NASA Technical Memorandum 102767

Robust Fault Diagnosis of Physical Systems in Operation

Kathy H. Abbott

LANGLEY
IN-03
1578
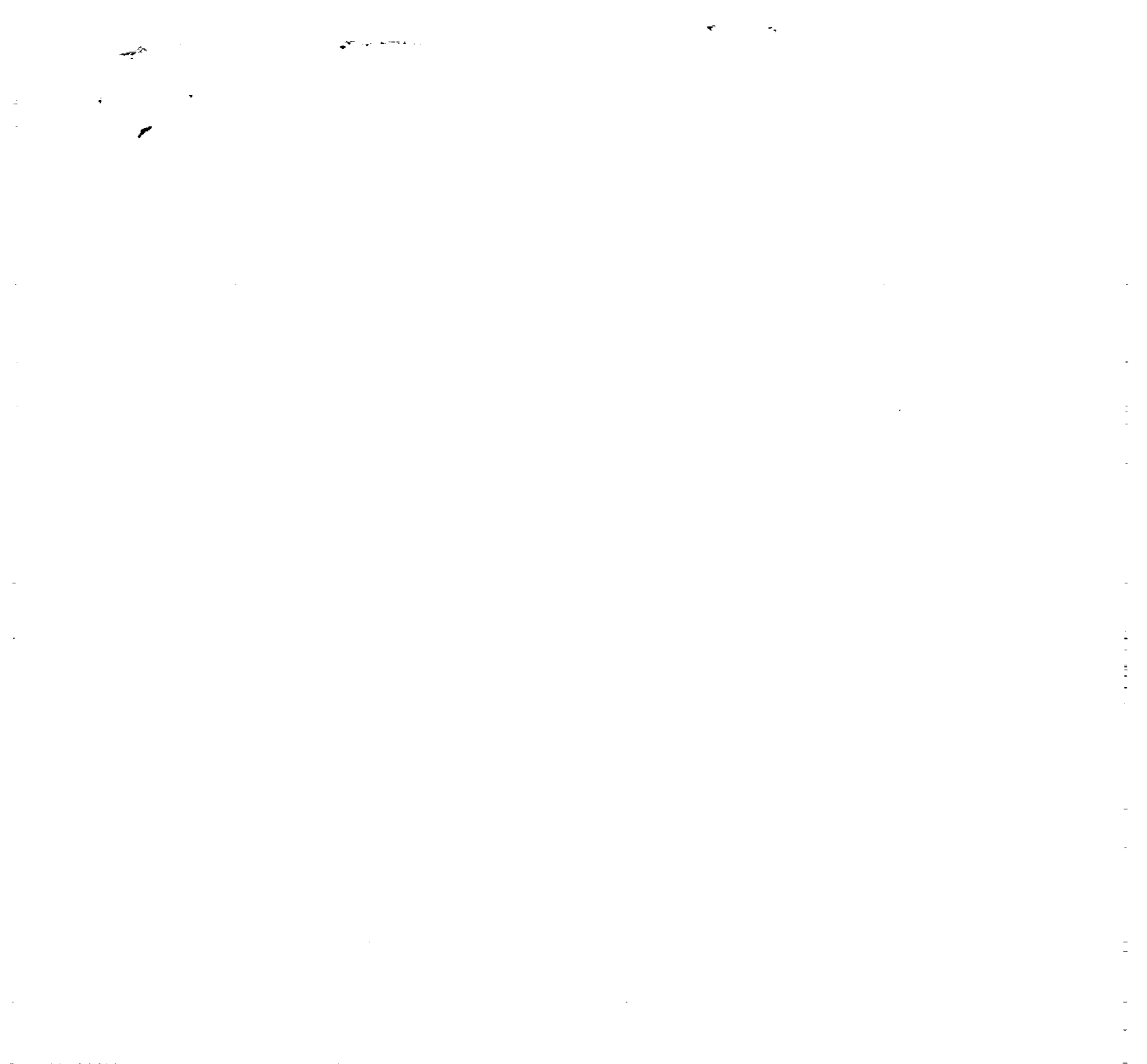*p166*

January 1991

NASA

ABSTRACT OF THE DISSERTATION

ROBUST FAULT DIAGNOSIS
OF PHYSICAL SYSTEMS IN OPERATION

by KATHY HAMILTON ABBOTT, Ph.D.

Dissertation Director: Professor Louis Steinberg

This thesis presents and demonstrates ideas for improved robustness in diagnostic problem solving of complex physical systems in operation, or operative diagnosis. The first idea is that graceful degradation can be viewed as reasoning at higher levels of abstraction (less detail) whenever the more detailed levels prove to be incomplete or inadequate. A form of abstraction is defined that applies this view to the problem of diagnosis. In this form of abstraction, named status abstraction, we define two levels. The lower level of abstraction corresponds to the level of detail at which most current knowledge-based diagnosis systems reason. At the higher level, this thesis presents a graph representation that describes the real-world physical system. The thesis demonstrates an incremental, constructive approach to manipulating this graph representation that supports certain characteristics of operative diagnosis. We show the suitability of this constructive approach for diagnosing fault propagation behavior over time, and for sometimes diagnosing systems with feedback. We also show a way to represent different semantics in the same type of graph representation to characterize different types of fault propagation behavior. We demonstrate an approach that treats these different behaviors as different fault classes, and the approach moves to other

classes when previous classes fail to generate suitable hypotheses.

These ideas are implemented in a computer program named Draphys (Diagnostic Reasoning About Physical Systems) and demonstrated for the domain of inflight aircraft subsystems, specifically a propulsion system (containing two turbofan engines and a fuel system) and hydraulic subsystem.

Acknowledgements

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Diagnosis of complex physical systems can be a difficult business, and automation of this diagnostic process for real-world physical systems must address many important issues. One particularly important issue is *robustness*, or the ability to reason about a variety of faults in a reasonable way, including the ability to degrade gracefully in response to unanticipated inputs. Another important issue is ability to reason about fault behavior in *complex physical systems*, including feedback and fault propagation. This thesis research addresses aspects of both of these issues. In particular, the research is concerned with diagnosis of complex physical systems in operation, or **operative** diagnosis.

Robustness is important because a diagnostic system cannot be designed to specifically anticipate every possible fault situation. Such situations are inevitable, because the number of ways a complex system can fail is so large, and our ability to identify every possibility in detail is limited. One goal of the research is to identify and explore ideas for achieving robust problem solving behavior in diagnosis in general, and operative diagnosis in particular.

Fault behavior in physical systems can be complex in many ways. One such way is that a single initial fault can have multiple consequences. In operating physical systems, this set of consequences can increase as time passes and the effect of the fault propagates. These systems are often very interconnected and have feedback, making the fault propagation behavior more complex. Therefore, another goal of the research is to explore ways to diagnose such systems while in operation.

The research explores these issues in the domain of non-digital devices. In particular,

the approach is implemented for an aircraft propulsion system (including two turbofan engines and a fuel subsystem) and hydraulic system in a computer program called Draphys (*D*iagnostic *R*easoning *A*bout *Phy*sical *S*ystems).

The following sections contain a description of the problems addressed by the thesis research. Current approaches to these problems are discussed, then the specific contributions of this thesis research are presented. The assumptions on which the work is based are summarized and an overview of the thesis is given.

1.1 The Problem: Diagnosis of Complex Physical Systems

Diagnosis is the process of determining why some object is not behaving as it should. Diagnosis of complex physical systems presents particular challenges, some of which are addressed by this thesis research. We identify these challenges below, first examining diagnosis of complex systems in general, then the additional issues associated with operative diagnosis. We then discuss robustness. Robustness in problem solving means the ability to handle a variety of inputs in a reasonable way, even if the inputs have never before been encountered. This thesis addresses two aspects of robust problem solving in diagnosis: graceful degradation and multiple classes of faults.

Before we describe these challenges of diagnosing complex physical systems, it is appropriate to define some of the basic terminology used throughout the thesis. A *physical system* (or just system) is the object being diagnosed; e.g., a human body or an aircraft engine. A *device* or *artifact* is a physical system that was designed and built by humans. A *component* is some piece of the physical system; e.g., an engine is a component of an aircraft. Components may be made up of subparts which are also components; for example, the engine is made up of the fan, compressor, combustor, and other components. A *fault* in a component is something that has broken or gone wrong with that component; e.g., a fuel line leak (although the description of what is wrong with a component can be provided at varying levels of detail). *Behavior* is what the system does, usually in the form of parameter values that describe various aspects

of operation; e.g., oil temperature or pressure. A *symptom* is a discrepancy between expected normal behavior and actual behavior.

1.1.1 Challenges of Diagnosing Complex Physical Systems

Complex physical systems, especially continuous or analog systems, have characteristics which can make diagnosing faults in them challenging. We address several of those challenges in this thesis, including multiple consequences of a single initial fault, especially when the fault does not immediately propagate to all potentially affected components; different manifestations of the same kind of fault; and having sensors which are not optimally placed for diagnostic reasoning.

When a component in a physical system fails, the effect of the fault will propagate, especially (but not necessarily exclusively) to components which normally depend on the faulty component for their proper operation. This fault propagation results in multiple consequences of a single failure, although all the propagation may not happen immediately. Because the components are highly interconnected in physical systems such as aircraft subsystems, the fault propagation can have many consequences. Since many of these physical systems also contain feedback among the components and among subsystems, the resulting behavior of the affected components can be difficult to predict. Sometimes the affected components fail as a result of these faulty inputs, making the diagnostic reasoning even more complex. It is desirable for the diagnostic reasoning to identify those consequences, because corrective actions may be necessary. This is especially true in operative diagnosis, as we discuss below.

Another challenge in diagnosing complex physical systems is that a particular fault may manifest itself as many different behaviors. For example, turbine blades may break off and cause damage in many ways, depending on how many blades separated from the turbine disk, what other blades they damaged, and what other parts of the device were physically damaged by the separated blades. Moreover, we cannot model the behavior of the broken blades in any detail, because we cannot predict the way that the

blade(s) will bounce around, nor can we predict the aerodynamic effects of the break on the blades, among other behaviors. Because the broken turbine blade can manifest itself in so many different ways, the resulting system behavior may differ qualitatively as well as quantitatively. The more detailed the description of the fault, the more different behaviors are possible. This multitude of behaviors means that the mapping from behavior to faults is very complex.

The placement of sensors that provide measurements about the physical system may add additional complexity to the diagnostic process. Sensors provide information on some aspect of system behavior, but not necessarily operational status of the components. For example, in an aircraft fuel system, often there is a fuel temperature sensor. A normal sensor reading for this sensor does not necessarily indicate a normal operational status of the fuel line to which it is attached. In contrast, digital circuits provide measurements of the input or output of components.[1] If the output of a component is incorrect or symptomatic, one can assume that the component is affected by the fault, if only because its input was incorrect. It is important to reason about the relationship of the measurements to the system being diagnosed to understand what exactly is being measured. Other aspects of sensor placement that complicate the diagnostic reasoning include redundant sensors, sensors which are computed (for example, engine pressure ratio is computed from two sensor readings), and lack of sensors on every component.

1.1.2 Operative Diagnosis

Operative diagnosis, or diagnosis of physical systems in operation, is a variant of diagnosis of non-operating systems (such as, maintenance diagnostics), as described in [19], [50], [53], and as discussed below. Particularly important issues associated with operative diagnosis include: the information a fault hypothesis must contain, dynamic fault propagation behavior which means that the diagnosis will take place while the fault is still propagating, and limited testing for additional information. Although these

---

[1]This is generally true at the level of abstraction at which circuit behavior is usually examined.

last two issues can arise in maintenance diagnosis, they are particularly important in operative diagnosis.

A factor which affects the information that a fault hypothesis contains is the purpose for which the diagnosis is being done. In operative environments, the diagnosis is done to facilitate continued operation of the system under consideration. Therefore, the information contained in a hypothesis should support the choice of actions available to support that continued, safe operation. An example of a type of information that could prove useful in an operative environments is the paths of interaction in the device along which the fault is propagating. Knowing the propagation path might lead a human operator to prefer certain actions to prevent further fault propagation. Moreover, identifying the effects of the fault can be important, because it is often necessary to take corrective actions to compensate for the failure's effects, even if affected components are not physically broken. For example, an airplane engine provides power for one of the electrical generators. If the airplane engine fails, the generator will stop running, even though there is nothing physically wrong with the generator itself. Action must be taken to compensate for the lack of power, so the operator must know that the generator is no longer running. In another example, in medicine, the goal of the diagnostician is to prescribe some medication or treatment. Often the symptoms and effects must be treated as well as the disease itself, so knowing the disease alone may not be sufficient. Therefore, the information a hypothesis should contain for operative diagnosis should include the cause of the fault, its effects (or system status), and the fault propagation path.

In maintenance diagnosis, however, the purpose is to determine which part to fix or replace. It is often sufficient to identify the source of the problem, but knowing how the device is faulted, or the effects of the fault, might be unnecessary. For example, in electronic troubleshooting, the purpose is usually to determine which part or component to replace. In other domains, such as maintenance of mechanical devices, it is often desirable to repair the part, so knowing what parts are physically affected by the failure may be important. However, it is only necessary to know the parts that

are physically broken, so in the example of the electrical generator, it would not be necessary for the maintenance diagnostician to repair the generator. As this example shows, the information contained in a fault hypothesis for maintenance is different from the information required for operative diagnosis.

Because of the information contained in a fault hypothesis for operative diagnosis, some automated systems view diagnosis as the process of constructing a model or explanation of the illness or fault, to account for both cause and effects (e.g., [41]). Draphys belongs in this group of systems.

Dynamic behavior of the physical system may create other challenges in operative diagnosis. In an operating physical system, the effect of the fault propagates and the set of symptoms often changes as time (and the fault) progresses. This reflects the characteristic of non-zero-time propagation, where not all effects of a fault happen immediately. In maintenance, the diagnosis is usually done after all propagation has taken place. In operative diagnosis, the propagation often will still be occurring while the diagnosis is performed. Therefore, the set of symptoms that must be used to diagnose faults depends on when the sensor readings are sampled. Moreover, measurements are usually sampled at predetermined intervals, and each time the sample is taken the set of symptoms may change. The timing of these changes can be very difficult to predict, since the time required for fault propagation depends on how the fault manifests itself. However, the changes in the set of symptoms reflecting fault propagation behavior can be a very powerful means of discriminating hypotheses if the diagnostic process knows how to use them. Most current diagnosis systems assume a static diagnosis environment.

