Fault Detection and Identification in an Intelligent Restructurable Controller

K. M. PASSINO and P. J. ANTSAKLIS
Department of Electrical and Computer Engineering, University of Notre Dame, Notre Dame, IN 46556, U.S.A.

Abstract. The fault detection and identification problem in an intelligent restructurable controller is addressed using a combination of algorithmic and artificial intelligence methods. An architecture is developed to address this problem. The integration of a variety of distinct knowledge representations and diagnostic reasoning techniques, and the system design and implementation is facilitated by the introduction of a novel knowledge representation graph.

1. Introduction

Intelligent restructurable control [1, 2] is a paradigm of future advanced adaptive control of aircraft for the case when significant changes in the system's underlying structure due to environment or component failures must be compensated for using control reconfiguration if necessary. The actions of the intelligent controller, which will depend on the detected changes, the avionic systems and the pilot, are made in an intelligent manner involving on-line decision making processes. Central to the restructurable controls problem is the issue of fault detection and identification (FDI) since before any control law reconfiguration is possible, the failure must be reliably detected, identified, and conveyed to the pilot and avionic systems.

Regardless of whether the aircraft is equipped with special control reconfiguration capability, reliable FDI information is extremely important to the pilot. A case in point is the Delta flight 1080 from San Diego to Los Angeles where the elevator became jammed at 19 degrees up and the pilot was given no indication of this type of failure [1]. Quick presentation of fault information to the pilot can often enable him to accommodate for the failure. This pilot cannot, however, recover from some types of failures so it becomes necessary to aid the pilot with special reconfiguration strategies.

Here we focus on the FDI problem but keep in mind that it must be integrated into a system whose goal will be to accommodate for the failure. Consequently, the FDI system must generate information useful for the control law reconfiguration, and the

interface issue with the pilot and avionics systems must be addressed. The approach taken here is a hierarchical one. At a low level, algorithmic techniques are used, for example, to detect failures; while at a higher level, artificial intelligence (AI) techniques are used to interpret data generated by various algorithms and to interface with the pilot. Although the FDI system and methodology developed here may be appropriate for other plants and component failures, our main interest in this paper is FDI problems for aircraft actuators. The architecture used to integrate the AI and algorithmic techniques, together with the emphasis on the pilot and avionics interface constitute a new approach to the problem.

In the next section, the FDI problem is defined and the important issues and tradeoffs are identified and discussed. This is followed by a system description and some guidelines on types of approaches to be taken. AI approaches to FDI are overviewed and an expert system architecture for FDI is proposed. Cognitive modeling studies of pilots in a failure diagnosis task set the foundation for the architecture and indicate that wide varieties of knowledge and reasoning, both deep and shallow, should be used. For instance we will need shallow knowledge such as general diagnostic heuristics, but we will also need to reason from deep knowledge such as a causal model of the aircraft. To deal with the wide variety of knowledge representations we propose that a graphical approach called the knowledge representation graph (KR graph) be used to represent both the deep and shallow knowledge. From this unified representation, the interface between deep and shallow reasoning becomes transparent and the expert system is relatively easy to code. The AI portion of the FDI system, named the 'Fault Detection and Identification Expert System' (FDIES), was implemented using the OPS5 expert system development tool and was exercised to perform diagnosis on several types of actuator failures.

2. Fault Detection and Identification on Aircraft

2.1. ISSUES AND TRADE OFFS

As is often done, we assume that all the faults occur abruptly in the aircraft. Then the FDI problem can be broken into three distinct tasks: [3] (i) Detection: making a binary decision, either that there is a fault or that there is not; (ii) Isolation: problem of determining the source of the failure; (iii) Estimation: determining the extent of the failure. Here we group *isolation* and *estimation* into *identification* and obtain FDI. Computational complexity increases as we go from detection, to isolation, and estimation, but if FDI capabilities are increased, costly hardware redundancy can be reduced. Generally for failure detection algorithms there is a trade-off between performance and speed of response, and performance generally increases with complexity, although reliability may not (e.g. sensitivity increases with complexity). For example, if one were to incorporate *a priori* knowledge concerning specific failure modes, both complexity and performance increase. The upper bound for complexity is dictated by the complexity-computational burden trade off. The implementation of

an FDI system may also require special failure sensors to be added or that current ones be properly repositioned [4].

2.2. PILOT/AVIONICS INTERFACE IN FDI

Pilot performance in the event of a failure is improved if only the most vital information is presented when a failure occurs. The FDI system can aid in filtering out insignificant data, could be particularly helpful by giving a graphical display of the failure problem, and could suggest corrective actions [5–7]. There is also a need for flow of information from the pilot to the FDI system. For instance, when a pilot is trying to determine what has failed in the aircraft, often he will perform, possibly at the suggestion of the FDI system, some form of 'active test'. That is, he will perturb the system and observe its response. From this he can formulate a failure hypothesis which could be used in the FDI system to validate a failure hypothesis or to resolve failure hypothesis conflicts [8]. Avionic systems can provide valuable information for the failure diagnosis task. For example, a weather monitoring system could provide cold temperature information which, if used in the diagnosis, would increase the probability of a stuck actuator. Failure information will also be given to the avionic systems for control law reconfiguration and mission planning. For example, estimation information will provide for the accurate determination of the failed aircraft model which is useful in reconfiguration.

