedded Environment</u>

An smart structures such as the example described, has to exhibit strict real-time performance. For instance, in an unstable aircraft such as the X-29, such a system has to produce the correct answer in at most two sampling instants whenever the inflicted damage could cause a flight control instability. Therefore, just being predictably fast enough most of the time or just providing an answer within a time limit are not satisfactory criteria for real-time performance. Hence, the worst case execution time performance of an expert system has to be determinable before embedding into a time-critical application. Therefore, a real-time expert system shell should support user defined search strategies so that the fault diagnosis strategy of the domain expert can be incorporated into the expert system design.

The CLIPS implementation of the smart wing expert system contains approximately 50 rules. The executable image of the rule-based expert system is approximately 350 Kb. Since the standalone run-time CLIPS is 280 Kb, the rule base introduces an additional 70 Kb. Since increasing software size imposes additional weight requirements on an aircraft (more memory, wires, power, etc.), the comparison underscores the importance of generating a tight expert system code for embedded applications.

In terms of execution speed, a standalone (i.e., noninteractive) version the CLIPS implementation was not generated to evaluate execution time performance. However, based on our analysis of similar applications [67], an interpretive shell approach is not currently feasible for real-time operation with a large rule base. Our study in [67] outlines a rule set compiler technique which allows the interpretation to be performed off-line, thus allowing a faster real-time performance. The rule set compiler alleviates the overhead associated with pattern matching, and fact assertion and retraction found in conventional expert system shells.

Although not implemented, there are at least two logical hybrid implementations of the smart wing example. The first one would be using a procedural code for reading the measurements and computing the various estimation parameters (sequential algorithmic tasks performed at every sampling instant) as user defined functions in CLIPS. This would reduce the number of rules by about 50%, increase execution speed over the standalone CLIPS version with an accompanying slight decrease in program size. The other alternative would be the replacement of rules in a procedural language version with a CLIPS call for performing the damage detection, isolation and assessment functions. We suspect that the program size and execution speed of this hybrid implementation would be comparable to the first one.

3.5 Requirements on the Expert System Development Tool

Here, we outline the desirable attributes of an ideal expert system shell for developing smart structure applications.

3.5.1 External Environment Interface

Real-time fault monitoring systems for smart structures have to read in data at a fixed rate from a set of sensors (e.g., piezoelectrics, fiber optics sensors, etc.). Hence, a real-time expert system for onboard applications should support efficient input and display data functions. Since conventional expert systems development presuppose an interactive environment, an efficient repetitive data read-in and assignment facility is not available in most expert system shells.

3.5.2 Temporal Reasoning

In real-time systems, an expert system has to reason about past, present, and future events. Moreover, the temporal sequence of events has to be accounted for as well. In the theory of temporal reasoning, a number of formulations have been developed [61]. The two most important ones are based on, first, assertions about time intervals and, second, assertions about time points. For instance, in an interval based formalism, one deals between interval relations such as before, after, overlaps, starts, finishes, etc. Such a temporal logic propagates constraints about intervals by transitivity.

Most expert system shells do not support such temporal reasoning. Only in hybrid expert systems supporting dynamic objects, objects and their links to classes can be modified at runtime. In the smart wing example, the temporal reasoning is implicitly contained in the sampling of the measurements. In general, every physical model of a physical dynamic system would dictate a different time interval for which the input and output measurements have to be saved. Ideally, an expert system shell should support the specification of the memory attribute of a dynamic object (the time interval over which the reasoning about a fault has to be performed).

3.5.3 Integration into Conventional Software

Since most current embedded applications dictate either Ada or C, an expert system shell written in one of these languages would allow an easy integration into conventional software. CLIPS is an example of such a shell written in C. It is completely embeddable in other applications written in FORTRAN, Ada, and C by building an appropriate interface package. In a real-time onboard expert system, such an interface should be accomplished without incurring any significant computational overhead.

3.5.4 Symbolic and Numeric Reasoning

In the smart wing example, the reasoning about the interconnections between the structural elements (ribs, flanges, hydraulic lines, etc.) need a topological knowledge representation which requires symbolic reasoning. Other higher level information such as hydraulic system test results, maintenance history about a specific unit can be easily incorporated into such a representation, thus allowing additional reasoning power for asserting malfunctions. In contrast, the expressions on the left hand side of if-then-else rule in the smart wing involve mathematical computations (e.g., computation of the strain measurement residuals). This example is a fairly simple application; in most systems, more elaborate mathematical computations (involving, for instance, operations with matrices vectors, etc.) would be needed. Hence, an ideal expert system shell for onboard real-time smart structure applications should support extensive domain algebra in rule expressions.