Even in maintenance diagnosis that takes place before all fault propagation happens (which occurs much less often than in operative diagnosis), the diagnostic process stops when the source of the fault is identified. In operative diagnosis, even when the fault has been identified, the need for system status information requires the reasoning process to continue to identify when components become affected by the fault.

Another issue associated with operative diagnosis is that the testing for additional information is even more limited than in other types of diagnosis. In these other types of diagnosis, measurements usually can be taken to provide discriminatory information, although this type of testing may be costly. However, in operative diagnosis, any tests to obtain additional information about the system's state are limited to those which do not endanger the system's continued operation. This constraint means that the information available to discriminate hypotheses is sometimes restricted. Tests can sometimes be made by perturbing the physical system with known inputs and observing the resulting behavior, but this must be cautiously done in a faulty system which must continue operating. We assume in this research that the diagnosis must be accomplished with the currently available sensor information.

Since fault hypotheses for operative diagnosis are distinguished by their cause, propagation path and system status, the hypothesis space can be large. Considering multiple types of fault propagation and multiple independent faults makes the space even larger. Therefore, efficiency of exploring this hypothesis space is an important issue. This thesis investigates ideas for efficient management of that hypothesis space by using knowledge in the diagnostic process.

1.1.3 Robustness

Robustness in problem solving means the ability to handle a variety of inputs in a reasonable way, even if the inputs have never before been encountered. This thesis addresses two aspects of robust problem solving in diagnosis: graceful degradation and multiple classes of faults.

Graceful Degradation

Diagnosis is the process of determining why some object is exhibiting abnormal behavior. Many current approaches formulate this problem as the selection of an appropriate solution from a previously enumerated set of possible solutions, where solutions are the

identification of the fault. Choice of one of the solutions is determined by the abnormal behavior of the system, which is often described as symptoms, or discrepancies between expected normal behavior and the actual behavior of the system. These symptoms are matched to the set of known faults based on knowledge about specific[2] fault-symptom associations. However, when faults occur for which there is no associational knowledge, approaches that depend on such knowledge are inadequate. These approaches degrade precipitously, even if the variation of problem inputs is small. A key point here, and one which has been discussed extensively, is that attempting to describe in detail all occurrences of how something can fail is futile [12]. It is futile because there is no way to guarantee completeness.

A well-established approach to overcoming this precipitous behavior is to reason about how the system works, rather than how the system fails [11], [13]. By using models of normal system structure and behavior, and reasoning about discrepancies between the actual system behavior and the model of normal behavior, faults can be diagnosed that are not specifically described within the diagnosis system. As pointed out in [11], a consequence of this reasoning is that one is trading breadth of fault coverage in the model-based approach for specificity of the empirical associations. This is a useful approach to achieving a certain amount of graceful degradation in diagnostic problem solving.

This research expands on that notion, by exploring the idea that graceful degradation can be achieved in a structured way by using abstraction. The basic idea is that graceful degradation is not achieved by simply exploring whether something is known about faults, but at what level of detail is it known. Knowledge at different levels of specificity can provide different fault coverage, but increasing fault coverage is achieved at the cost of degrading specificity. For example, if we know that the input to a component is normal and the output of that component is abnormal, we can infer

---

[2] Here and throughout the thesis, when we refer to specific fault-symptom knowledge, we are referring to knowledge about how a system fails (e.g., fuel line leak) and symptoms that describe how a parameter differs from its expected value (e.g., fuel flow high).

that the component is faulted in some way. However, if we want to determine *how* the component is abnormal, we need more specific information than just normal/abnormal. Lacking that specific information, we may still be able to determine that the component is broken, but we lose specificity of the fault hypothesis. In this thesis, a form of abstraction named status abstraction is presented and demonstrated to provide graceful degradation.

Multiple Classes of Faults

Another aspect of robustness arises because not all faults can be diagnosed using the same problem solving approach or the same knowledge. Simon discusses this issue for problem solving of ill-structured problems, and talks about alternating among multiple problem spaces as necessary [57]. Davis explores this issue for diagnostic problem solving, by showing that different models of the physical system are useful for diagnosing different kinds of faults [11]. These different models are useful because they represent different types of adjacency, or different ways components can be "close," or "interact" (e.g., components that interact electrically or magnetically).

The definition of fault categories is based on the kind of knowledge available to diagnose the fault; in Davis' work, the kind of knowledge used was knowledge about adjacency. This thesis builds on the idea that fault classes should be defined based on adjacency, and adds the notion of specificity of available knowledge about the fault. That is, we describe the type of paths of interaction followed by the fault (which are based on functional or physical adjacency), and the level of detail of information about the faulted component and what is being propagated. A less detailed description describes a component as being broken (e.g., fuel pump abnormal), with propagation of abnormal status to functionally adjacent components. A more detailed hypothesis would describe how the fuel pump is abnormal (e.g., clogged) and what abnormal parameter values are propagated to adjacent components.

1.2 Contributions

This thesis presents ideas for improved robustness in problem solving, and demonstrates these ideas for diagnosis of complex physical system in operation. We summarize the ideas below, then expand on each one.

1. Graceful degradation can be viewed as reasoning at higher levels of abstraction (less detail) whenever the more detailed levels prove to be incomplete or inadequate. A form of abstraction is defined that applies this view to the problem of diagnosis. In this form of abstraction, named status abstraction, we define two levels. The lower level of abstraction corresponds to the level of detail at which most knowledge-based diagnosis systems reason.

2. At the higher abstraction level, this thesis presents a graph representation that describes the real-world physical system. The representation at the higher abstraction level is simple enough that reasoning can be done, yet not so abstract that important characteristics are missing. The thesis demonstrates an incremental, constructive approach to manipulating this graph representation that supports certain characteristics of operative diagnosis. That is, we show the suitability of the constructive approach for diagnosing fault propagation behavior over time. We also show its suitability for diagnosing systems with feedback under some circumstances.

3. We show a way to represent different semantics in the same graph representation to characterize different types of real-world fault behavior, and we show a way to compose hypotheses in a manner which corresponds to the way that these different behaviors occur in the real world.

4. We demonstrate an approach that treats these different behaviors as different fault classes, and the approach moves to other classes when previous classes fail to generate suitable hypotheses.

We now expand on each of these ideas.

1. **Graceful degradation can be viewed as reasoning at higher levels of abstraction (less detail) whenever the more detailed levels prove to be incomplete or inadequate.** A form of abstraction is defined that applies this view to the problem of diagnosis. This form of abstraction is named *status abstraction*, because the operational status of the components of the physical system is being abstracted. It is useful for graceful degradation because less specific knowledge is needed at higher levels of abstraction, although a correspondingly less specific hypothesis is generated.

This thesis explores two levels of status abstraction. At the higher level, component operational status is described as either normal or abnormal. At the lower level, a more specific description of the component's operational status is described. For example, at the lower level, a hydraulic line might be described as "clogged." At the higher abstraction level, the hydraulic line's operational status is abstracted to "abnormal." Figure 1.1 shows this relationship for this and other examples. The lower level of abstraction corresponds to the level of detail at which many current knowledge-based diagnosis systems reason.

The choice of status abstraction was inspired mainly by discussions with diagnosticians. If a major goal of the diagnostician is to select a remedial action to take in response to the fault, the information should be generated to support that selection. During the interviews of experts, they described default actions that they would take if they did not recognize the fault or if there were multiple hypotheses. This action was generally a conservative response to the fault. For example, if the pilot knew he had a fan failure, but did not know how the fan was broken, he would shut down the engine. However, if he knew it was icing, he would turn on the de-icing system. The important point to notice is that he had an action associated with fan failures that was (potentially) different from the action associated with the specific fan hypothesis. Similarly, a doctor who can identify that a patient has a bacterial infection, but cannot identify what kind of bacteria, will often prescribe a wide-spectrum antibiotic. If the doctor is able to identify the bacteria, he may prescribe an antibiotic intended for that

Decreasing
Level of
Detail

Fuel Pump Abnormal	Fuel Pump Normal	Hydraulic Line Abnormal	Fuel Heater Normal

Fuel Pump Leak   Fuel Pump Clogged   Fuel Pump Starting   Hydraulic Line Clogged   Fuel Heater On   Fuel Heater Off
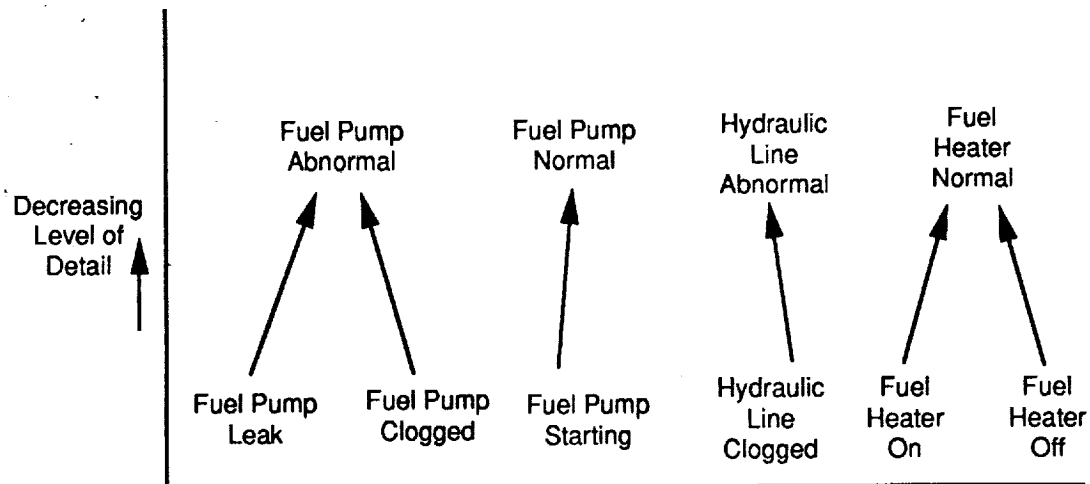
Figure 1.1: Levels of Component Status Abstraction.

particular type of bacteria.

Motivated by these and other examples, we characterize the hypotheses that the diagnosticians used when they did not recognize the fault as being one which they knew in detail. It appeared the diagnosticians were creating more general hypotheses as they had less knowledge about the fault. For example, the doctor might recognize the general characteristics of a bacterial infection, but might not have sufficiently detailed knowledge to identify the specific type of bacteria. In the aircraft domain, these more general hypotheses describe less specific information about the operational status of the components in the physical system.

This form of abstraction is designed so that diagnostic reasoning falls back to the higher level of abstraction when faults cannot be diagnosed specifically. At the higher level, hypotheses are produced that identify what component is faulty, without identifying how the component is broken. Since we abstract the operational status of the component, we named it status abstraction.

Fuel Flow Abnormal   Fuel Flow Normal

Decreasing
Level of
Detail

Fuel Flow High   Fuel Flow Low   Fuel Flow Normal

Fuel flow error is   Fuel flow error is   Fuel Flow Error is 0
358 lbs/hour   -200 lbs/hour
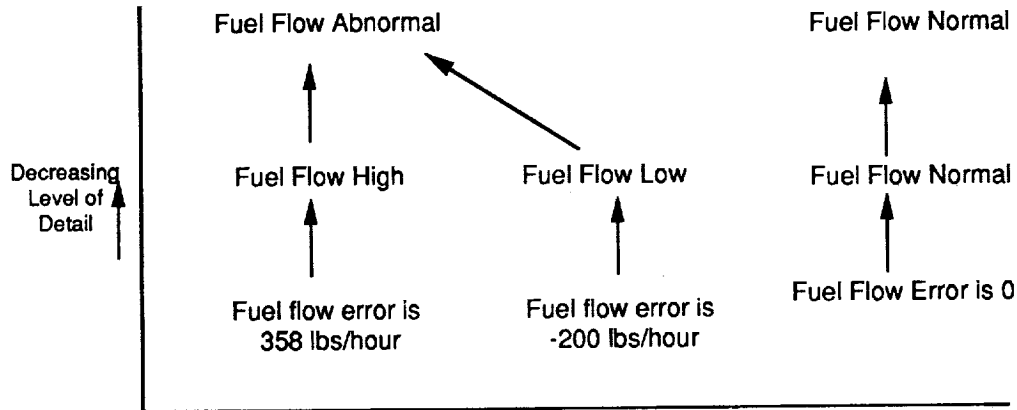
Figure 1.2: Levels of Parameter Status Abstraction.

Because the diagnostic reasoning at the higher abstraction level is designed to iden-
tify the component that is faulty, but not how it is faulty, the behavior used in the
diagnosis can be abstracted as well. We show that this abstraction is useful for graceful
degradation because less specific information about faulted system behavior is needed
to generate hypotheses. Although it is necessary at the lower level to identify how the
symptomatic sensor reading compares with its expected normal value (e.g., high or low),
it is not necessary to make this distinction at the higher level. It is only necessary to
identify that the value of the sensor is abnormal. This is also status abstraction, but it
is the parameter value status that is abstracted. Figure 1.2 illustrates the relationship.

This abstraction of behavior helps illustrate why this form of status abstraction
is useful as a means of coping with the complexity of the physical system's behavior.
Consider the example of a broken turbine blade. As mentioned earlier, we do not
know how to model many of the possible behaviors of a broken blade (or blades) in
any detail, because we do not know how the blade may bounce against other parts
of the system, nor do we know how to model the aerodynamics of broken blades in

detail. This inability to provide detailed fault models for all possible manifestations of broken turbine blades is the reason such an occurrence may appear to be a novel fault. However, by modeling the behavior of the physical system at the higher level of status abstraction, we do not have to model the behavior of the broken blade in any detail. We merely have to identify that the component is abnormal, and identify the propagation of abnormal status rather the propagation of specific parameter values.

In Draphys, the reasoning at the higher level of abstraction is a generate-and-test process, although the definition of the abstraction levels is independent of the diagnostic reasoning technique used at that level, and other techniques besides the one chosen could be used. When symptoms first appear, the generator localizes the fault in a component hierarchy, resulting in a set of candidate components that might be the source of the problem. It then constructs fault hypotheses by simulating fault propagation from each of the candidates. Each resulting hypothesis is tested to determine if it is valid; that is, if it explains all the current symptoms. Often this generate-and-test process results in multiple valid hypotheses.

The contribution here is the idea of using status abstraction as a framework for supporting graceful degradation in diagnostic problem solving.

2. At the higher status abstraction level, we present a graph representation that describes the real-world physical system. The thesis demonstrates an incremental, constructive approach to manipulate this graph representation that supports certain characteristics of operative diagnosis. The graph representation describes the physical system as a collection of components and their interconnections. The nodes in the graph represent the components, and the links represent potential paths of interactions among the components.