3. FDI System Description

A general FDI system block diagram is shown in Figure 1. The *Inputs* consists of the pilot commands for degrees of elevator, rudder, etc. The *Outputs* are airspeed, pitch angle, etc. The *Other Information* may be the state vector x, or it may also be actual
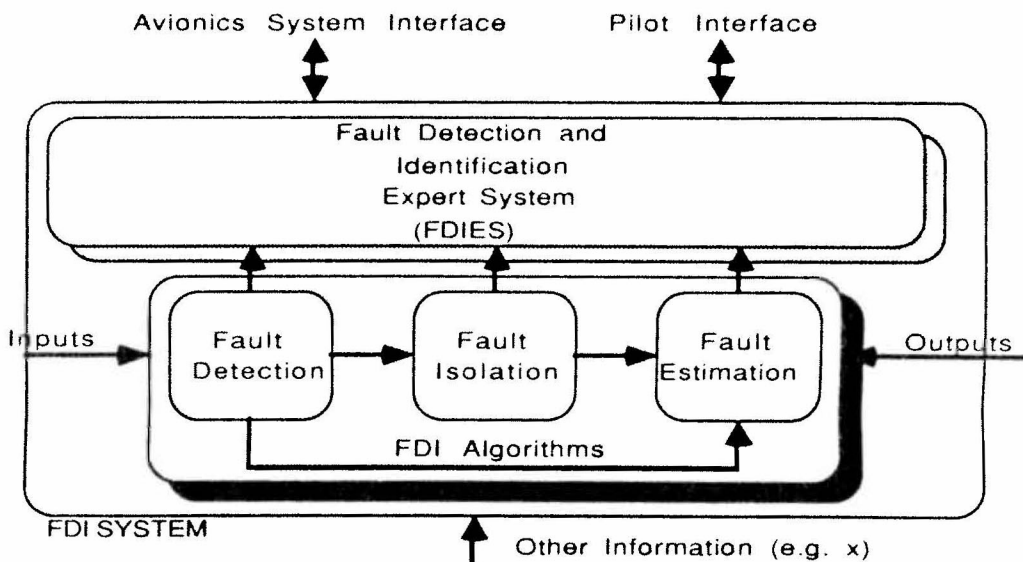

Fig. 1.   Fault detection and identification system.

surface positions, or special information from failure sensors. All the FDI subsystems have access to the information from the FDI system interface (e.g. u, y). The fault detection algorithms monitor the systems for faults. Upon detection all other subsystems are alerted, and using aircraft data the isolation and estimation subsystems begin identifying the failure. Concurrently, FDIES begins informing the pilot and other avionic systems. When it receives data from the algorithms, it begins a failure diagnosis which also considers the pilot and avionics information that is provided. Upon completion of the diagnosis, the processed information from the algorithms is provided to the pilot (e.g. 'Elevator jammed at 19°') and avionics systems for failure accommodation.

4. Algorithmic Approaches to FDI

Algorithmic approaches to FDI are well studied and basically all address the problems with *voting* techniques to achieve analytical rather than hardware redundancy [3, 9, 10]. There is, however, no unified algorithmic approach especially when one considers the necessity to extract essential information for the pilot and avionic system's failure accommodation task. For the present study, following the system vulnerability studies in [15], the failure detection system was designed to monitor the system at the lowest level (hardware level) for failures. We concentrated on actuator failures and used the failure detection and identification filters (FDIF) described in [15, 16] for isolation and estimation. One could also use multiple hypothesis filter-detectors, jump process formulations, or an innovations based approach such as the generalized likelihood ratio method [11–14]. In Section 6 we show how to integrate the AI and algorithmic approaches to FDI.

5. Artificial Intelligence Approaches to FDI

5.1. INTRODUCTION

The field of artificial intelligence (AI) is receiving an increasing amount of attention from the scientific and engineering communities. In particular there has been mounting excitement over a field of applied AI called '(Knowledge Based) Expert Systems'. The excitement should not however, eliminate an objective analysis of where they are most useful in FDI. Since this type of problem has only recently developed no firm criteria exist for the choice; however, some guidelines can be useful [2]. It is our belief that using AI techniques for problems that are better suited for an algorithmic approach, generally produces complicated, computationally intensive AI programs with large portions of code used for nonsymbolic data processing, and the necessity to ignore certain vital information about the system such as noise, continuous change, delays, and time. We also believe that using algorithmic techniques for problems that are better suited for an AI approach generally produces a breakdown of 'unified' algorithmic approaches, and an inability to deal with combinatorial explosion,

heuristics, and symbolic information. Combinatorial explosion occurs in FDI since we try to find the change in the underlying structure of the system which is causing the difference in behavior [19]. Often there is an overwhelming number of possible failures. We use these guidelines to justify the various FDI subsystem divisions.

In Section 2.2, 'Pilot/Avionics Interface in FDI', there were several issues which suggested the need for AI techniques. These were:

(1) The proper presentation of failure information to the pilot for accommodation.
(2) The need for the capability to use the pilot's inputs in the diagnosis.
(3) The need for a coordinated *Active test*.

The first issue is often studied under the title of 'natural language interface' or 'expert interface' and is also a topic to be addressed in the reconfiguration portion of the intelligent controller. For the second issue, an expert system approach enables the pilot to put information into the diagnostic process. Moreover, if the pilot does not have any inputs, the performance of the expert system should degrade gracefully [17]; the expert system can conclude that no diagnosis is possible with the current information or present a diagnosis with a small attached certainty. In this case the expert system could suggest an active test so that the FDI system and the pilot can gather more information, so that a more certain diagnosis could be performed. The expert system could provide a facility for the pilot to ask the system to give an explanation of why it came up with a diagnosis. The expert system is designed to capture the knowledge of how an expert pilot would perform the diagnosis task. Judging from some pilots' past successful recoveries from major failures [1] this seems useful.