3.6 Smart Structure Design Methodology

The first step in smart structure development cycle is to study the structure to be described. It is not important at this stage of the cycle to fully specify the entire structure, even if such knowledge is available as would be the case in an already existing system. What is important is to identify the structural objects (elements) of interest and to organize a complex structure description in terms of a nested hierarchy where much of the lower level details are temporarily hidden by a high level description. An appropriate high level description may, for instance, consist of less than a dozen objects along with their interconnections. Those objects at one level in the hierarchical arrangement can later be represented as entire subsystems at the immediately lower level; these subsystem descriptions may be supplied at a later time. The goal here is to use the power of nested representation to hide low level details so as to avoid having such details overwhelm the entire modeling effort.

The next step in modeling a structure is to examine any available component libraries for reusable subsystem descriptions. For example, one library may contain composed of flanges, webs, ribs, while another may contain various types fiber optic based sensors, while another can contain various joiner models.

The third step of the system modeling task is to write a first attempt at a user description of the structure using the selected expert system shells. This first try should use only the highest layer of the modeled system along with any usable library subsystems. After the user description of the structure is written, it can be run through the expert system shell compiler to detect and report various errors even though not enough information may be present for damage detection or assessment activities.

Once this high level topographical description is proved syntactically correct, the fourth step of incorporating procedural information in the expert system description of the structure. At this stage in the model cycle, it would be prudent to first write the procedural information required for system state presetting and simulation and insure its proper functioning before implementing diagnostic knowledge.

The fifth step is to try interpreting the system model simulation using the expert system shell, and continuing refinements in the model based upon observations of its behavior. As more confidence in the correctness of the model is gained, the fidelity of the model can be improved with the expansion/substitution of various components throughout the model with lower level subsystem representations. Ultimately, every component in the model's topological knowledge is either atomic (undivisible) or represented by a subsystem; additionally, the entire system has simulation code present and tested.

Once a system model is established with a complete topography and complete simulation procedural knowledge, the sixth stage of model development is to provide the system model with diagnostic procedural knowledge. For each system and component in the model, diagnostic ruleset code is written to perform tests upon functions for that part of the model. The goal of achieving high speed diagnostic capability can be met by designing damage signature checks for the higher level components; when these checks are passed, it alleviates time consuming examination of checks at lower levels.

However, as is the case with simulation ruleset code, diagnostic code for a given system only has to be written once and can then be duplicated and reused for other system models. Also, both diagnostic and simulation procedural information for subsystems can be written by specialists and then later used by generalists without requiring the generalists to be fully familiar with the lower level details.

For certain time critical applications, circumstances may occur so that it may not be possible to run all of the desired diagnostic code in the limited time available. For these applications, a family of consistency checks may be written such that the quicker running (more general) checks are used first and slower running (more specific) checks are used should time remain available. This graded strategy helps ensure that at least some diagnostic results are generated even if the interval allowed for diagnosis is insufficient for the circumstances for a particular cycle. The reasoning here is that it is better to derive a general, partially useful result instead of no result at all.

The exact details of a diagnostic ruleset will vary among differing structures. However, for system models with multiple levels, a top-down selective approach would be appropriate for fixed time interval diagnosis. This approach, unlike the bottom-up full evaluation approach used for simulation, will spend time working on only those subsystems where problems are suspected. In order to test the diagnostic code, the developer can purposely introduce faults in the system simulation. Such introduction can be performed by various techniques: reading in a faulty system state via a preset operation, writing deliberate faults in the system topography or simulation information, or by providing for the interactive prompting for critical information during simulation.

Now armed with a well-tested system model, the developer should now review the model for any potentially reusable components, and to take such components and add them to the system library for future smart structure application.

The final step in the development cycle is to use the expert system shell to generate a standalone compiled version of the expert system in a conventional programming language in a manner suitable for porting the model to an embedded computer environment. Under this stage of the development, the standalone interpreted/compiled version of the expert system would be tested, first, using a nonreal-time simulation generated sensor data, second, using, laboratory recorded sensor data in a nonreal-time simulation environment, third, using a real-time laboratory simulation environment, and fourth, using the embedded program in a flight test.