We present an incremental, constructive diagnostic approach to manipulating the graph representation that is suitable for diagnosing fault propagation behavior over time, and for sometimes diagnosing systems with feedback. In operative diagnosis, new symptoms often appear as time passes and the effect of a fault propagates. It is usually

undesirable to wait until all symptoms have appeared before performing a diagnosis, since critical damage can occur that might have been prevented. In this approach, we incrementally construct hypotheses by simulating fault propagation behavior. This simulation involves traversal of the graph representation to reflect the propagation path followed by the fault. We incrementally add to hypotheses created in previous time steps by continuing the simulation. When new symptoms appear, the previously created hypotheses are extended to account for the new symptoms. This is done by continuing the simulation of propagation from the point where propagation had stopped previously. An advantage to this approach is that it efficiently reasons about new symptoms when they appear, rather than starting from scratch at each time step.

Another advantage is that this approach can sometimes diagnose systems with feedback. That is, given a subsystem with feedback that has several sensors, such as the engine, not all sensors may reflect the fault's effect immediately. The sequence in which the sensors become symptomatic may help identify the source of the fault. The diagnostic approach presented in this thesis is designed to take advantage of the sequencing of symptoms. Of course, if all sensors become symptomatic simultaneously, this approach cannot distinguish where the problem began.

3. We show a way to represent different semantics in the same type of graph representation to characterize different types of real-world fault behavior, and we show a way to compose hypotheses in manner which corresponds to the way that these different behaviors occur in the real world. The semantics referred to are different types of fault propagation behavior. The types of fault propagation that we are concerned with are *functional*, which is propagation along the normal, intended paths of interaction designed into the device, and *physical*, which is fault propagation along unintended paths of interaction, but it occurs because of physical adjacency of components. One consideration is that when physical propagation occurs, functional propagation is also likely to follow, although the reverse is not very likely. Therefore, certain combinations of these two types of fault propagation behavior are more reasonable to expect than others. We can diagnose such complex

fault behavior by composing hypotheses which individually describe a single type of propagation (either functional or physical) based on heuristics about fault propagation behavior (e.g., functional propagation follows physical propagation).

Davis first presented the idea about adjacency in different models [11]; our contribution is in using adjacency to look for particular types and combinations of fault propagation behavior. This approach uses the hypotheses that were constructed for a single type of fault propagation, so the composition process is efficient and straightforward. Also, the approach can easily be extended to accommodate other heuristics, such as "physical propagation can follow functional propagation." This reasoning is at the higher level of status abstraction, so we do not need to know how the physical damage will specifically affect behavior.

4. We demonstrate an approach that treats these different fault propagation behaviors as different fault classes, and the approach moves to new classes when previous classes fail to generate suitable hypotheses. We define four single-fault classes, each of which is explored in turn when the previous ones fail to diagnose the current symptoms. These fault classes are: (1) specific, functionally propagating faults (this class is the one in which Draphys groups known faults in fault-symptom associations); (2) abstract, functional-propagation faults; (3) abstract, physical-propagation faults; and (4) abstract, hybrid-propagation faults (these hypotheses include both physical and functional propagation). The last class, describing multiple faults, is the final class that would be explored after the single fault classes fail to produce acceptable hypotheses. This final class is not yet implemented in Draphys, but most probably would be divided into subclasses describing different combinations of differently-behaving multiple faults.

The order in which these classes are examined is shown by traversing the leaves of the tree shown in Figure 1.3 from left to right. The leaves represent the fault classes, and the nodes at higher levels in the tree represent the assumptions that must be true for a fault to be a member of this fault class. The order in which these classes are explored is based on likelihood of occurrence of a fault in that class.
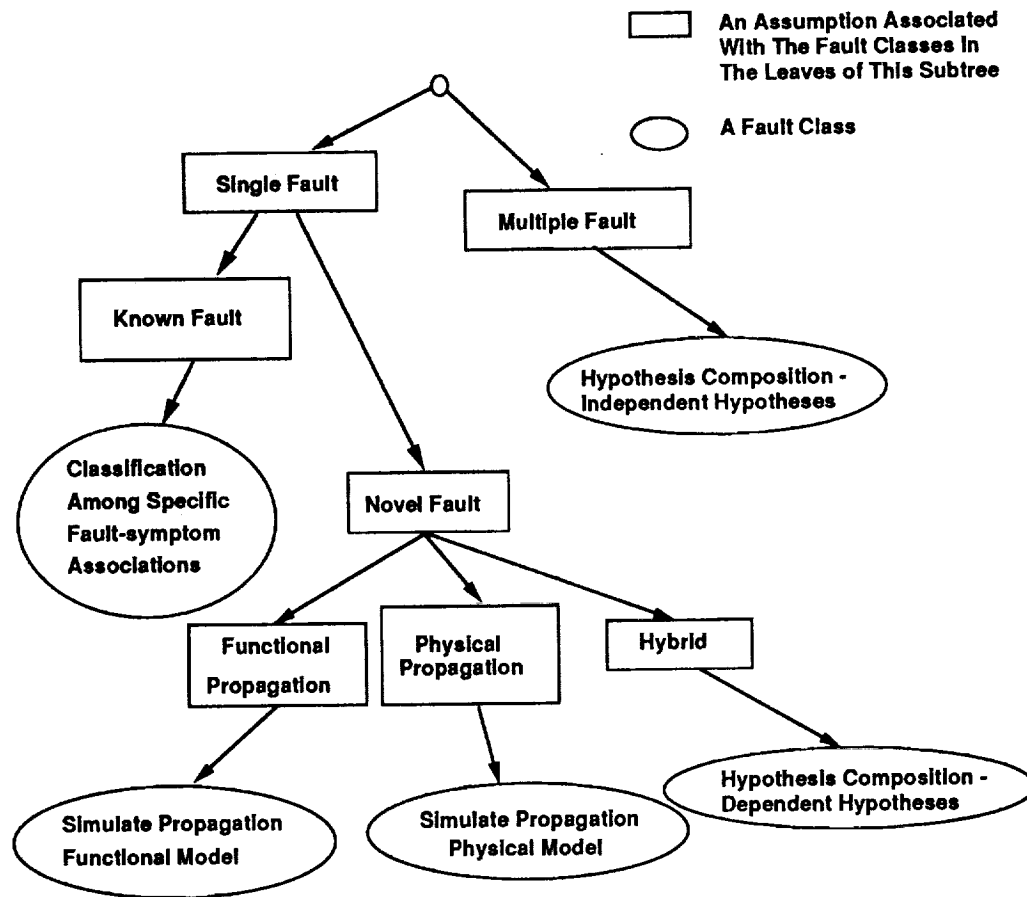
Figure 1.3: Hypothesis Space Partitioning

We consider this partitioning of the hypothesis space to be a taxonomy for diagnostic problem solving, since it associates assumptions about the fault behavior with the appropriate diagnostic problem solving techniques and physical system models. The taxonomy that is implemented in Draphys is not exhaustive, but does provide a framework for integrating new problem solving techniques into the overall diagnostic process. At present, it does not include, for example, design errors, timing errors, or intermittent faults.

A few points are worth making about these fault classes. First, they are similar in several ways to the layered fault categories defined in [11]. They are ordered by likelihood, and they are based on assumptions. Davis' work defined fault categories

based on adjacency; this thesis research adds another dimension, that of abstraction level. We claim that this contributes to increased robustness by supporting fault classes at varying levels of specificity, each of which can be explored when previous classes fail to generate hypotheses.

This notion of fault classes also contributes to efficient management of the hypothesis space. By grouping the elements of the hypothesis space into these fault classes, and reasoning about these fault classes in order of likelihood, the hypothesis space is constructed and explored in an efficient, knowledge-directed manner.

1.3 Scope

The purpose of this section is to characterize the scope of the thesis in terms of the assumptions made, the functionality of the diagnosis, and the implementation of the ideas.

1.3.1 Assumptions

In this section, we describe the assumptions made in the design and operation of the diagnostic reasoning. These assumptions include:

- We assume that we can predict normal behavior for every sensor reading available to the diagnosis system, that we only have the sensors currently available in the cockpit, and that sensor placement is fixed.

- Faults are not intermittent; that is, a component that becomes abnormal stays abnormal.

- During the diagnostic process, no action is taken by the human operator that corrects the problem while it is being diagnosed.

- Abnormality in sensor readings can be detected, and faults are not masked.

- The physical-system models are correct.
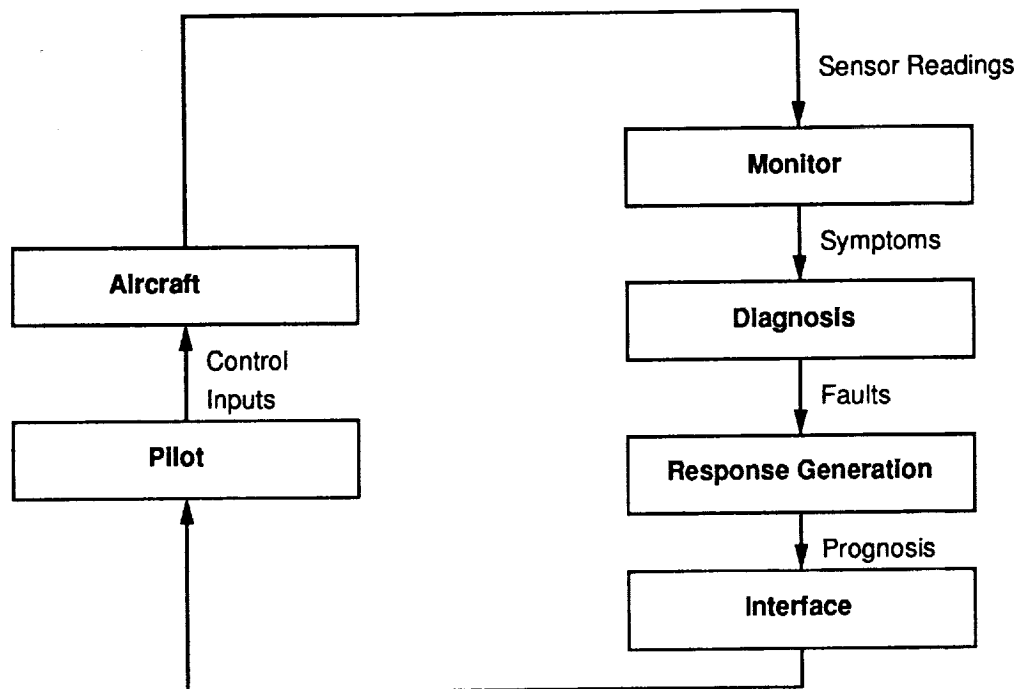
Figure 1.4: Functional Diagram of the Fault Management Process

A major design concern is to identify a general description of the functions involved when a human operator must manage faults, since diagnosis is only one of those functions. We describe each of these functions, especially the monitoring that provides inputs to the diagnosis function, to establish the context for discussion of the diagnosis approach that this thesis presents.

1.3.2 Fault Monitoring, Diagnosis, and Response Generation

It is important to understand where the diagnostic process fits into the total human/machine system, concerning management of faults. Figure 1.4 depicts a general functional diagram of the fault management process for a physical system in operation with a human operator. Besides the human operator and the physical system itself, there are four basic processes or functions that must be done: fault monitoring, fault diagnosis, response generation, and user interface.

The Fault Monitor

The physical system provides information to the fault monitor in the form of numerical sensor information. The monitor is responsible for discrepancy detection, or detecting when those sensor readings signify an abnormal situation. This detection of abnormal behavior is done by comparing the *actual* behavior of the system under consideration to the *expected* behavior of a healthy system under normal conditions. For physical systems, the expected quantitative behavior can be generated using a quantitative simulation model. Most quantitative simulation models of systems such as aircraft engines are empirical representations of the normal behavior of the physical system. Fortunately, such models are usually available for devices such as aircraft engines.

Figure 1.5 shows the fault monitoring process itself, which is implemented for the aircraft subsystems as described in [52]. The input data to the monitor are the current conditions (e.g., altitude, temperature), and control inputs (e.g., throttle setting). Based on this information, the monitor runs the device model to simulate current expected, normal behavior in the form of quantitative simulated sensor values. When the fault monitor detects a discrepancy between the actual and expected values, it transforms the quantitative sensor reading to qualitative information in the qualitative symptom.

The information contained in a qualitative symptom includes: the time the sensor reading was taken; the qualitative value of the sensor reading; the status of that value compared with the expected value (high or low); the qualitative value of the derivative; the status of the derivative value compared with the expected value; and the steadiness of the sensor reading. Both the reading and its derivative can take on qualitative values of positive (+), zero (0), or negative (-)[3]. These refer to the instantaneous value of the sensor reading or its derivative. The status of these values compared with the expected values are derived from the sign of the error. If the error is positive, the status of the

---

[3] Some values should never be negative (such as engine pressure ratio (EPR)), but even then, such a sensor could malfunction and produce an erroneous negative reading.
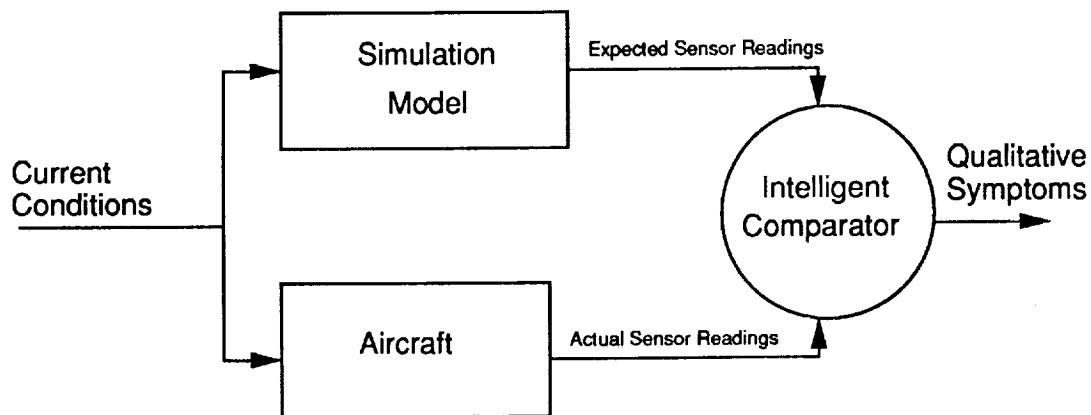
Figure 1.5: Fault Monitor Process

value is **high**. Similarly, if the error is negative, the value is **low**. The steadiness of the signal can take on the value of **stable** or **fluctuating**.