As discussed in Section 4, there are currently many good algorithmic approaches to parts of the FDI problem; however, there are not any general solutions. The consequence of this is that there is a need to combine and interpret various bits of dynamic information from different algorithms. That the inference engine of an expert system is valid for such a dynamic environment is evidenced by the fact that expert systems are designed to gather data, arrive at conclusions, decide upon actions, and carry them out in an environment of continuously changing circumstances [18]. Hence, as shown in Figure 1, we use good algorithmic approaches at the lowest level, and an expert system at the highest level to interpret the algorithmic data, perform failure diagnoses, and to interface to the pilot and avionic systems. Next we shall develop the expert system.

5.2. COGNITIVE MODELING OF PILOTS

In the expert system approach to FDI we begin by studying how to model the behavior of an expert pilot at a fault diagnosis task [20–23]. In [8] the author combined such studies with interviews of pilots and formulated a model of their behavior. This is depicted in Figure 2 below. The system contains a fault monitoring portion which drives the diagnosis portion. The fault monitoring system is the
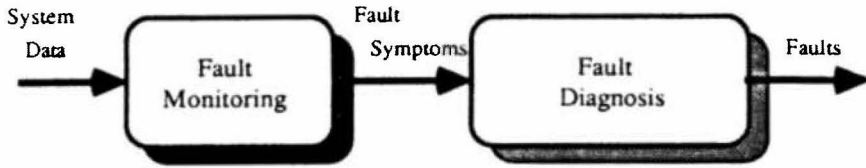
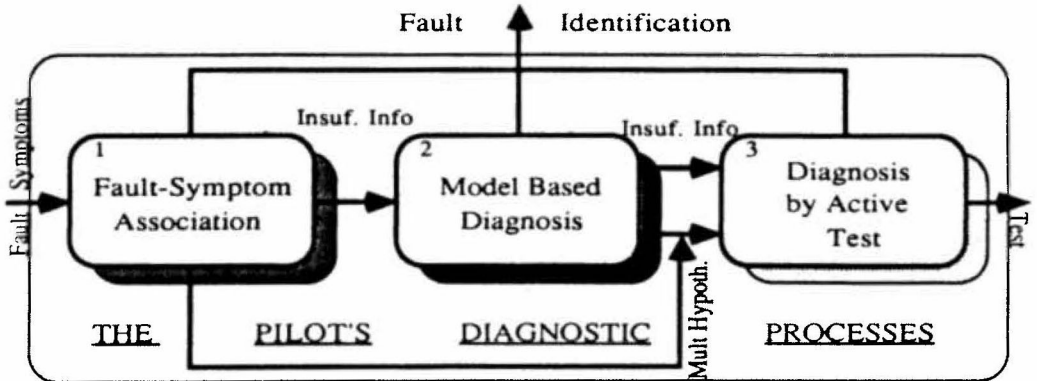Fig. 2. General diagnostic process.

Fig. 3. Pilot failure diagnostic processes.

hardware system that alerts the pilot when there is a suspected failure. The fault diagnosis block will be the model of the expert pilot's behavior in response to the failure symptoms obtained from the monitoring system. A detailed illustration of the fault diagnosis block is given in Figure 3 [8]. The pilot receives fault symptoms from the fault monitoring system and associates them with commonly occurring faults. If there is a match, then fault identification is immediate, if not then he reasons from his knowledge of how the aircraft acts. That is, he has a *model* of the aircraft's behavior in mind and compares how the plane should act, with how it is acting, to make conclusions about the origins of the fault. If he has insufficient information to perform the diagnosis or has multiple hypotheses he may decide to perturb the system in a controlled manner so that more information can be gathered. This is called an Active Test. Any step in the process may be able to identify the fault, but steps 1, 2, 3 take increasingly more time to perform. Notice that due to the feedback inherent in the process one could perform steps 1, 2, and 3, get the results of the active test and succeed in fault identification via step 1. This reflects the human's ability to *prune* the solution search space.

5.3. AI TECHNIQUES FOR FDI

In the following we will outline previous results on deep and shallow reasoning that will be used to model various portions of the pilot's diagnostic process. AI-FDI methods can be broken into two categories depending on whether they use *shallow* or *deep knowledge* about the domain of interest [18]. The shallow knowledge approach depends on prespecified relationships between fault symptoms and malfunctions,

while the deep knowledge approach utilizes intricate information about the structure of the system.

5.3.1. *The Shallow Knowledge Approach*

In the shallow knowledge approach, associations between symptoms of malfunctions and malfunctions themselves are made empirically. Typically, in the shallow knowledge approach, production rules are used to emulate the knowledge of a pilot who is very good at failure diagnoses. An example system that could be a candidate for this approach would be a MYCIN-type expert system.

The major bottleneck of the formation of an expert system of this type is proper knowledge acquisition and organization. This is the case since (1) enough knowledge must be encoded to cover a wide range of malfunctions, (2) the rules are formed empirically, (3) the knowledge base is specific to the particular plant and, (4) the number of rules may exceed practical limitations considering that many failures can occur over a continuum of values. To solve some of these knowledge acquisition and organization problems, the deep knowledge approach is used for parts of the FDI system.