4. CONCLUSIONS AND RECOMMENDATIONS

4.1 Conclusions

We have investigated and defined a baseline architecture for a smart aerospace structure consisting of:

- sensor arrays for immediate damage detection such as in-situ piezolectrics and fiber optic sensors used in conjunction with other rigid and flexible body conventional sensors
- a knowledge based expert system for damage detection, isolation and assessment, capturing the domain expert's expertise in diagnosing structural damage based on temporal and spatial damage signatures
- an identification system which transforms the structural damage information into a model suitable for the redesign of the active controller on-line
- a decentralized controller which reconfigures itself on-line based on detected damage conditions and their estimated levels
- an expert system implementation which operates real-time based on a knowledge compiler approach using general purpose numeric processors.

Our baseline architecture assumes availability of local sensors for structural damage detection such as embedded fiber optics. Furthermore, global sensors such as rigid body accelerometers can also be incorporated into the hierarchical description damage detection system as well. Starting from current actuator failure and flight control surface failure detection algorithms, we have formulated the structure of a hierarchical damage detection algorithm and outlined its shortcomings in smart structure applications. We have then investigated various expert systems based damage detection and isolation strategies which alleviate these shortcomings by analyzing the local temporal and spatial signature of a damage. We have then analyzed the implementation details using a CLIPS implementation of a smart wing example. In particular, we have identified the knowledge representation issues in building smart structures and resultant requirements on an expert system development tool.

In summary, our investigation has shown that a knowledge based expert system approach seems to be a viable method for implementing the "brain" of a smart structure. Such an approach requires an expert system shell allowing the specifications of both the physical connectivity and proximity of structural elements and the functional dynamic behavior of these structural components. Our analysis indicates that the local spatial and temporal signatures of a damage can be exploited in detecting and isolating a damage without running into a combinatorial explosion associated with conventional failure detection techniques. Moreover, we have shown how artificial neural networks can also play a complementary role in implementing such knowledge based damage detection and isolation strategies. Finally, the real-time performance of such a system in an embedded environment requires an expert system development tool such that the interpretive portion of the expert system can be performed off-line to minimize the size of the on-line software, and user defined search strategies can be implemented to maximize execution efficiency in an embedded environment.

4.2 Recommendations

Based on our Phase I analysis, we recommend the following:

- generalize and implement the smart wing concept for a real aircraft wing
- analyze performance using a realistic nonreal-time simulation
- analyze performance using actual undamaged and damaged wing data
- perform laboratory demonstration using a selected sensor array technology

- perform the necessary R&D to accomplish the four preceding tasks.

Under the Phase I analysis, we have identified potentially suitable knowledge representation, and damage detection and isolation strategies for implementing the brain of a smart structure. In this recommended follow-on, we would determine the best manner of implementing such a structure. In addition, a real application would uncover additional knowledge representation requirements and strategies exploiting additional information in determining damage. The analysis with a realistic wing simulation would allow the study of the impact of modelling errors on the developed strategy. It is crucial that the smart structure should be able to operate with an imperfect knowledge base. Finally, using the available wind tunnel data for a damaged wing would represent the ultimate test of the impact of such modelling errors on both representation and reasoning about faults. Finally, a laboratory demonstration or the use of laboratory recorded data would identify the necessary sensor interface and modelling issues to be resolved.

5. REFERENCES

[1] Chandler, P.R., "Self-Repairing Flight Control System Reliability and Maintainability Program Plan," AFWAL/FDL, February 1984.

[2] Caglayan, A.K., Rahnamai, K., Moerder, D.D., and Halyo, N., "A Hierarchical Reconfiguration Strategy for Aircraft Subjected to Actuator Failure/Surface Damage," Charles River Analytics Inc., AFWAL-TR-87-3024, May 1987.

[3] "Reconfiguration Strategies for Aircraft Flight Control Systems Subjected to Actuator Failure/Surface Damage," Lear Siegler Inc., AFWAL-TR-86-3110, March 1987.

[4] Davison, J., "Expert Systems in Maintenance Diagnostics for Self-Repairing Flight Control Systems," Proceedings of the Joint Services Workshop in Artificial Intelligence for Maintenance Diagnostics, AFHRL-TR-84-25, 1984.

[5] Gross, H.N., Chandler, P.R., and Eslinger, "Renewed Interest in Hinge Moment Models for Failure Detection and Isolation," NAECON 86, Dayton, OH, May 1986.

[6] Bailey, T., and Hubbard, J.E., "Distributed Piezoelectric-Polymer Active Vibration Control of a Cantilever Beam," Journal of Guidance, Control, and Dynamics, Vol. 8, No. 5, 1985, 605-611.