In reality, it is not so straightforward to determine when a signal differs from its expected value. Two major reasons for this are sensor noise and lack of model fidelity. Both these issues, together with approaches for handling them, are discussed in detail in [53] and [52].

Draphys assumes that all sensors can be monitored at all times. Other monitoring approaches address the problem of choosing which sensors to monitor when not all can be processed [17], [18].

The Diagnostic Process

The output of the fault monitor is the set of symptoms at the current point in time. The set of symptoms provide the input to the fault diagnostic process. The diagnostic

process occurs in stages that correspond to the way humans described their diagnostic reasoning. The first stage is diagnosis of specific, known, commonly occurring faults by associational rules. This corresponds to the stimulus-response type of reaction that human operators of physical devices are trained to have. It involves compiled knowledge about the association between symptoms and faults, so that the operator response in the presence of a known fault is rapid and efficient. However, when novel faults occur, this compiled knowledge is inadequate.

When novel situations occur, humans have been observed to revert to reasoning about underlying domain principles [45]. For physical artifacts, this often involves reasoning about a mental model of the device under consideration. We designed and implemented an approach in this second stage that uses models of structure and behavior. The models used in this stage are qualitative, in contrast to the quantitative model used by the fault monitor.

The first diagnosis stage corresponds to our lower level of status abstraction, and the second stage corresponds to our higher level of status abstraction. The motivation for the order in which abstraction levels are processed is based on observations of human diagnostic reasoning. This differs from the typical use of abstraction levels as exemplified by ABSTRIPS [48], where the processing is abstract-to-specific. We discuss the motivation for our choice, and some possible consequences and alternatives, in Chapter 4.

The output of the diagnostic process is a set of fault hypotheses. The information contained in and associated with a hypothesis includes: the *fault type*, the *cause* or *source* of the problem, the *propagation path*, and the *system status*. The *fault type* is either single or multiple fault, where multiple fault refers to multiple independent faults. The *source* is the physical component that is broken or the first one affected (e.g. in a bird ingestion, the fan is the first component affected although there is nothing physically wrong with it). The specific *cause* of the fault describes how a component is broken. The *propagation path* describes the order and manner in which components were affected. The *system status* describes the components affected by the fault and

their operational status. At present, one way that component operational status is designated is either *definitely affected* by the failure when symptom information justifies it, or *possibly affected* when there is reason to believe that the component might be affected but symptom information cannot confirm or refute it. We describe a hypothesis produced by Draphys as *valid* if it accounts for exactly the current symptoms, that is, if the component it identifies could have broken and resulted in the current symptoms. That is, the hypothesis must form a covering set [46].

Response Generation and Operator Interface

The response generation process takes the fault hypotheses as input and identifies corrective responses to the human operator. A rudimentary capability to generate responses is implemented, but more extensive work is outside of the scope of this thesis. An advanced capability for taking the output of Draphys and generating responses is being explored in [27].

Similarly, the human interface is rudimentary. There are many issues associated with providing this information to the human operator, such as display formats for presenting the information [3], and they are outside of the scope of this thesis.

1.3.3 Implementation

The diagnostic ideas presented in this thesis are implemented as part of an overall fault management system called Faultfinder [1], [38]. Faultfinder, including Draphys, is implemented on a Symbolics 3600 series computer in Symbolics Common LISP, using Flavors. The physical system implemented for the present version of Faultfinder includes two Pratt and Whitney JT8D-7 turbofan engines, oil subsystems for each engine, a fuel subsystem, and the hydraulic system for a Boeing 737 twin-engine transport aircraft.

1.3.4 Empirical Evaluation

The ideas presented in this thesis were tested by applying them to actual fault cases that occurred in civil transport aircraft accidents that resulted in loss of life and property, and failure incidents that did not result in accidents. The set of accident cases was divided into two subsets of equal size. The faults in the first set were used to develop the diagnostic concepts and to identify the capability required to diagnose each real-life situation. The second set of cases was set aside and not examined until the prototype computer program was completed. Once Draphys was implemented, the second set of test cases was reconstructed for evaluation. All the test cases were used to test Draphys, with very promising results. Out of eight accident cases, Draphys successfully diagnosed seven, and the eighth case was partially diagnosed. Details are described in Chapter 6.

1.4 Guide to the Thesis

The remainder of this thesis expands on the ideas briefly summarized in this chapter.

Chapter 2 describes the aircraft subsystems being diagnosed and presents a series of example faults that illustrate the scope of Draphys' diagnostic capability. These examples describe *what* the diagnostic reasoning does rather than *how* it is done. The details on *how* the diagnosis is done are described in later chapters.

Chapter 3 describes the reasoning about fault propagation behavior in known faults. In particular, the reasoning about sequences of symptoms over time is presented. We show that the reasoning described here is useful but not sufficient.

Chapter 4 discusses graceful degradation of the problem-solving process in the presence of novel faults as reasoning at higher levels of abstraction. The choice of abstracting the information in specific hypotheses is discussed. Again, we show that the reasoning here and the models are also useful, but not sufficient.

Chapter 5 describes the partitioning of the hypothesis space to accommodate the multiple problem solving techniques and models needed. The different assumptions

made by the diagnostic process, the different problem solvers required, and the techniques for reasoning with problem solvers and models are described.

Chapter 6 discusses an experimental evaluation of the diagnostic approach on actual fault cases. Each case and the resulting hypotheses are presented. An analysis of the resulting successes and failure show that the diagnostic approach strongly depends on the models of the physical system and the type of symptoms provided by the fault monitor. The analysis also highlights the importance of the choice of abstraction level for particular fault classes. An analytical evaluation is also presented, identifying the aspects of the approach that are particularly important to its success. As part of the analytical evaluation, we present a knowledge degradation analysis.

Chapter 7 concludes with a summary of the research contributions. Limitations of the approach are summarized as well, with implications for future research.

The reader interested in knowing what the diagnostic approach does, without necessarily knowing how it does it, should read Chapter 1, the overview, and Chapter 2, the examples. The reader interested in more technical depth on any of the major issues should read Chapters 3, 4, or 5, depending on their particular interest. Anyone with a desire to thoroughly understand the technical details and how to apply them to a specific problem should give particular attention to Chapter 6, especially the analytical evaluation of the approach.

Chapter 2

Diagnosis Examples

We describe here a series of examples that illustrates the diagnostic problem solving process proposed in this thesis. The emphasis in this chapter is not on *how* the process works but rather on *what* the process does and the variety of faults that can be diagnosed.

We begin by describing the aircraft domain; in particular, we describe a turbofan engine. We then present a series of examples, each illustrating some diagnostic capability of the thesis approach. The first example illustrates diagnosis of a known fault using associative reasoning. Later examples illustrate diagnosis of novel faults, with progressively complex propagation behavior.

2.1 The Aircraft Domain

To fully understand the examples, a short description of the application domain is appropriate. The aircraft, a two-engine civil transport, contains hundreds of complex components. We simplified the aircraft model for this research by only including two major subsystems, the propulsion and hydraulics systems. This simplification allowed us to address the complexities of fault diagnosis, including propagation and feedback, without having to model the entire aircraft. The propulsion subsystem consists of two turbofan engines and a fuel subsystem. The hydraulics system has two subsystems that correspond to the control surface in each wing and the required hydraulics support. Figure 2.1 shows the component hierarchy, where the links shown represent a subpart relationship. A total of thirty-eight components make up the model. Twenty-eight are primitive components that have no subparts, and the remainder are composite

components. It is important to note that several of the primitive components are sensors, whose purpose is to provide some type of information about system operation.

The examples shown here focus on engine faults. The type of aircraft engine chosen was a turbofan engine, one commonly used on civil transport aircraft. Figure 2.2 shows a schematic of the engine.

Its functioning is described here. The air enters the fan, a low-pressure compressor. The fan compresses the air, which flows to the high-pressure compressor. There the air is compressed further. It passes to the combustion section, which sprays fuel to mix with the highly compressed air, and ignites them. Ignition increases the velocity and temperature of the air, turning the turbines as the air flows to the exhaust section. The turbine section is divided into two stages. These two stages are connected to the fan and compressor with concentric shafts. The first turbine stage drives the compressor and the second stage drives the fan.

The engine has five sensors whose readings provide the following parameter values: N_1, N_2, Fuel flow (FF), exhaust gas temperature (EGT), and engine pressure ratio (EPR). The N_1 and N_2 sensors measure the rotational speeds of the fan and high-pressure compressor, respectively. The fan and compressor generally rotate at different speeds because they are connected to different turbine stages. Fuel flow measures the rate at which the fuel is entering the engine. EGT is the exhaust gas temperature. EPR is a ratio of the air pressure at the exhaust divided by the air pressure at the engine inlet.

2.2 Diagnosing Known Faults

Suppose a turbine blade fails because of erosion. The initial symptoms will be fluctuation in the sensors that measure characteristics of the turbine section, EPR and EGT. This may be accompanied or followed by fluctuations in N_1 and N_2. Subsequently, EGT will increase because energy is not being extracted properly by the turbines. Because the turbines are not operating properly, EPR will decrease. When this happens,
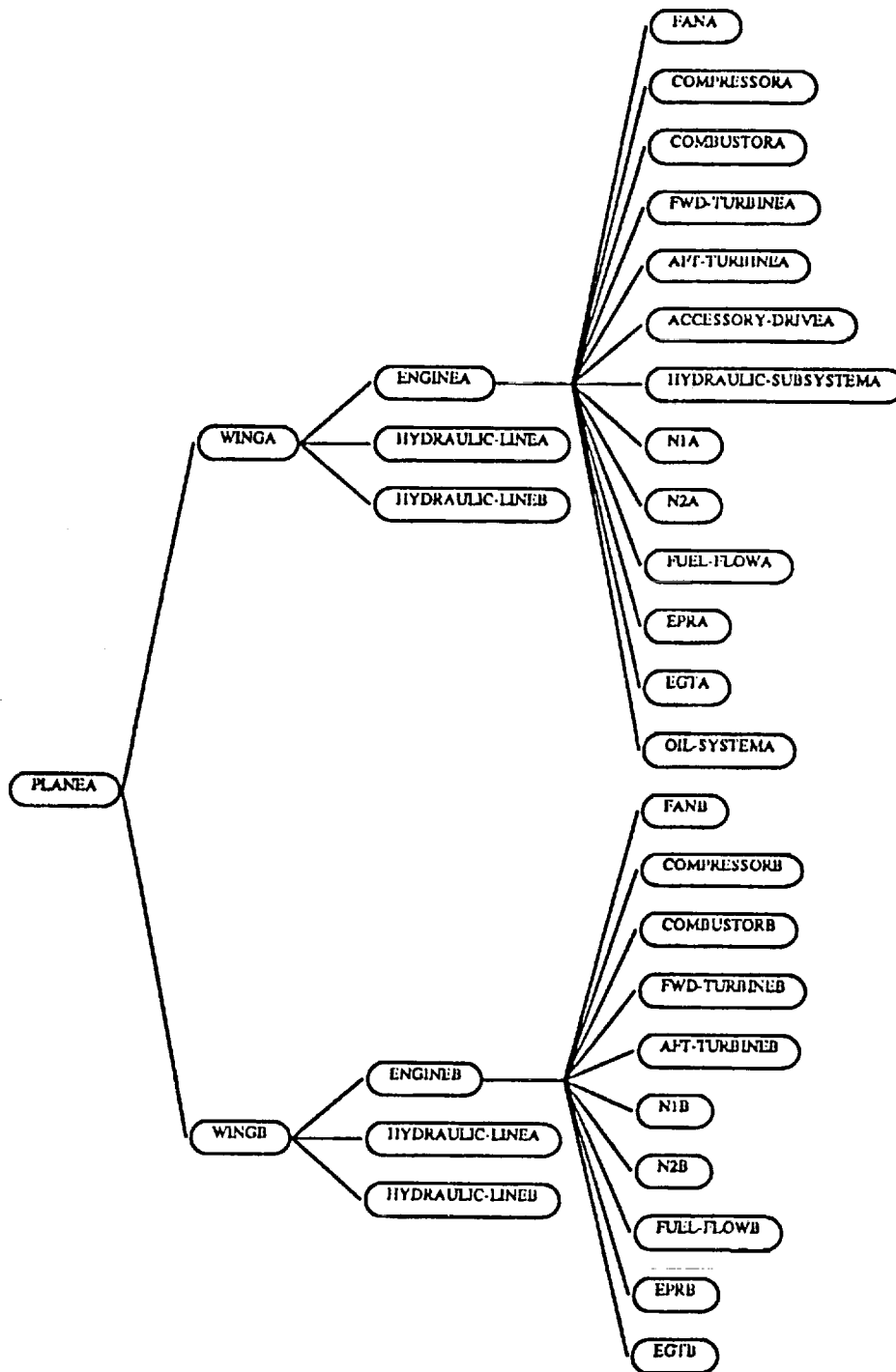
Figure 2.1: Aircraft Component Hierarchy.
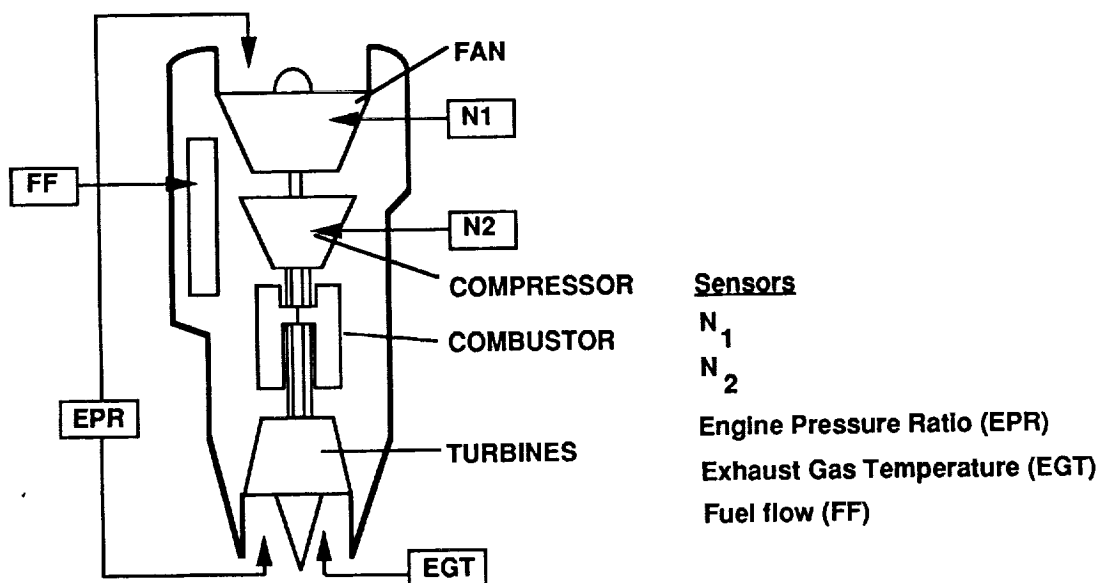
Figure 2.2: Schematic of a Turbofan Aircraft Engine.