5.3.2. *The Deep Knowledge Approach*

When we consider the knowledge about aircraft behavior, physical laws can be incorporated into the diagnosis. Since the underlying principles of the domain are encoded in the knowledge, the need to consider every possible fault scenario is eliminated and more complex diagnosis tasks can be addressed. We shall discuss three deep knowledge approaches here.

In the *causal search* approach [18, 26], diagnosis is viewed as the task of tracing process disturbances back to their source. The disturbances are *linked* to their sources by cause and effect relationships. These relationships can be just binary (e.g., yes, a failure in the right stabilator can adversely affect pitch) or it could have attached a magnitude of its effect. For even more complex systems one can attach the associated delay between cause and effect or probabilities. One must limit the addition of such information or efficiency can be degraded.

The *governing equations* method [18, 24], involves associating the differential equations of the aircraft with certain fault conditions that will be violated if a failure occurs. The diagnosis proceeds by a logical inference on the pattern of violated fault conditions. This amounts to comparing expected plant behavior to current behavior by examining the outputs of the plant and checking whether their difference is greater than some threshold. Notice that the analog of this method in the algorithmic approaches is the innovations approach.

In the *hypothesis/test* method [18, 19] a series of hypotheses are formulated, tested, and selected. The approach emulates a human postulating a cause for a failure and

comparing hypothesized behavior with actual aircraft behavior. As with the algorithmic approaches of this type it can be computationally demanding. There are virtually an infinite number of types of failures possible on the aircraft so a correspondingly large number of hypotheses must be stored or generated. The hypothesis/test method is different from the governing equations method in that it can also be used for higher level diagnostic reasoning.

5.3.3. *Combining Shallow and Deep Inference*

It is important to note the advantages and disadvantages of each of the techniques above. For instance, although the shallow knowledge approach does have knowledge acquisition and organization disadvantages, it does have the advantage that high level diagnostic heuristics can be added to the FDI process. An example of such a heuristic might be 'use the governing equations method first, if after one second it fails, try a causal search'. This can result in a more efficient diagnostic process. The disadvantages of the deep knowledge approaches, such as computational complexity, are avoided by incorporating the algorithmic techniques into the FDI system. This points to the necessity to use both AI and algorithmic approaches to FDI. Only in this way can efficient systems be built.

6. An AI/Algorithmic Approach to FDI

Next, we introduce the architecture for the FDI system, choose reasoning techniques for the subsystems, and introduce the knowledge representation graph to integrate the various knowledge representation and reasoning techniques and facilitate the expert system implementation.

6.1. FDI SYSTEM BLOCK DIAGRAM

In this section we combine the algorithmic and AI approaches to FDI to obtain a hybrid system. The general FDI block diagram with the expert system architecture included is shown in Figure 4. We add the expert interface to make for a friendly interface to the pilot, to parse the information given to various avionic systems, and to carefully manage information flow into the system. The information from the FDI algorithms such as the FDIF and fault detection systems is made available to all the reasoning processes in the expert system. The aircraft data is made available to all parts of the system. Notice that an 'Executive' sub-module was added. This module represents the highest level diagnostic reasoning processes of the pilot. It manages information flow at a high level and makes decisions about, for example, switching the type of diagnosis being performed. Each sub-module is shadowed with a different shade. The black indicates a pure algorithmic approach. The other three shades range from dark to much lighter corresponding to deeper, deep, and shallow reasoning. The
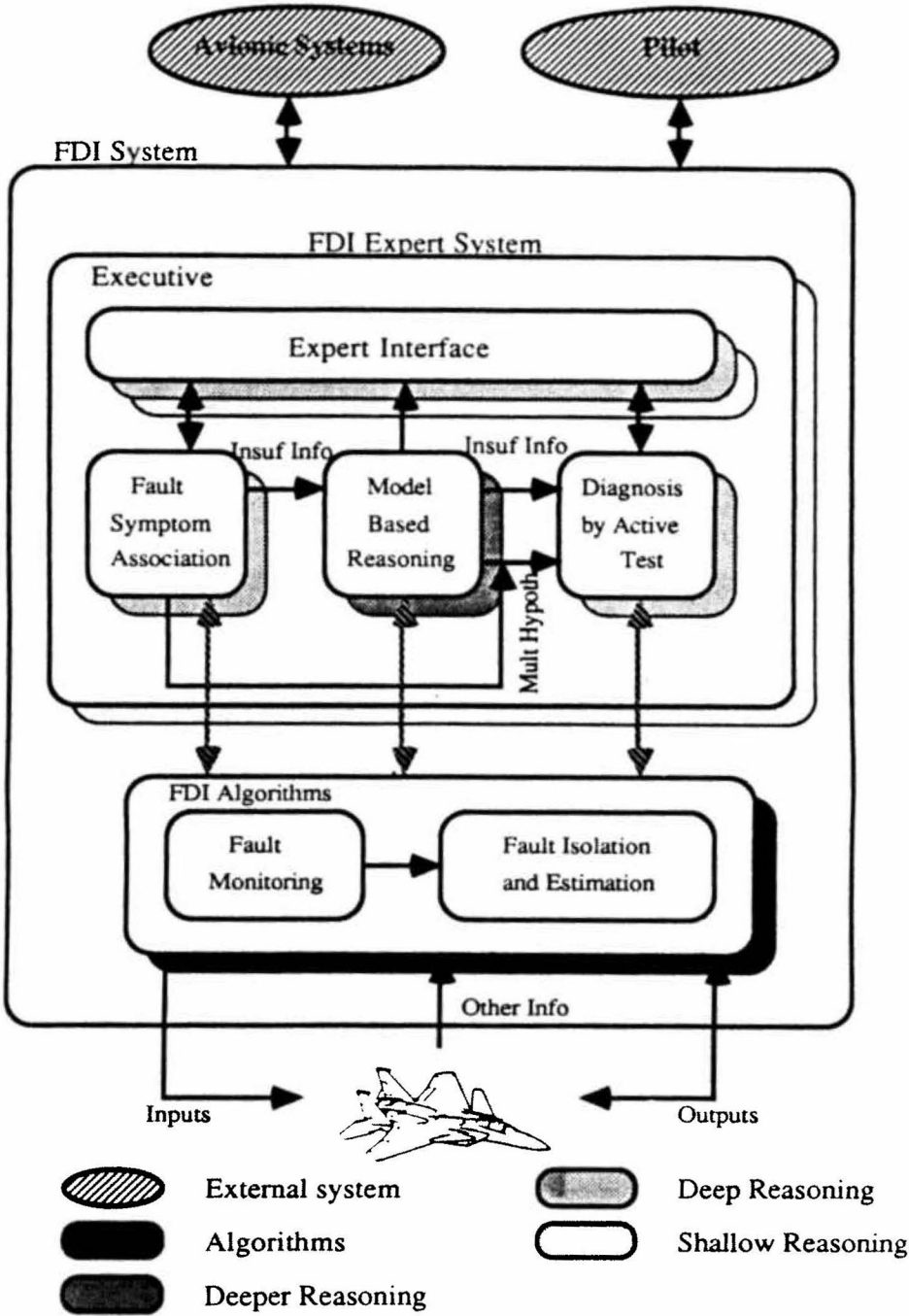
Fig. 4.   FDI system block diagram and expert system architecture.