[7] Burke, S.E., and Hubbard, J.E., "Active Vibration Control of a Simply Supported Beam Using a Spatially Distributed Actuator," IEEE Control Systems Magazine, August 1987.

[8] Sirlin, S.W., Laskin, R.A., "The Softmounted Inertially Reacting Pointing System," AAS Guidance and Control Conference, February 1986.

[9] Crawley, E.F., and de Luis, J., "Use of Piezo-Ceramics as Distributed Actuators in Large Space Structures," 26th Structures, Structural Dynamics, and Materials Conference, 1985.

[10] Crawley, E.F., and de Luis, J., "Experimental Verification of Distributed Piezoelectric Actuators for use in Precision Space Structures," 27th Structures, Structural Dynamics, and Materials Conference, 1986.

[11] Hanagud, S., Obal, M.W., and Meyyappa, "Electronic Damping Techniques and Active Vibration Control," 26th Structures, Structural Dynamics, and Materials Conference, 1985.

[12] Rogowski, R.S., Heyman, J.S. and Claus, R.D., "The Evolution of Smart Composite Materials," NASA Tech Briefs, Vol. 12, No. 9, October 1988.

[13] Witsenhausen, H.S., "A Counterexample in Stochastic Optimal Control," SIAM J. of Control, Vol. 6, pp. 131-147, 1968.

[14] Bismut, J.M., "An Example of Interaction Between Information and Control," IEEE Trans. on Automatic Control, Vol. AC-18, pp. 63-64, 1973.

[15] Whittle, P., and Rudge, J.F., "The Optimal Linear Solution of a Symmetric Team Control Problem," J. of Applied Probability, Vol. 11, pp. 377-381, 1974.

[16] Kosut, R.L., "Suboptimal Control of Linear Time-Invariant Systems Subject to Control Structure Constraints," IEEE Trans. on Auto. Control, Vol. AC-15, October 1970, pp. 557-563.

[17] Wenk, C.J., and Knapp, C.H., "Parameter Optimization in Linear Systems with Arbitrary Constrained Controller Structure," IEEE Trans. on Auto. Control, Vol. AC-25, June 1980.

[18] Schmidt, D.K., "Optimal Flight Control Synthesis via Pilot Modelling," J. of Guidance and Control, Vol. 2, August 1979.

[19] Fennel, R.E., and Black, S.B., "Integrated Airframe Propulsion Control," NASA CR-3606, August 1982.

[20] Caglayan, A.K., Halyo, N., and Broussard, J.R., "The Use of the Optimal Output Feedback Algorithm in Integrated Control System Design," National Aerospace and Electronics Conference, Dayton, OH, May 1983.

[21] Speyer, J.L., "Computation and Transmission Requirements for a Decentralized Linear Quadratic-Gaussian Control Problem," IEEE Trans. on Auto. Control, Vol. AC-24, No. 2, April 1979.

[22] Sundareshan, M.K., "Generation of Multilevel Control and Estimation Schemes for Large-Scale Systems: A Perturbational Approach," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-7, No. 3, March 1977.

[23] Balas, M.J., "Active Control of Flexible Systems," Journal of Optimization Theory and Applications, Vol. 25, No. 3, 1978, 415-436.

[24] Schaechter, D., "Optimal Local Control of Flexible Structures," Journal of Guidance and Control, Vol. 4, No. 1, pp. 22-26, January 1981.

[25] West-Vukovich, G. Davison, E.J., and Hughes, P.C., "The Decentralized Control of Large Flexible Space Structures," In Proceedings of the 20th Conference on Decision and Control, pp. 949-955, Albuquerque, NM, December 1981.

[26] Kida, T., Yamaguchi, I., and Ohkami, Y., "An Approach to Local Decentralized and Colocated Control Units," In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Snowmass, CO, August 1985.

[27] Finzi, E., Lanz, M., and Mautegazza, P., "Active Structural Control with Decentralized and Colocated Control Units," In Proceedings of the AIAA Guidance, Navigation, and Control Conference, Snowmass, CO, August 1985.

[28] Sundararajan, N., Montgomery, R.C., and Williams, J.P., "Adaptive Identification and Control of Structural Dynamic Systems using Recursive Lattice Filters," NASA TR2371, January 1985.

[29] Banda, S.S., Ridgely, D.B., and Yeh, H-H., "Robustness of Reduced Order Control," Proceedings of the Fourth VPI & SU/AIAA Symposium on Dynamics and Control of Large Flexible Structures, 1983.