N_1 and N_2 decrease.

Draphys diagnoses faults such as this by matching the sequence of symptoms that occur with sequences associated with known faults. The associational rules express temporal relationships among parameters as part of the conditions for rule satisfaction. The rule for a turbine blade separation is stated in English paraphrase as "EPR and EGT fluctuating is accompanied or followed by N_1 and N_2 fluctuating, EPR and EGT fluctuating is also followed by EGT increasing and EPR decreasing. EGT increasing and EPR decreasing are followed by N_1 and N_2 decreasing." Draphys will identify all rules satisfied by the current set of symptoms.

2.3 Novel Fault Diagnosis, Functional Propagation

Suppose the fault is a fan failure. We describe it as a novel fault because *how* the fan is failing is not described in detail in our associational knowledge, as our fan blade

failure was in the previous section. In such a failure, the first sensor affected would be the N_1 sensor. Since the fan would not compress air properly, the effect of that failure would propagate to the high-pressure compressor and thus to the N_2 sensor. It would then propagate to the combustor since the under-compressed air would not ignite as efficiently. Therefore, the expanding gases resulting from combustion would not turn the turbines as rapidly as they normally would. *EGT* and *EPR* would be symptomatic to reflect this. Also, since the turbines would not be extracting energy, the fan and compressor would not turn as fast since they derive some of their power from the turbines.[1] Thus the faulty response would be perpetuated.

For this fault, suppose that the first symptom that Draphys detects is in N_1. Since N_1 is an engine parameter, the second diagnosis stage, which performs diagnosis at the higher abstraction level, is able to localize the fault to the engine subsystem. Each component in the engine subsystem is then proposed as the responsible component or source of the fault.

For each proposed responsible component, Draphys generates a fault hypothesis by qualitatively simulating the fault propagation behavior. That is, it uses simulation to determine the extent of the failure's effect. For example, in one hypothesis, Draphys will propose the fan as the responsible component. Draphys reasons about a model of the engine and its interconnections to determine that the high-pressure compressor and the N_1 sensor depend on the fan for their proper operation; that is, they functionally depend on the fan. This knowledge is modeled in a graph representation that describes the physical system as a collection of components and their interconnections. The nodes in the graph represent the components, and the links represent the functional dependencies among the components. Draphys then uses the same simulation process from each remaining candidate component to construct the hypothesis corresponding to that component.

---

[1]The fan and compressor would still be turning, because the air flow into the engine provides a "windmilling" effect, even when the engine is not running. However, the aircraft must be moving through the air for this "windmilling" to occur.

Knowing these interconnections, Draphys then attempts to continue simulating the propagation of the failure to functionally dependent components. In this example, it checks whether the fault's effect in the actual system has reached the high-pressure compressor. This is done by examining the symptoms to determine if N_2 is symptomatic. If it is, then the failure is assumed to affect the high-pressure compressor and Draphys continues the process from there. If N_2 is not symptomatic, as in this example, simulated propagation halts on this path. Draphys then explores all remaining functional propagation paths.

In this example, Draphys generates a set of valid hypotheses that has two distinct possibilities for a responsible component. The first is that the fan is the source, and the second is that the N_1 sensor failed. A fault in either component could result in the current symptoms. Figure 2.3 shows the hypotheses that result from a symptom in N_1. Note that these hypotheses describe what components are affected, but not how they are affected, because the reasoning is being performed at the higher abstraction level.

Extending this example further, assume that a short time after the N_1 symptom was first detected, a symptom in N_2 is also detected. Draphys then tries to extend the propagation path of all the valid hypotheses to explain the new symptoms. These paths are extended by continuing the qualitative simulation from the end of the propagation path in the old hypotheses. For instance, in one valid hypothesis propagation stopped at the fan, because the next component on this functional propagation path was the high-pressure compressor. Since earlier there was no symptom in N_2, and because we could expect N_2 to be abnormal if the compressor were abnormal (because we assume that propagation between the compressor and the N_2 sensor to be instantaneous), Draphys assumed that the compressor was unaffected. Now that there is a symptom in N_2, Draphys updates the system status for this hypothesis and continues the simulated propagation.

The resulting hypothesis, shown in Figure 2.4, accounts for all symptoms. It is the only member of the set of old valid hypotheses that can do so. Draphys eliminates all others because simulation could not extend them to account for the symptom in N_2.

HYPOTHESIS 1 OF 2

Current Symptoms:
    N1 Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:

Propagation Type: Functional
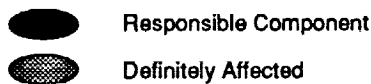
HYPOTHESIS 2 OF 2

Current Symptoms:
    N1 Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:

Propagation Type: Functional

Responsible Component

Definitely Affected
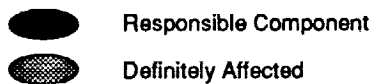
Figure 2.3: Valid Hypotheses Resulting From a Symptom in N_1.

2.4 Novel Fault Diagnosis, Physical Propagation

Suppose that the fault was fan blade separation and that the fan blade broke off and damaged a hydraulic line in the wing to which the engine was attached. Draphys detects symptoms in N_1 and in the hydraulic pressure sensor. Draphys cannot explain these symptoms by simulating functional propagation, because there is no functional relationship between them. However, a physical proximity relationship does exist. Therefore, by knowing that the fan is physically adjacent to the wing containing the hydraulic line, Draphys can identify propagation from the engine to the wing. This knowledge is contained in a graph representation similar to the representation of functional dependencies, except that the links in the graph represent potential paths of fault propagation that are due to physical proximity. Draphys uses a model of the physical propagation to simulate this type of propagation.

Another hypothesis might be that there are two independent faults, one in the

Current Symptoms:

    N2 Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:

Propagation Type: Functional

Functional Propagation

Physical Propagation

Responsible Component

Definitely Affected

Possibly Affected
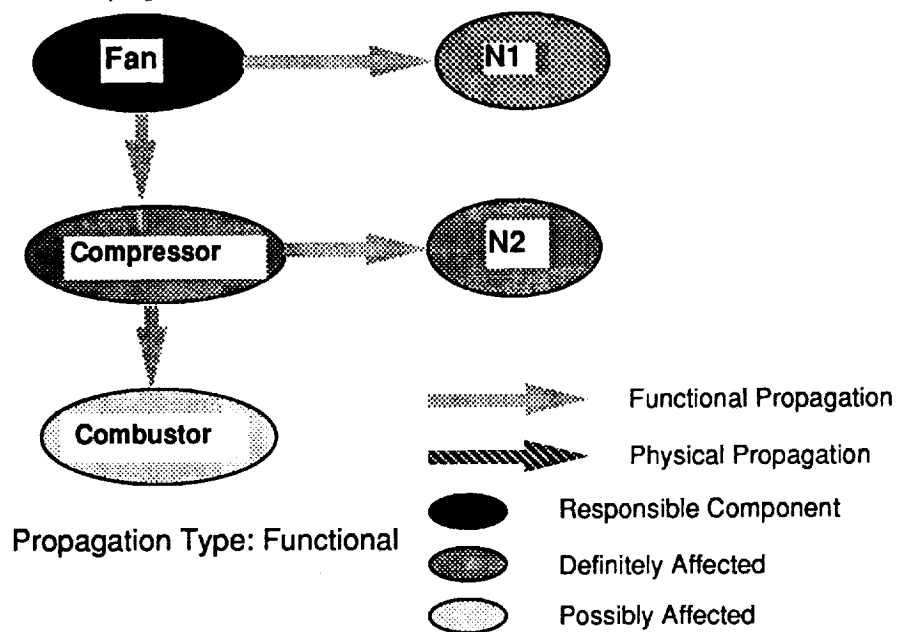
Figure 2.4: Remaining Hypothesis After a Symptom in N_2.

engine and one in the hydraulic system. Draphys does not describe the multiple fault, because it uses a heuristic that it does not produce multiple-fault hypotheses when it can produce valid single-fault hypotheses that account for the current symptoms.

2.5   Novel Fault Diagnosis, Hybrid Propagation

Once such a fan blade separation has caused damage in both the engine and in the hydraulic system, the effect of the fault will propagate functionally in both subsystems. The initial propagation was physical, but later propagation was functional. Therefore models of both physical and functional structure are required to explain the current fault behavior. Draphys handles this situation by composing the primitive hypotheses that describe the propagation within a single model.

Suppose we have a fan blade failure, and resulting symptoms in N_1, N_2, and the hydraulic pressure sensor. The hypothesis Draphys generates for this example is presented in Figure 2.5. It shows that the fan is the source of the problem, with physical propagation to the hydraulic line, and functional propagation from the fan and hydraulic to their functionally dependent components. This hypothesis is the composition of three primitive hypotheses, where each primitive hypothesis describes a single type of propagation. Each primitive hypothesis is outlined in the figure.

One primitive hypothesis represents the initial physical propagation from the fan to the hydraulic line. Another primitive hypothesis represents the functional propagation within the engine resulting from the broken fan, and the third primitive hypothesis describes the functional propagation that resulted from the hydraulic line damage.

Continuing this example even further, suppose we now see symptoms in EPR and EGT. The hypothesis shown in Figure 2.5 is extended by continuing simulation from the point(s) in the propagation path where it stopped, resulting in the hypothesis shown in Figure 2.6. Once propagation to the turbine is confirmed, the status of the combustor is updated to "definitely affected," since the reasoning assumes that the fault would not affect the turbine without affecting the combustor.

HYPOTHESIS 1 OF 1

Current Symptoms:
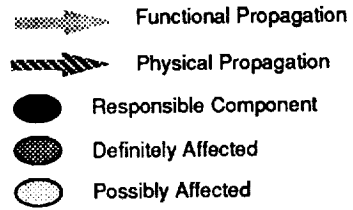
    N1 Abnormal

    N2 Abnormal

    Hydraulic Pressure Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:
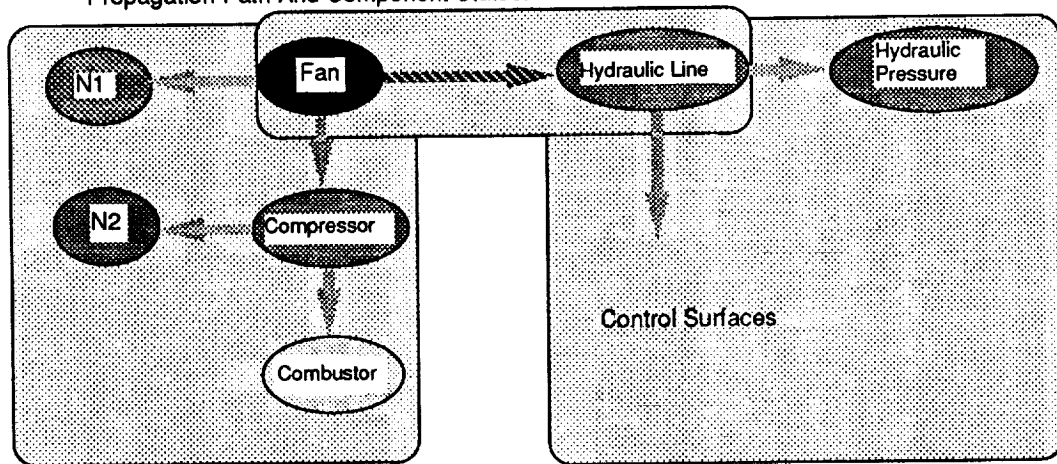
Propagation Type: Hybrid

Figure 2.5: Composed Hypothesis With Physical and Functional Propagation.

HYPOTHESIS 1 OF 1

Current Symptoms:

    N1 Abnormal

    N2 Abnormal

    Hydraulic Pressure Abnormal

    EPR Abnormal

    EGT Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:
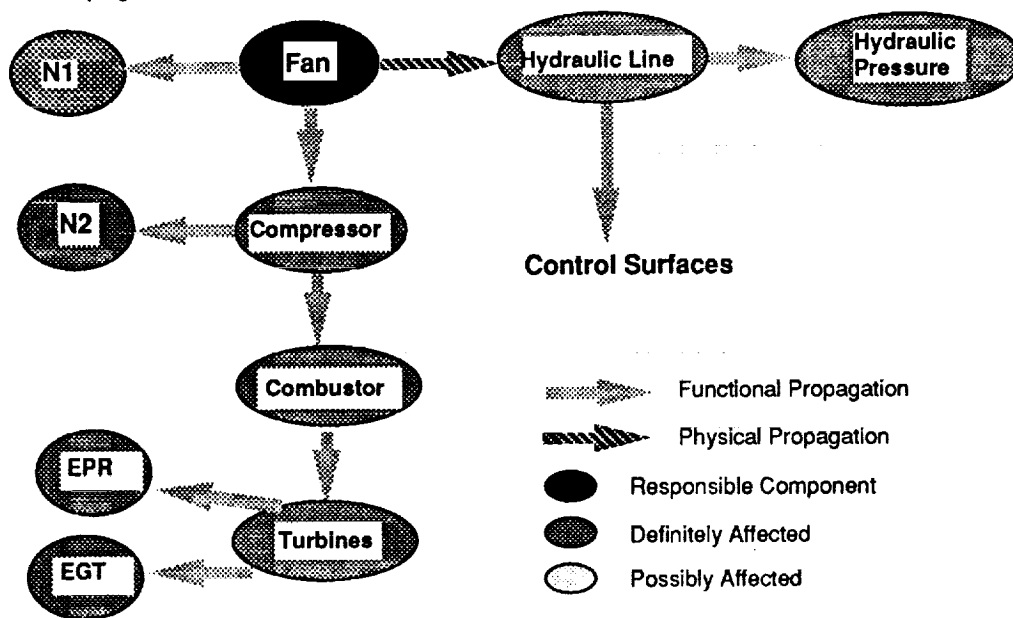
Propagation Type: Hybrid

Figure 2.6: Hypothesis After Symptoms in EPR and EGT.

2.6 Summary

In this chapter, we showed a set of increasingly complex fault situations. The initial example was that of a known fault, or one for which we had specific knowledge about how the component was faulty and the corresponding symptoms. The later examples were complex in the following ways: the fault propagation behavior increased the set of symptoms (and the corresponding consequences) as time progresses, and the fault behaved in such a way that a single model was not sufficient to diagnose it.