fault symptom association module uses the deep knowledge approach of 'governing equations', while the model based reasoning module uses 'causal search'. The active test module is in between a shallow and deep approach in that it must work with deep knowledge information, multiple hypotheses from deep reasoning processes, algorithmic data, and some type of coordination with the pilot. It is for these same reasons that the expert interface uses both shallow and deep knowledge approaches. The Executive uses only shallow reasoning.
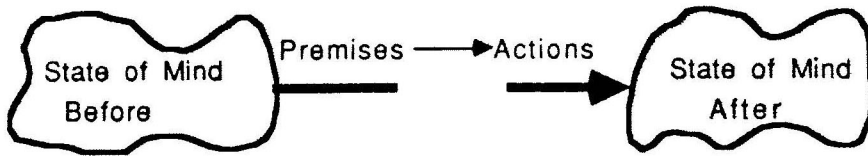
Fig. 5.   Knowledge representation graph building block.

6.2. KNOWLEDGE REPRESENTATION

Since the FDI approach taken here is a hybrid one, the knowledge representation (KR) schemes normally chosen will be quite distinct. For instance, for the deep knowledge approach one might build a causal model whereas for the shallow approach one would carefully begin writing rules. If we suppose that our expert system will be *rule based* then the lack of homogeneity between these representations can be minimized in the following manner. Regardless of which KR scheme is used, for the rule based approach, our thought process is:

(1) We are in some state of mind where we are considering what to do next,
(2) Some particular premises become satisfied which lead us to
(3) Execute some actions; consequently we find ourselves back at step (1).

The inference mechanism is embedded in these three steps. We represent this graphically in Figure 5. In proceeding between the two states of mind, when the Premises are true, the Actions occur instantaneously. In the general case we obtain a connected graph where the structure and organization of the knowledge becomes evident. We shall call this the KR graph. The effect of the choice of inference strategy on how the graph is traversed also becomes clear. For the shallow and deep knowledge approaches all that changes is the type of states of mind, premises, and actions.

6.3. KR FOR THE FAULT DETECTION AND IDENTIFICATION EXPERT SYSTEM

The knowledge representation for the Fault Detection and Identification Expert System (FDIES) whose architecture is shown in Figure 4 is developed here. The development begins by giving an example of KR for the fault symptom association module.

6.3.1. *KR for the Fault Symptom Association Module*

This module gets inputs from the FDI algorithms and tries to associate various sets of failure indicators with pre-specified failure scenarios. The deep knowledge approach of governing equations is used. The 'Premises' of Figure 5 are logical combinations of the failure indicators and the 'Actions' are results of the associations such as 'failures indicate that the right stabilator is stuck'. They could also be 'the right stabilator actuator appears to have failed but there is insufficient information to determine exactly how'. In this last case the module fails to identify the failure and

the Executive decides to invoke model based diagnostic reasoning. As an example of the KR graph knowledge representation scheme, we consider actuator failures and use the FDIF to provide the algorithmic information to the expert system. Make the following definitions: (propositional connectives & means *and*, V means *or*, - means *not*, --> means *implication*).

States of Mind:	M_1 = Waiting for a failure
	M_2 = Suspect a failure
	M_3 = Confirmed failure, What is it?
Premises:	i = Refers to the ith actuator
	failure(i) = Indication from the fault monitoring system that some type of failure has occurred in the ith actuator
	fdif(i, f, v) = Information from the FDIF about the ith actuator being f = stuck or f = biased at magnitude of v
	fm(i, stuck) = Indication from the failure monitor that the ith actuator is stuck
Actions:	wait(T_d) = Wait T_d seconds
	fsa-diagnosis(h) = Fault Symptom Association module's diagnosis; either h = valid, lack enough info. for diagnosis, or none made yet
	stuck(i, v) = ith actuator stuck at position v
	biased(i, v) = ith actuator biased with the value v.