[30] Fanson, J.L., Caughey, T.K., "Positive Position Feedback Control for Large Space Structures," AIAA 87-0902.

[31] Cunningham, T., Carlson, D., Hendrick, R., Shanar, D., Hartmann, G. and Stein, D., "Fault Tolerant Digital Flight Control with Analytical Redundancy," AFFDL-TR-77-25, AFFDL, WPAFB, Ohio, May 1977.

[32] Deckert, J.C., Desai, M.N., Deyst, J.J. and Willsky, A.S., "Reliable Dual-Redundant Sensor Failure Detection and Identification for the NASA F-8 DFBW Aircraft," NASA CR-2944, February 1978.

[33] Beattie, E.C., LaPrad, R.F., McGlone, M.E., Rock, S.M., and Akhter, M.M., "Sensor Failure Detection System Final Report," NASA CR-165515, PWA-5756-17, NASA-Lewis Research Center Contract NAS3-22481, August 1981.

[34] Morrell, F.R. and Russell, J.G., "Design of a Developmental Dual Fail Operational Redundant Strapped Down Inertial Measurement Unit," NAECON 1980.

[35] Caglayan, A.K., and Godiwala, P.M., "A Preliminary Design for Flight Testing the FINDS Algorithm," NASA CR-178043, March 1986.

[36] Caglayan, A.K., Rahnamai, K., and Moerder, D.D., "Reconfiguration in Stages Part I: Detection and Identification of Control Impact in a Self-Repairing Flight Control System," NAECON 86, Dayton, OH, May 1986.

[37] Potter, J.E. and Suman, M.C., "Thresholdless Redundancy Management with Arrays of Skewed Instruments," AGARDOGRAPH 224, Integrity in Electronic Flight Control Systems, 1977.

[38] Beard, R.D., "Failure Accommodation in Linear Systems Through Self-Reorganization," NASA CR-118314, 1971.

[39] Chien, T.T., "An Adaptive Technique for a Redundant Sensor Navigation System," CSDL Report T-560, February 1972.

[40] Montgomery, R.C., and Caglayan, A.K., "Failure Accommodation in Digital Flight Control Systems by Bayesian Theory," Journal of Aircraft, Vol. 13, No. 2, February 1976, pp. 69-75.

[41] Willsky, A.S.and Jones, H.J., "A Generalized Likelihood Ratio Approach to the Detection and Estimation of Jumps in Linear Systems," IEEE Trans. on Auto. Control, Vol. AC-21, pp. 108-112, February 1976.

[42] Majors, C.S., "A Demonstration of the Use of Generalized Parity Relations for Detection and Identification of Instrument Failures on a Free-Beam," Masters Thesis , MIT Department of Aeronautics and Astronautics, September 1981.

[43] Baruh, H. and Choe, K., "Sensor Failure Detection for Flexible Structures," AIAA J. of Guidance and Control, Vol. 10, No. 5, October 1987.

[44] Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F. and Sacerdoti, E., "The Organization of Expert Systems: A Prescriptive Tutorial," Xerox Report No. VLSI-82-1, Palo Alto, CA, Jan. 1982.

[45] Brown, J.S., Burton, R.R., and Bell, A.G., "SOPHIE: A Sophisticated Instructional Environment for Teaching Electronic Troubleshooting," Bolt Beranek and Newman, Inc., Report No. 2790, Cambridge, MA 1974.

[46] Shortliffe, E.H., Computer-based Medical Consultations: MYCIN, American Elsevier, New York, NY, 1976.

[47] Duda, R.O., Gaschnig, J., Hart, P.E., Konolige, K. Reboh, R., Barrett, P., and Slocum, J., "Development of the PROSPECTOR Consultation System for Mineral Exploration Inc., Final Report, Project No. 6415, SRI International Inc., Menlo Park, CA, 1978

[48] Buchanan, B.G., and Feigenbaum, E.A., "DENDRAL and Meta-DENDRAL: Their Applications Dimension," Artificial Intelligence, Vol. II, 1978.

[49] Regenie, V.A. and Duke, E.L., "Design of an Expert-System Flight Status Monitor," NASA TM 86739, August 1985.

[50] Abbott, K.H., Schutte, P.C., Palmer, M.T., and Ricks, W.R., "Faultfinder: A Diagnostic Expert System with Graceful Degradation for Onboard Aircraft Application," Proceedings for Symposium on Aircraft Integrated Monitoring Systems, Friedrichshafen, West Germany, September 15-17, 1987.