Chapter 3

Diagnosing Known Faults Via Associational Knowledge

Fault propagation results in changes to symptoms as time and the effects of the fault progress. In this chapter, we discuss reasoning about the patterns of symptoms over time for known, commonly-occurring faults. We define known faults as faults about which we have specific knowledge; that is, we have knowledge about how the faulty component was broken (e.g., fuel pump clogged), and how the affected system parameters differ from their expected values (e.g., fuel flow is high).

We should emphasize two points about this chapter and its contents. First, we include the information in this chapter for completeness and to help clarify the relationship between known faults and novel faults, but techniques for diagnosing known faults is not the primary focus of this research. Second, we used associational knowledge in the diagnostic reasoning, but other techniques could have been used, as described later in the chapter.

3.1 Associational Knowledge

The approach implemented in Draphys for diagnosing known faults was to embed reasoning with temporal predicates into a rule-based system. The rules express the association between a fault and a set of symptoms, and the symptoms' changes over time.

The fault-symptom associations for the aircraft domain were obtained by interviewing domain experts (pilots and engine designers) and by examining actual fault cases. The experts often described symptoms as a sequence of events over time. For example, a foreign object ingestion was described by one pilot in the following way: "First

performance values will fluctuate, then EGT and EPR will decrease." The sequence of symptoms often helped distinguish among different fault hypotheses that had the same initial symptoms.

To provide the capability of reasoning about sequences of symptoms over time, a rule-based system was extended to permit temporal reasoning functions within the rules in a knowledge base. The temporal functions permitted as part of the rules are summarized in Table 3.1. These functions are the temporal functions developed by James Allen [4].

As an example, the rule for a possible foreign object ingestion looks like:

```
((foreign-object)
    (and
            (equal  EPR       fluctuating
                    EGT       fluctuating)
            (equal  EPR       fluctuating
                    Fuelflow  fluctuating)
            (meets  EPR       fluctuating
                    EPR       decreasing)
            (meets  EGT       fluctuating
                    EGT       decreasing)))
```

To paraphrase, this rule states that the hypothesis of foreign object is a possibility when EPR, EGT, and Fuelflow are fluctuating simultaneously, EPR fluctuating is followed immediately by EPR decreasing, and EGT fluctuating is followed immediately by EGT decreasing. When this rule is fired, the intervals during which the parameters have the qualitative values are tagged with the times during which they occur, and the relationships between intervals are identified using those time tags.

To illustrate how the diagnosis is tailored to reason about dynamic behavior of the

Table 3.1: Temporal Functions

Temporal Functions

Temporal Functions	Explanation of Function
Starts X Y	Intervals X and Y begin at the same time but X ends before Y. XXX YYYYY
Finishes X Y	X and Y end at the same time but X begins after Y. XXXX YYYYYY
Before X Y	X is completed before Y begins. XXX YYY
Overlaps X Y	X begins before Y begins ends after Y starts, but before Y ends. XXXX YYYYY
Meets X Y	X ends at the same time that Y begins. XXXYYY
During X Y	The time interval for X is entirely contained within Y's time interval. XXX YYYYY
Equal X Y	X and Y occur during the same time interval. XXX YYY

physical system, suppose we are given the following symptoms from the fault monitor:

Signal	Value	Time
(EGT	fluctuating	5)
(EPR	fluctuating	6)
(Fuelflow	fluctuating	6)
(EGT	decreasing	12)
(EPR	decreasing	14)

The symptoms are presented here in a compressed form, so that only the abnormal aspects of the sensed values are shown. These symptoms represent the qualitative value of the sensor error at the beginning of the time interval during which the value is valid. Until a new qualitative value is generated by the fault monitor, this qualitative value is assumed to hold. In the example presented above, EGT began fluctuating at time 5 and continued to have that value until time 12. The fault monitor sends qualitative symptoms to the diagnosis process at the beginning of a time interval, and only sends another value for that sensor when the qualitative value changes. Thus the interval during which a symptom's qualitative value holds is the interval used for Allen's interval-based temporal logic. These symptoms satisfy the rule for the foreign object ingestion described earlier.

Allen's temporal predicates were chosen because they are based on an interval-based temporal logic. An interval-based representation was well-suited for representing the sequences of symptoms, because the qualitative values of sensor data (e.g., EPR high) generally hold over a time interval. Since we want to reason about the sequences of qualitative values of the sensors, and not just the set of symptoms, we reason about them based on the time intervals in which they hold.

Using dynamic information in this stage reflects the diagnostic reasoning described by human experts in identifying known faults because it helps to distinguish among

faults which may have the same initial symptoms but different subsequent behavior. In this application domain, there are several examples of such faults. Two such faults are icing and foreign object ingestion, since both have initial symptoms of engine performance values (EPR and EGT) fluctuating.

3.2 Difficulties With the Specific Associational Reasoning

The reasoning as described provides diagnostic capability to a degree, but it has some difficulties. The difficulties essentially fall into two problem areas. The first problem area is that the choice of a rule-based representation leads to some difficulties in reasoning about the fault propagation. The second area, which is independent of the representation chosen, is the difficulty in getting the knowledge about faults and their propagation behavior at this level of detail.

3.2.1 Limitations of The Rule-Based Representation

The temporal functions appear to be adequate for describing the types of temporal relationship that occur, but the rule-based representation of fault behavior that we use leads to some difficulty in reasoning about fault propagation. The major problem arises when the rules are applied before the entire sequence of symptoms has occurred. For example, if symptoms are detected and the diagnosis process is triggered while the performance values are fluctuating, it may not be possible to distinguish yet between icing and foreign object ingestion, since they both have the same initial symptoms. Therefore, both hypotheses must be maintained. By using a rule-based representation as described and implemented, the entire sequence of symptoms must take place before the rule is satisfied. However, it might be important to identify those fault hypotheses whose initial temporal sequence is satisfied, even if subsequent symptoms have not yet occurred.

Three approaches could be used to reason about fault propagation in a rule-based representation: (1) have a separate rule for each possible propagation sequence (or

extent of propagation) that could occur; (2) have each rule represent a separate propagation step; or (3) change the inference strategy to deal with partial rule satisfaction.

Having a separate rule for each possible propagation sequence would require many rules for each fault, where each rule would represent a sequence for a different extent of propagation. Using this approach, the appropriate rule would be fired for a particular instance of that fault at that point in time, corresponding to the propagation behavior exhibited.

Each rule could represent a separate propagation step or a temporal change in the sequence of symptoms for a fault. However, to reason about temporal relationship among propagation steps for each fault, there would either have to be additional rules for reasoning about the propagation-step rules, or a means for the rules to communicate with each other about the possible interrelationships. As discussed in [14], rules are best used when they do not need to communicate with each other and they represent independent pieces of knowledge.

Another approach would be to change the inference strategy to allow rules to fire when some symptoms have occurred that match at least one of the temporal predicates in the antecedent, but other fault behavior described in that antecedent may not yet have occurred. This is certainly a feasible approach, but the motivation for making such a change would be to support reasoning about the behavior of the device as the effect of the fault propagates. In essence, then, such a change would be made to support simulation of that behavior. Although the augmented rule-based representation seemed reasonable at first, we now know some of the representational difficulties that are not completely overcome by using the temporal predicates.

These approaches for fixing the rule-based representation, while feasible, represent fixes that are required because of the choice of representation. Another choice of representation, such as a graph-based representation, may be more suited to supporting the simulation of fault propagation behavior. The nodes would represent the qualitative symptoms and the links would represent the temporal relationships. Thus the

graph would represent a state-transition network, where the states are the qualitative symptomatic sensor readings. Associated with the states might be the corresponding components affected, so system status could be easily determined. Such a representation would be similar to the representation used by Pan [39], and augmented with the temporal reasoning predicates defined by Allen.

The representation issue has potential consequences of awkwardness in characterizing the knowledge and in the reasoning. However, even considering the representation issues resolved, another problem area exists which is independent of the representation: the difficulty of getting the specific knowledge about faults and their behavior to put into the representation.

3.2.2 Getting the Fault Knowledge

In most domains, getting complete, detailed knowledge of faults and their behavior is highly unlikely. In the domain of physical systems such as the one implemented in Draphys, it is particularly hard to get detailed knowledge of fault behavior, for the reasons discussed below.

A key to this problem is the term *detailed* when referring to fault knowledge. In this thesis, the most detailed behavior we diagnose is the qualitative description of how sensor readings differ from their expected value. In this domain, the more detail needed, the less fault coverage that one is likely to get. There are two major reasons that it is difficult to get detailed knowledge of fault behavior for such physical systems: first, it can be difficult to predict all the different manifestations that a particular fault can exhibit; and second, diagnosis at this level of detail had not been attempted before, so sources of fault knowledge are difficult to find.

Depending on the severity of a fault, the resulting behavior can appear quite different in different occurrences. Take, for example, a foreign object ingestion. Usually this occurs when birds are ingested into the engine. Depending on the number of birds and the current conditions under which the engine is operating, the resulting symptoms

could differ both quantitatively and qualitatively. Representing all the different possible behaviors is not possible, although this situation occurs often enough to warrant representing as many as possible. The different manifestations for this and many other faults cannot be completely predicted, because we have not seen them all to get the empirical knowledge, and because we do not know how to model physical systems well enough to completely model faulted physical systems (discussed in section 3.3.1).

The second reason it is difficult to get the knowledge is because diagnosis at this level of detail has not been attempted before, mainly because the type of monitoring presented in this thesis produces symptoms at a level of detail that was not previously available. Before this, the monitoring of sensors generally used fixed thresholds, usually extreme upper and lower bounds that represent the normal operating range of the physical system. Whenever the sensor readings varied beyond those limits, an alert was raised. The type of monitoring that provides input to Draphys does not use fixed thresholds. Rather, the monitor calculates the expected values for each sensors under the current operating conditions, and raises an alert when the actual and calculated sensor readings differ by more than some expected sensor noise level. This monitor can detect abnormal sensor readings before they exceed the fixed threshold; therefore, the symptoms are detected more quickly.

Because the monitor is more sensitive to sensor deviations from normal, the fault-symptom associations used in the fixed-threshold approach do not usually apply. Also, the human experts (in this case, pilots) do not monitor symptoms at the level of detail of the monitor. The pilots monitor the aircraft systems based on sensor information presented on the gauges and dials available in the cockpit, in a manner very similar to operators of other complex process control systems. Therefore, the monitoring of the sensors is dependent on the resolution of those gauges and dials. Because the resolution of these devices is not great, the pilots do not depend on instantaneous detection of aberrant measurements, or transient measurements. Instead, they reason about behavior over time periods that may be short but are not instantaneous. They seem to ignore transients and only reason about steady state behavior over a period

of time, albeit a short period. Because the pilots do not have information at the same resolution as the monitor, and because they approximate transients (which they may not be able to detect very well because of the information's resolution), the pilots were not always very good sources of fault-symptoms associations.

In contrast, the symptoms generated by the fault monitor for input to Draphys are instantaneous. That is, the monitor identifies discrepancies as soon (within some small allowance for sensor noise) as they occur. Humans can do the same, but it takes much time, attention, and mental computation. For these reasons, and because humans are inherently poor monitors [40], pilots do not monitor the aircraft sensor readings in the same manner that the fault monitor does. Thus Draphys has more detailed information about the fault's symptoms than the pilots can provide.

Since the pilots do not have the same kind of symptoms provided to Draphys, they do not perform diagnosis in a way that can use our more detailed symptoms. Indeed, pilots rarely perform diagnosis at all. Their behavior can best be described as a stimulus-response behavior; that is, when they see certain symptoms, they take certain actions. These represent mental shortcuts that skip the diagnostic process. These shortcuts can have serious consequences [51], if the shortcuts do not include all the information necessary to correctly and completely respond in the current fault situation. This also means that we cannot acquire the fault-symptom associations from pilots, because they do not have the knowledge we need.

Because of the reasons mentioned above, it is difficult to acquire knowledge empirically or analytically. The point is not that we cannot get *any* knowledge, but rather that the knowledge is sure to be incomplete.

3.2.3 Other Issues

Several other issues arise, which include the causal relationships among malfunctions, temporal duration of symptoms, and reasoning with uncertainty.

Causal Relationships

Another consideration that affects the issue of representation is causal relationships among malfunctions and the resulting symptom coverage issue. The issue of symptom coverage is the question of whether a fault hypothesis should be considered valid if it only covers a subset of the current symptoms. This question arises because a malfunction of a particular component (e.g., compressor stall) may cause other components to malfunction. Stating this a different way, some malfunctions may cause other malfunctions to occur simultaneously. This can occur in a compressor stall, which may cause a flameout in the combustion section. It is important to recognize that these may not be mutually exclusive malfunctions, but can be related causally (although each could occur independently without causing the other). Therefore, it can be important to identify when a hypothesis does not account for all symptoms, and when multiple hypotheses could be related causally.

Although this knowledge currently is not included in Draphys, it might be useful to include it in the future to help direct the reasoning process.

Temporal Duration of Symptoms

Draphys currently does not represent the temporal duration of symptoms. Given two faults with the same initial symptoms, the length of time that the symptoms last may allow elimination of one of the hypotheses. However, inclusion of the possible range of symptoms' duration in our fault knowledge for all possible manifestations of a fault would be difficult in this domain, for several reasons. First, the knowledge of the symptom duration would have to describe a range of time. This is because the duration of a particular symptom will probably vary, depending on factors such as the severity of the fault. Moreover, this type of duration information may be very difficult to obtain, as it is in this domain.

False Positives

In the design of a diagnostic approach, a choice should be considered whether to prefer false-positive hypotheses (identifying a fault that is not the current problem) by being less stringent about the requirements on a hypothesis, or whether to prefer false-negative hypotheses by being more stringent. Draphys was designed to prefer false-positive hypotheses to false-negative hypotheses. The motivation behind this was that, in general, it is better to be conservative, wasting some time thinking something might be faulted that was not, than to have an accident thinking something was working that was not. Therefore, the known-fault associations were designed to have minimally stringent conditions.

3.3   Related Work

We describe related work in association-based approaches, temporal reasoning, and reasoning about fault propagation behavior.

3.3.1   Association-Based Approaches

The output of most diagnosis systems is a set of fault hypotheses that correspond to the appropriate set of symptoms. These symptoms are either provided as input or determined as part of the diagnostic process. Diagnostic approaches based on specific fault-symptom associations have been extensively explored. Here we consider two areas relevant to operative diagnosis where progress has been made. These areas include the source of the knowledge, and reasoning about propagation behavior of the fault or disease over time.