With these definitions we develop the KR graph. It is shown in Figure 6. Consider one possible thought process: In state of mind M_1, we are waiting for a failure. Once the detection algorithm flags a failure, we wait T_d seconds, then decide to confirm the failure. If the failure is confirmed we take no actions but change our state of mind to M_3; deciding what the failure is. Using the governing equations method we make logical inference on the combination of failures. This is represented graphically by the three arcs leaving M_3. For example, on the top arc leaving M_3, if the failure monitoring system is still indicating a failure and it is indicating that the ith actuator is stuck and the FDIF also indicates a 'stuck' failure then our diagnosis is that we have
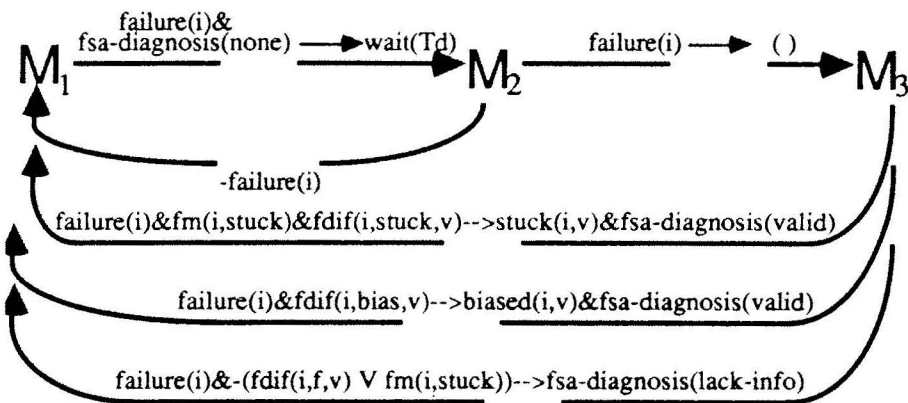

Fig. 6.   KR graph for fault symptom association module.

the ith actuator stuck and the diagnosis is valid. If the failure combination does not match any of the premises associated with the three arcs leaving M_3 then the Executive automatically switches to model based reasoning.

6.3.2. *KR for the Model Based Reasoning and Active Test Modules*

Here we use the deep knowledge approach of a causal search and hence must develop a causal model of our system. A more detailed example of the development of a causal model for an aircraft is given in [25, 26]. We consider how an actuator failure will affect certain aircraft variables and develop the simple causal model shown in Figure 7. To develop the KR graph the following definitions are made:

States of Mind: M_4 = What aircraft variables are misbehaving?
                      M_5 = What failure would produce this behavior?
                      M_6 = What conclusions can be made?

Premises/Actions: x = An aircraft output
                      misbehave(x) = Aircraft variable x is misbehaving
                      (e.g. x = pitch, yaw)
                      mbr-suspect(x) = Suspect failure in the aircraft variable x
                      mbr-diag(i, n) = Model Based Reasoning diagnosis is that
                      n = suspect or no conclusions possible
                      (none) for the ith actuator

The KR graph is shown in Figure 8. From state of mind M_4 we determine which aircraft variable is misbehaving then from M_5 we make the causal relationship between the misbehaving variable and the actuator that we suspect has failed. From M_6 we make the diagnosis.
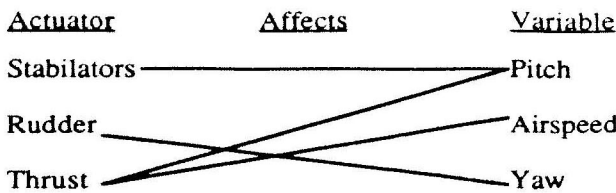

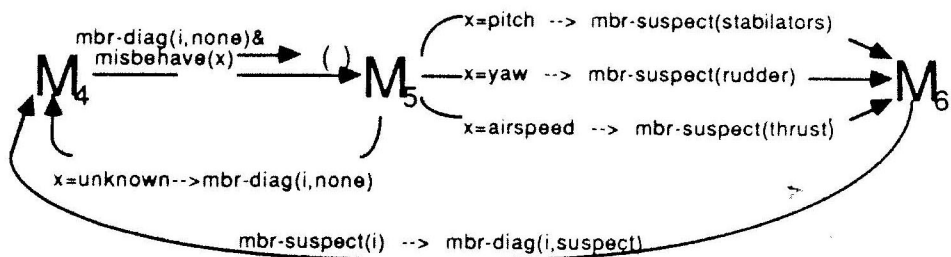Fig. 7. Simple causal relationships.

Fig. 8. KR graph of model based reasoning.

Due to our guidelines in the choice of whether to use AI or algorithmic techniques we chose not to use an extremely complex causal model. Reasoning over a very complex causal model can easily become computationally unmanageable. Due to the simple model's lack of resolution all that is produced is either a suspected diagnosis or none at all. For the diagnosis by active test module we use a combined shallow and deep knowledge approach. The expert knowledge of what active test should be chosen is loaded into the KR graph. This graph was produced for the implementation. The only significant difference with it is that some 'actions' call for communication with the pilot as they do in the expert interface.

6.3.3. *KR for the Expert Interface to the Pilot*

Here we study only a portion of the full expert interface. We do not address the problems associated with the natural language interface, or 'friendliness' problems. What we propose is that an up-to-date model of the current failure status of the aircraft be kept inside the expert interface. The diagnoses from all modules and the FDI algorithms will update this model. Information on this model will be accessible by both the avionic systems and the pilot. Again we can use a KR graph approach but with some differences:

(1) We change *states of mind* to *locations* in which parts of the process dwell (e.g., whether an actuator is floating in the up or down position).
(2) The 'Premises' will take the form of logical combinations of events such as a position change of an actuator or the outcome of a diagnosis.
(3) The 'Actions' will be communications to the pilot and avionic system.