[51] Davis, R., Shrobe, H., Hamscher, N., Wreckert, K., Shirley, M., and Polit, S., "Diagnosis Based on Descriptions of Structure and Function," AAAI Proceedings, Carnegie-Mellon Univ., Pittsburg, PA, August 1982.

[52] Davis, R., "Reasoning from First Principles in Electronic Troubleshooting," Int. J. Man-Machine Studies, Vol. 19, pp. 403-423, 1983.

[53] Davis, R., "Diagnostic Reasoning from Structure and Behavior," Artificial Intelligence, Vol. 24, pp. 347-410, 1984.

[54] Davis, R., "Knowledge-Based Systems: The View in 1986," in AI in the 1980s and Beyond - An MIT Survey, (editors Grimson and Patil, The MIT Press, Cambridge, MA, 1987.

[55] Forbus, K.D., "Qualitative Process Theory," A.I. Memo No. 694, M.I.T. Artificial Intelligence Laboratory, Cambridge, MA, February 1982.

[56] Forbus, K.D., "Qualitative Process Theory," Ph.D. Thesis, M.I.T., Dept. of Electrical Engineering and Computer Science, 1984.

[57] Forbus, K., "Interpreting Observations of Physical Systems," IEEE Trans. on SMC, Vol. 17, No. 3, May 1987.

[58] Forbus, K., "Qualitative Physics: Past, Present and Future," in Exploring Artificial Intelligence, Morgan Kaufman Publishers, 1988.

[59] Govindaraj, T., "Qualitative Approximation Methodology for Modelling and Simulation of Large Dynamic Systems: Applications to a Marine Steam Power Plant," IEEE Trans. on SMC, Vol. 17, No. 6, 1987.

[60] Rasmussen, J., "The Role of Hierarchical Knowledge Representation in Decision Making and System Management," IEEE Trans on SMC, Vol. 15, No. 2, 1985.

[61] Shoham, Y., Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence, The MIT Press, Cambridge, MA, 1988.

[62] Dallery, F., "Failure Detection and Identification for a Reconfigurable Flight Control System," NASA CP 2451, December 1983.

[63] Handelman, D.A., "An Application of Artificial Intelligence Theory to Reconfigurable Flight Control System," NASA CP 2451, December 1983.

[64] Laffey, T.J., et. al., "Real-Time Knowledge Based Systems," AI Magazine, Vol. 9, No. 1, Spring, 1988.

[65] Gupta, A., "Parallelism in Production Systems: The Sources and the Expected Speed Up," Expert Systems and their Applications, Fifth International Workshop, Avignon, France, 1985.

[66] Forgy, C.L., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," Artificial Intelligence, Vol. 19, 1982.

[67] Edwards, S.J. and Caglayan, A.K., "Expert System for Real-Time Monitoring and Fault Diagnosis," Charles River Analytics Inc., Report No. R8901, Cambridge, MA, November 1988.

[68] Anderson, J.A. and Rosenfeld, E., Neurocomputing - Foundations of Research, The MIT Press, Cambridge, MA, 1988.

[69] Kohonen, T., Self-Organization and Associative Memory, Berlin:Springer, Verlag, 1984.

[70] Cohen, M.A. and Grossberg, S., "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," IEEE Transactions on SMC, Vol. 13, No. 5, October 1983.

[71] Grossberg, S., Studies of Mind and Brain, Boston:Reidel, 1982.

[72] Hecht-Nielsen, R., "Artificial Neural Systems Technology," TRW Rancho Carmel AI Center, San Diego, CA, June 1986.

[73] Myers, M.H., "Some Speculations on Artificial Neural System Technology," IEEE Paper, No. 0547-3578, 1986.

[74] Pau, L.F., Failure Diagnosis and Performance Monitoring, Marcel Dekker, Inc., New York, 1981.

[75] Mozgalevskii, A.V., "Technical Diagnostics: Continuous Systems," Automatika i Telemekhanika, Vol. 39, No. 1, January 1978.

[76] Willsky, A.S., "Failure Detection in Dynamic Systems," AGARD-LS-109, Fault Tolerance Design and Redundancy Management Techniques, September 1980.

[77] Basserville, M., "Changes in Statistical Models: Various Approaches in Automatic Control and Statistics," Rapports de Recherche I.R.I.S.A., No. 72, May 1981.

U.S. GOVERNMENT PRINTING OFFICE: 1989-648-0