Sources of Associational Knowledge

Associational rules are often used to represent the correspondence between symptoms and faults. These associational rules are usually generated in one of two ways. The first, used in expert systems, is to acquire experiential rules from a diagnostic expert in

the domain of interest.[1] The second is the fault dictionary approach, which stores fault-symptom associations, used in conventional hardware diagnostics. With each approach the cost of *generation* and *retrieval* of the compiled knowledge must be considered.

Experiential rules acquired from experts may provide a very powerful means of doing diagnostic problem solving. Experts use heuristics based on much experience, and these heuristics often work very well. However, experiential rules have two major difficulties: their acquisition and their completeness. Acquisition is difficult because the experts often cannot articulate their problem solving techniques. Even if this problem is overcome, there is no way to guarantee that the set of associational rules is complete for a particular domain. This lack of completeness is a major problem for an automated diagnostic system that only reasons with specific associational knowledge, and it does not reflect the graceful degradation of the human expert's problem solving capabilities.

The fault dictionary approach commonly used in hardware troubleshooting [8] uses a model of the physical system to simulate the fault's behavior.[2] The resulting symptoms are stored in fault-symptom associations that are retrieved later from this dictionary of faults. This approach is used in electronic circuits, because this domain can be modeled readily and the class of faults considered is easy to simulate in the model. This approach requires a realistic model capable of simulating faulted system behavior together with knowledge of the type of faults to be simulated.

While this may be satisfactory for a specific fault type in a domain such as electronic circuits, it is difficult to apply this approach in other domains. For example, in devices such as a turbofan jet engine, modeling of **normal** behavior is well understood, but modeling of **abnormal** behavior is not. Simulation models for these devices are based on certain assumptions, such as steady state behavior. By their very nature, faults in such devices are often transient, although they may eventually achieve some steady state behavior. Because of this, and because of lack of knowledge of the physics of

---

[1] Experts' rules may represent problem solving heuristics as well as fault-symptom associations, but we only consider the associations here.

[2] See [11] for a discussion of traditional approaches to hardware troubleshooting.

complex devices, modeling of faulted system behavior to acquire symptomatic behavior, especially behavior over time, is not feasible because of limitations in current modeling technology [5], [44].

The fault dictionary approach is useful in the electronic circuit domain because it is easy to model the behavior of the fault that is the most common (a "stuck-at" fault). However, other types of faults can occur that were not one of the types modeled, such as a bridge fault. If a fault occurs which was not included in the fault dictionary, the fault dictionary approach cannot diagnose it.

Each of the two approaches for acquiring associational knowledge has advantages. The fault dictionary approach may be complete for a particular class of faults, assuming the model used was complete. However, the experiential rules may represent fault-symptom associations for fault behavior that we do not know how to model. Moreover, the expert may provide heuristics that make the problem solving process more efficient.

Even if the fault dictionary were complete, and we assume every type of fault is represented, one must consider the cost of retrieval at diagnosis time. The cost of generating a fault dictionary may be quite high, but the compiled fault dictionary will at least be more efficient than constructing it "on the fly." The cost in time and storage requirements of searching for the appropriate association becomes unacceptable, especially for operative diagnosis, and particularly if one includes multiple faults.

3.3.2 Temporal Reasoning

As discussed in [56], there are several ontologies for representing temporal relationships. We chose Allen's interval-based formalism [4], because it was a natural means of expressing the relationships among qualitative symptoms, where the intervals represent the time interval during which a sensor maintains the qualitative value. We found this to be adequate for expressing the associational knowledge.

3.3.3 Diagnosing Fault Propagation

In operative diagnosis, the effect of a failed component will propagate through the device. There are two categories of propagation. One is the case where the fault causes the component to produce output that is outside its normal operating range, thereby providing input to another component outside its normal range of input values. This may cause that component to malfunction.

The other category is the case where the faulty component produces output that is abnormal for the current operating conditions, but is not outside the normal operating range of the affected component. The component is operating in a degraded mode, compared with its expected operation under the current conditions. As we discussed earlier, it can be important to identify when the component is operating abnormally for the total functioning of the system, even though the component is working correctly for its inputs.

For the goal of providing corrective actions in operative diagnosis, the distinction between these two categories may be important. For example, if affected components are malfunctioning but are not physically broken, they can continue operating. However, if the component is physically broken (or will be shortly), the action taken in response may be quite different.

Pan [39] addressed the problem of dependent failures, where a fault in one component causes a fault in another. In particular, he addresses the situation where a fault in one component generates output that takes on values outside of the normal range of values. Therefore, the input to the downstream component is outside of the normal input range. He explicitly represented knowledge of how individual components could fail, and used a state-transition network to represent the causal relationships that result in dependent failures.

Medical diagnosis must deal with the issue of propagation for much the same reason we do in diagnosis of artifacts: diagnosis of a physical system in operation. The propagation behavior of diseases is often represented in a causal network that includes

the different disease manifestations [58] [43] [42].

3.4 Limitations

The technology for representing specific associational knowledge is available, and could be applied to operative diagnosis. However, no matter what approach is pursued for acquiring and representing associational knowledge, its completeness cannot be guaranteed. As these approaches are applied to increasingly complex physical systems, this lack of completeness becomes a more serious problem. In an association-based diagnostic approach, we consider that we have encountered a novel fault when the current symptoms do not correspond to any of the associational knowledge. In practice, it is very important to diagnose faults to support the continued, safe operation of the physical system. An important issue to be addressed, then, is what diagnosis information should we produce when the symptoms encountered do not correspond to faults in our knowledge base.

3.5 Summary

In this chapter, we discussed the need to represent the dynamic behavior of faults. We approached this by acquiring knowledge from human experts, who in our domain are pilots. We represented the reasoning that they described in production rules, and implemented the capability of using temporal predicates in the conditional part of the rules.

We learned several lessons from this. First, the reasoning they described used the temporal sequence in which symptoms occurred to disambiguate hypotheses. Next, it was very difficult to acquire the knowledge. Because the knowledge is difficult to acquire

Techniques are available for representing associational knowledge about known faults. The rule-based representation is probably not optimal. Other approaches might improve upon the rule-based representation, but independent of the representation used,

an important issue (and a primary focus of this research) is diagnosis of faults when the symptoms do not match the specific associational knowledge. In the next chapter, we describe our approach to dealing with this issue.

Chapter 4

Diagnosis of Novel Faults Via Abstraction

Diagnosis of novel faults is generally considered to be a major capability of a diagnostic system that can degrade gracefully. But what exactly does it does it mean to degrade gracefully? In hardware, graceful degradation usually means the ability to continue operating reasonably well even when portions of the hardware system are broken. One would not expect the performance of the system to be as high as an unbroken system, but at least to be adequate.

In humans, especially experts, graceful degradation has an additional meaning. It also refers to the ability to solve problems unlike those seen before and to know the boundaries of one's knowledge. Problem solving behavior changes when these boundaries are reached. Indeed, such a capability is considered to be part of expertise. If an expert cannot immediately solve a problem, he does not just give up. He tries other problem-solving approaches and calls on other types of knowledge than those initially used.

Such robust problem-solving behavior requires that the human know when the current problem solving approach is not succeeding, what other problem solving techniques to use, when he should use the other techniques, and what knowledge is required to use each technique. In this chapter, we present an approach to robust problem solving that uses abstraction as a technique for dealing with novel faults. In this approach, we reason at a higher level of abstraction about the device's operational status. In doing so, we produce less specific information about the device when novel faults occur. That is, the specificity of the diagnosis is what degrades.

4.1 Diagnosis of Novel Faults

What does it mean for a fault to be novel? We define a fault as novel if we do not have specific knowledge relating the current symptoms to a fault. However, even though we do not have specific knowledge about the fault, we may still be able to provide less specific diagnostic knowledge. For example, we may be able to say *what* is broken without saying *how* it is broken. That is, we choose to view diagnosis of novel faults as an issue of specificity. The basic idea is that graceful degradation is not achieved by simply exploring whether something is known about faults, but at what level of detail it is known. Knowledge at different levels of specificity can provide different fault coverage, but increasing fault coverage is achieved at the cost of degrading specificity. For historical reasons, we shall continue to use the term "novel fault" to mean a fault which the diagnostic system cannot identify specifically. If the diagnostic system cannot identify exactly what the fault is by using specific knowledge about the physical system, it can still generate useful diagnostic information, even if it is less detailed than desired. We consider diagnostic information to be useful if providing it to the human operator helps him continue safe operation of the physical system, either by a more complete or correct response, or by aiding him in better understanding the status of the faulted system. An explanation of what motivated the choice of abstraction, and why the less specific information can be useful, follows.

Before presenting the approach, it is useful to explore what information we should abstract and why. If the goal of the diagnostician is to select a remedial action to take in response to the fault, the information should be generated to support that selection. During the interviews of experts, they described default actions that they would take if they did not recognize the fault or if there were multiple possible hypotheses. This action was generally a conservative response to the fault. For example, if the pilot knew he had a fan failure, but did not know how the fan was broken, he would shut down the engine. However, if he knew it were an eroded fan blade, he might reduce power on that engine. The point is that he had an action associated with fan failures that was

(potentially) different from the action associated with the specific fan hypothesis.

Motivated by this and other examples, it seemed useful to form general categories of faults with associated default actions, and to examine the relationship of the resulting hypotheses with the known-fault hypotheses. In the aircraft domain, these categories are formed according to the components in the physical system, as exemplified above. When novel faults occur, diagnostic reasoning takes place at a higher level of abstraction. Hypotheses are produced that identify what component is faulty, without identifying how the component is broken. The relationship between the so-called known faults and the general categories is a specificity relationship. That is, the operational status of the component is abstracted to "normal" or "abnormal," so we named this abstraction *status abstraction.*

Since we designed the diagnostic reasoning to identify the component that is faulty, thus reasoning at the higher level of status abstraction, we can abstract the symptoms, also. Although it is necessary at the lower level to identify how the symptomatic sensor compares with its expected value (e.g., high or low), it is not necessary to make this distinction at the higher level. It is only necessary to identify that the value of the sensor is "abnormal" compared with the expected value. This is also status abstraction, but it is the status of the parameter value that is abstracted. Figure 4.1 illustrates the relationship between the fault hypotheses and the corresponding symptoms at both abstraction levels currently used. We discuss this abstraction in more detail in section 4.3.2.

Another motivation for the approach taken is that it is useful to the human operator to identify the faulty component, even if we cannot provide a large amount of detail about how it failed. One reason is that humans have sensors that cannot easily or economically be duplicated. They hear things, see things, and use their intuition. Therefore, the human may have additional information that can help him select a corrective action.

The pilots also described reasoning in a structural hierarchy to perform localization
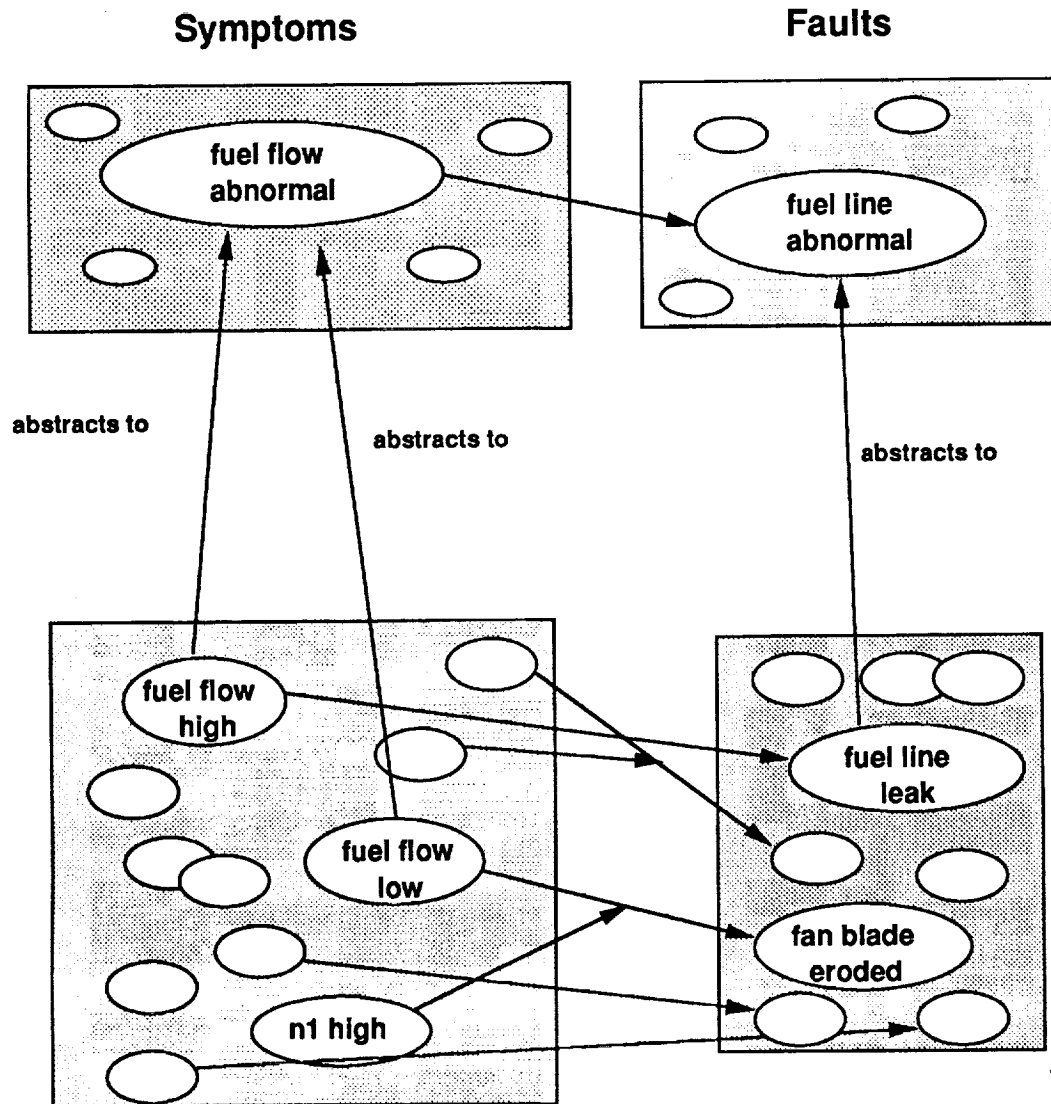
Figure 4.1: Levels of Status Abstraction.

of the fault. Having this localization information seemed to help in narrowing the choices of actions to take. Providing the localization information may prevent errors, such as shutting down the wrong engine (as occurred in the recent British Midlands 737 crash). We use structural abstraction in Draphys for localization.