To form the KR graph for the ith actuator define the following:

Locations: U -up   D -down   S -stuck
UB -Surface in the up position with a bias failure
DB -Surface in the down position with a bias failure
UF -Surface floating in the up position
DF -Surface floating in the down position

Premises: $pd(i)$ = event that the position of the surface goes down
$pu(i)$ = event that the position of the surface goes up
$s(i)$ = ith actuator jammed, $b(i)$ = ith actuator biased, $f(i)$ = ith surface floating
$atd(x)$ = active test diagnosis x

Actions: $pilot(x, y)$ = alert pilot/avionic systems where: x is either $pu(i)$ or $pd(i)$, and y is $s(i)$, $b(i)$, or $f(i)$

The KR graph is shown in Figure 9. The events $pd(i)$ and $pu(i)$ continually update the failure model and the informaton such as $s(i)$, $f(i)$, and $b(i)$ is the result of
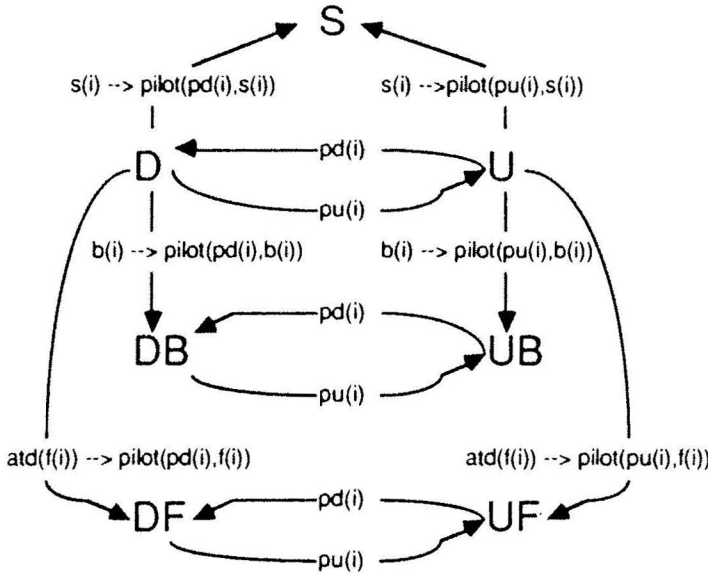
Fig. 9.   KR graph for portion of the expert interface.

diagnostic reasoning done in the system. For instance, if the ith actuator is in the down position and it becomes stuck, the pilot is informed and the failure model goes to the 'stuck' state.

6.3.4. *Remarks on Knowledge Representation/FDIES*

The KR graphs provide a nice tool to check consistency and completeness of knowledge. They also illustrate the structure of the knowledge. The characteristics of the inference strategy of a production system (OPS5) [27] can be defined in terms of the KR graph. For example, the *conflict set* is a subset of the arcs emanating from a state of mind (the ones with their associated premises satisfied). *Refraction* is seen from the fact that one moves forward on the graph (when forward chaining is used). *Recency* is illustrated by examining any particular state of mind; if we look *back*, the working memory element created as the result of the last Action has the highest recency, and so on. We see *recency-specificity* by examining the number of premises of each arc emanating from a particular state of mind. *Test-specificity* is illustrated by examining the number of relational tests contained in particular premises of each arc emanating from a particular state of mind. Efficiency issues in the production system become clear when the KR graph is used. For example, it is desirable to minimize the size of the conflict set and to properly order the premises. Each of the KR graphs produced for the actuator failure case is modular. That is they represent the diagnostic processes needed no matter which particular actuator failed. This feature essentially reduces the need for say k rules to k/m rules, where m is the number of actuators. This is a good example of why the number of rules alone is a poor measure of the expert system's quality. The KR graphs are good documentation vehicles and make the task of explaining the actions of the expert system easier.
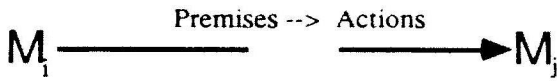
$$M_i \xrightarrow[\text{Premises} \to \text{Actions}]{} M_j$$

Fig. 10. General KR graph link.

```
(p rule-name (Mᵢ) (Premises) --> (Actions) (Mⱼ) )
```

Fig. 11. General OPS5 rule.

6.4. IMPLEMENTATION OF OPS5

6.4.1. *Translating the KR Graph to OPS5*

If the KR graphs are specified, then programming the expert system in OPS5 [27] is nearly automatic. Consider the general link in the KR graph shown in Figure 10. The translation of this to a general OPS5 rule produces Figure 11. Where M_i is properly set up as an element class and M_j is input into working memory in an appropriate manner. This characteristic was not obtained free of charge; the work has been transferred into the formation of KR graphs.

6.4.2. *FDIES. Example Diagnoses*:

Using the KR graphs an expert system was developed in OPS5 to perform the failure detection and identification task. With the algorithmic data from the FDIF, several actuator failure scenarios were created for the expert system. Besides these failure scenarios, the expert system allowed the user to enter failure scenarios. FDIES would then begin reasoning from the data, first with fault-symptom association (implemented with the governing equations approach to diagnosis). For the biased and jammed cases, FDIES performed the diagnosis task using only fault-symptom association. This is the case since the FDIF algorithms performed well for these failure types. For the actuator floating case the fault monitoring system appropriately detected a failure but the FDIF algorithms could not estimate the failure. Fault symptom association failed and the executive switched the reasoning method to model based reasoning using causal search; consequently the system determined which actuator it suspected as having failed. The executive switched the type of reasoning to 'active test' and using the 'suspect hypothesis' from the model based reasoning module it suggested to the pilot (FDIES user) how to perturb the aircraft so that more failure information might be gathered. The failure diagnosis for the actuator jammed case is successful at this step. In any of the cases described above, once a diagnosis is successful, the executive interrupted, the failure model in the expert interface was updated, and the pilot and avionic systems (user) were alerted to the diagnosis results.

7. Conclusions

The fault diagnosis problem on an aircraft is appropriate for AI techniques. An expert system approach can produce an invaluable assistant to the pilot when a failure

occurs. The architecture of the FDI system is necessarily hierarchical. We take advantage of well understood algorithms to perform some of the detection and identification and use AI techniques to solve some of the less well structured problems in the FDI process. To unify the representation of all sorts of information used in the FDI of a physical system, the KR graph was introduced. Besides being a good knowledge acquisition and organization tool, since it helps to clarify complex relationships in the knowledge base, it also helps to implement and document the expert system.

The work presented here represents the first step at the development of a complete FDI system. A number of issues need to be examined. The interface between the pilot and avionics must be considered in detail, additional algorithms need to be developed, and many other types of failures need to be considered. It is hoped that the necessity for the integrated AI–algorithmic approach has been established in this paper.

Acknowledgement

References

1. Montoya, *et. al.*, Restructurable controls conference, NASA Langley Research Center, NASA CP-2277 (1979).
2. Passino, K. M., Restructurable controls and artificial intelligence, McDonnell Douglas Corp. Internal Report, MDC-IR0392 (April 1986).
3. Willsky, A. S., A survey of design methods for failure detection in dynamic systems, *Automatica*, **12**, 601 (1976).
4. Cunningham, T. B., Failure management technology for high survivability, Honeywell (1980).
5. Hanson, D. C., *et. al.*, Aircraft alerting systems standardization study: Phase IV, Accident implications on systems design, Dept. of Transportation/FAA/RD-82-26 (1982).
6. Rouse, W. B., Understanding and aiding the human in fault diagnosis tasks, Proc. Joint Automatic Control Conf. (1979).
7. Rouse, W. B., Human problem solving performance in a fault diagnosis task, *IEEE Trans. Systems, Man, and Cybernetics*, 258–271 (April 1978).
8. Abbott, K. H., Exploration of expert systems concepts for onboard diagnosis of faults in a turbofan aircraft engine, Proc. American Control Conference, Boston (1985).
9. Iserman, R., Process fault detection based on modeling and estimation methods – a survey, *Automatica*, **20**, 387 (1984).
10. Himmelblau, D. M., *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*, Elsevier, NY (1978).
11. Willsky, A. and Jones, H., A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems, *IEEE Trans. Automatic Control* **21**, 108–112 (1976).
12. Deyst, J. J. and Deckert, J., RC5 jet failure identification for the space shuttle, *Proc. IFAC* (1975).
13. Chow, E., *et. al.*, A dual mode generalized likelihood ratio approach to self-organizing digital flight control system design, Research Report to NASA, MIT (April 1975).
14. Middleton, D. and Esposito, R., Simultaneous optimum detection and estimation of signals in noise, *IEEE Trans. Information Theory* **14**, 434 (1968).
15. Beard, R. V., Failure accommodation in linear systems through self-reorganization, PhD Dissertation, MIT (1971).

16. Jones, H., Failure detection in linear system, PhD Dissertation, MIT (1973).
17. Waterman, D. A., *A Guide to Expert Systems*, Addison-Wesley (1986).
18. Moore, R. L. and Kramer, M., Expert systems in on line process control, *3rd Int. Conf. Chemical Process Control*, Asilomar, CA, (Jan 1986) (see also 1985 American Control Conference).
19. Bobrow, D. G., Qualitative reasoning about physical systems: An introduction, *Artificial Intelligence*, **24**, 411 (1984) (Special issue on qualitative reasoning about physical systems).
20. Rouse, W. B., A model of human decision making in fault diagnosis tasks that include feedback and redundancy, *IEEE Trans. Systems, Man, and Cybernetics* 237–241 (April 1979).
21. Rouse, W. B., Problem solving performance of maintenance trainees in a fault diagnosis task, *Human Factors* **21**, 195–203 (1979).
22. Rouse, W. B., A model of human decision making in a fault diagnosis task, *ibid*, No. 5 (1978).
23. Rasmussen, J. and Rouse, W., *Human Detection and Diagnosis of System Failures*, Plenum Press, NY (1980).
24. Gertler, J. and Singer, D., Augmented models for statistical fault isolation in complex dynamic systems, *Proc. American Control Conference, Boston* (1985).
25. Stengel, R., AI theory and reconfigurable control systems, Princeton Univ. Report 1664-MAE (June 1984) (see also 1665-MAE with D. Handelman).
26. Handelman, D. A. and Stengel, R. F., Combining qualitative and quantitative reasoning in aircraft failure diagnosis, *AIAA Guidance Navigation and Control Conf.* (1985).
27. Brownston, *et. al.*, *Programming Expert Systems in OPS5*, Addison-Wesley (1986).