4.2 Diagnostic Reasoning at the Higher Status Abstraction Level

The reasoning at the higher level of status abstraction is a generate-and-test process. When symptoms first appear, the generator localizes the fault in a component hierarchy, resulting in a set of candidate components that might be the source of the problem. It then constructs fault hypotheses by simulating fault propagation from each of the candidates. Each resulting hypothesis is then tested to determine if it is valid; that is, if a fault in the hypothesized responsible component could cause all the current symptoms. Often, this generate-and-test process results in multiple valid hypotheses. If new symptoms arrive as time progresses, the generator tries to incrementally update the old hypotheses to determine whether they account for the new symptoms. If they do, they are retained. Otherwise, they are pruned.

To see how Draphys diagnoses failures, recall the example in Chapter 2 of a fan failure. We will step through the example again, elaborating each problem solving step.

4.2.1 Fault Localization

The first symptom detected is in N_1 only. Since N_1 is an engine parameter, Draphys localizes the fault to the engine subsystem.

Localization is the process of reducing the number of candidate faulty components by refinement within a structural hierarchy of a design description. This structural hierarchy describes a component hierarchy, where primitive components are grouped into higher level composite components.[1] Performing localization within a component

---

[1] The grouping of components into higher level components must follow certain guidelines, as described in Chapter 6.

hierarchy assumes that fault behavior occurs locally; that is, faults propagate from one component to another component that is "close" in some way. This assumption is almost always true for fault behavior over a short time period.

Figure 4.2 depicts the process of localization. Starting at the top of a component hierarchy, the process first refines to the next level within that hierarchy. The sensors associated with each component at that level are examined. If all symptoms are a proper subset of the sensors associated with that component and only that component, the fault is said to be localized to that component. The process refines the localized component into its subcomponents. This continues until the fault cannot be localized further.

As an example, consider symptoms in EPR and EGT and the component hierarchy depicted in Figure 4.3. The initial input is the airplane representation ($Airplane$) and the set of symptoms (EGT and EPR). When we refine in the component hierarchy, we produce the set of components that are the subparts of $Airplane$, the $Propulsion-system$ and the $Hydraulic-system$. Iterating through this set, start with $Propulsion-system$. We then identify all sensors associated with $Propulsion-system$, resulting in the set $Fuel-flow$, EPR, EGT, N_1, N_2. Since the set of symptoms (EGT, EPR) is a proper subset of this set, we repeat the entire process at the next level in the component hierarchy.

Note that the localization process is mainly useful when faults first occur. Because of the propagation behavior, the localization may be increasingly difficult and accomplish less pruning as time (and the effect of the fault) progresses.

4.2.2 Generating Candidate Components

> *Each component in the engine subsystem is then proposed as a candidate responsible component.*

Once the fault is localized to a subsystem or set of subsystems, the candidate generation process can proceed. At present, the set of candidates is the union of the set(s) of primitive components of the localized subsystem(s). For our example, the fault is

Component   **Symptoms**

```
┌─────────────────┐
│ Descend a Level │
│ In Component    │
│ Hierarchy       │
└─────────────────┘

┌──────────────────────┐
│ For Each Component   │
│ Get Associated Sensors│
└──────────────────────┘

┌─────────────────┐
│ Are Symptoms    │
│ a Proper Subset of│
│ Sensors For This│
│ Component?      │
└─────────────────┘
```

Yes   No

```
┌──────────────────────┐
│ Were All Components  │
│ At This Level Tested ?│
└──────────────────────┘
```

No

Yes
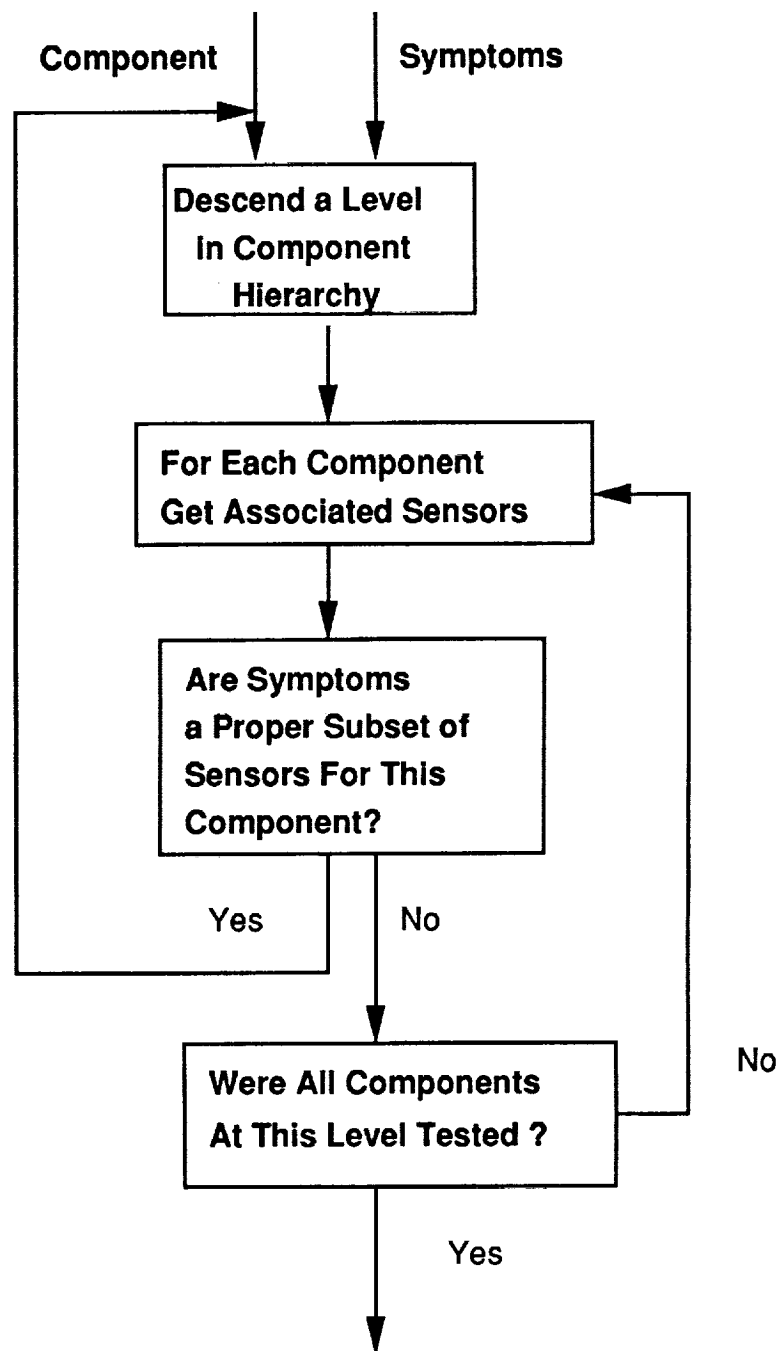
Figure 4.2: Localization Process.
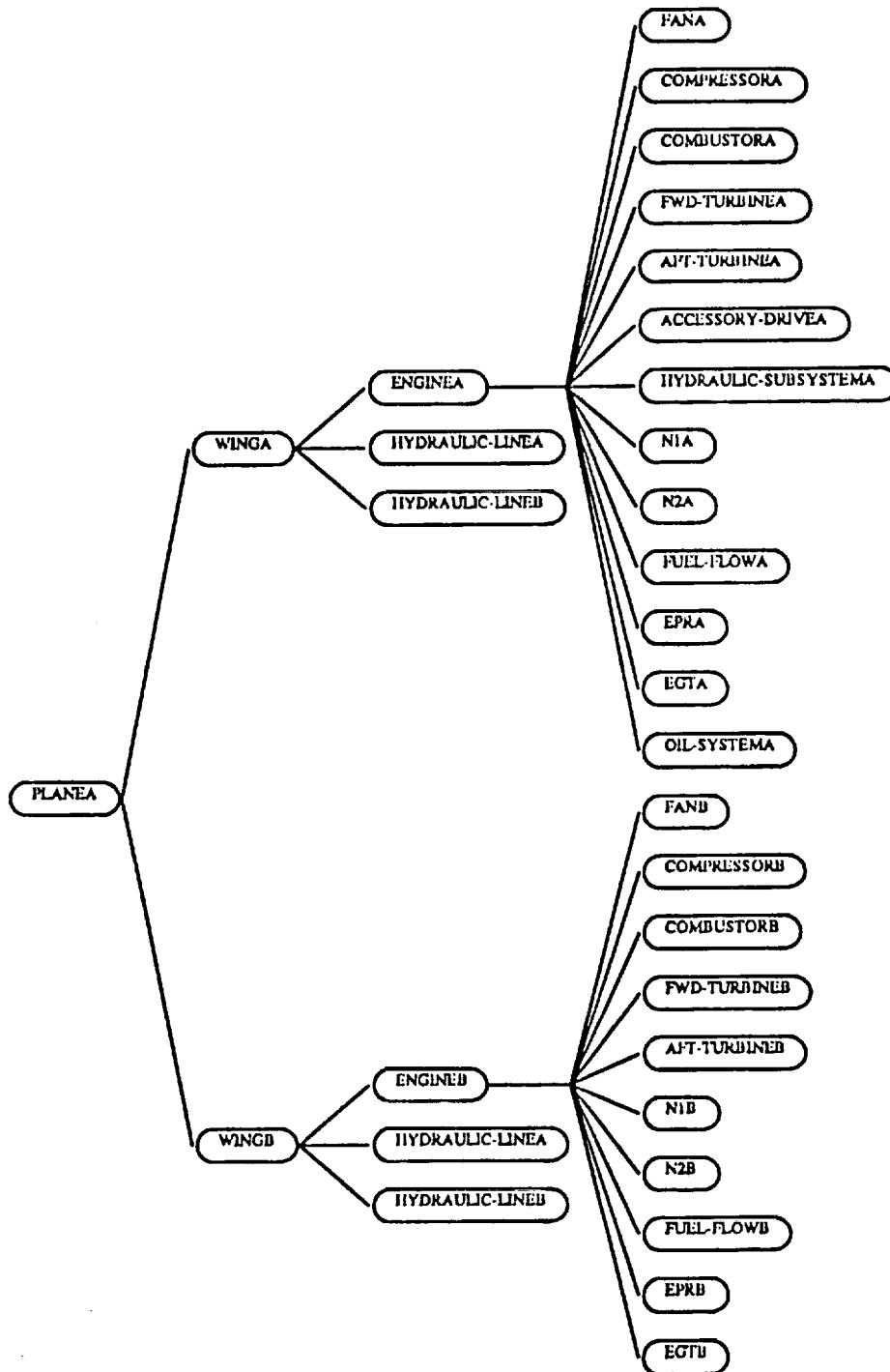
Figure 4.3: Functional Component Hierarchy.

localized to the engine. The set of primitive components in *Engine* is then generated, resulting in the following set: *Fan*, N_1, *Compressor*, N_2, *Combustor*, *Turbine*, *EGT*, *EPR*. Each component in this set is then proposed as the source of the fault. (Draphys does not order this set. If a probability of failure were known for each component, however, Draphys could order the set accordingly.)

4.2.3   Qualitative Simulation of Fault Propagation

For each proposed responsible component, Draphys generates a fault hypothesis by qualitatively simulating the fault propagation behavior. In this example, Draphys will first propose the fan as the responsible component. Draphys reasons about a model of the engine and its functional interconnections to determine that the high-pressure compressor and the N_1 sensor functionally depend on the fan. Knowing these interconnections, Draphys then attempts to continue simulating the propagation of the failure to functionally dependent components.

The input to this process is the set of symptoms and a candidate component. The output is a propagation path which begins with the candidate component and which identifies the system status consistent with the symptoms and candidate component.

Draphys determines the extent of the propagation of abnormal component status by simulating the propagation of the failure from the candidate component. First, the candidate component is proposed as the source of the problem. Then, all components that are dependent on the candidate component for proper operation are examined to determine whether the fault has affected them.

This simulation process is a traversal of the links in the graph representation of the functional model. Traversal of a link between two nodes represents simulation of a fault propagating along the path represented by that link between two components represented by the nodes connected by the link. Generating a hypothesis is the construction of a subgraph of the functional model to correspond to the part of the physical system affected by the fault. As the traversal process continues, each node that is determined to be affected, and each link that represents a fault propagation between two affected components, are added to the hypothesis subgraph.

Determining whether a component is affected by a fault is done by examining the

sensor(s) associated with the component. A sensor is *associated* with a component if it measures something about that component's operation. We have defined three association types, or ways that a sensor can be associated with a component. These three association types are related to what we can infer about the operational status of the component from the sensor status.

The first association type refers to sensors that can have abnormal readings without their associated component having abnormal operational status. If the sensor measuring some characteristic of a component's input is symptomatic, the effect of the fault is reflected in the input to the component, but it may not yet be affecting the operation of the component itself. Therefore, we do not consider the component to be affected by the fault unless further propagation can be identified. As an example, suppose we have a pressure sensor which measures the pressure of the air entering the fan. There may be an abnormal drop in air pressure, but this may not affect the fan yet.

The second type of association occurs when the sensor having a normal reading does not necessarily mean that the component's operational status is normal. For example, *EGT* measures the temperature of the exhaust from the turbine. If *EGT* is symptomatic, then we consider the turbine to be affected. However, *EGT* normal does not necessarily mean that the operation of the turbine is normal, because it might be faulted in some way without affecting the exhaust temperature.

The third association type occurs with sensors which reflect the operational status of the component. For example, a *Fuel − Pressure* sensor measures the pressure of the fuel coming out of the fuel pump. This sensor status reflects the status of the fuel pump, since this sensor provides a measure of the component's function. For tracking fault propagation, we consider a component to be affected if its sensor with this association type is symptomatic, and unaffected otherwise. That is, the component's operational status can be identified from the sensor's status.

There are other complicating factors in relating sensor status to component status. Computed sensors, such as *EPR*, may have associations with more than one component.

In Draphys, we identify the association between EPR and the fan and between EPR and the turbines. When EPR becomes abnormal, we must consider both associated components, since the individual pressure readings from which EPR is computed are unavailable.

.. If the component is determined to be unaffected, simulation of fault propagation stops along this particular path. Otherwise, propagation continues from each affected component. Draphys adds each affected component to a directed graph that represents the propagation path of the fault. The nodes in the graph represent the affected components, and a link from one node to another represents the propagation of the effect of the fault from the component corresponding to the first node to the component represented by the second node. When this propagation path contains all symptomatic sensors; i.e., when the explanation of the fault's behavior accounts for exactly the set of current symptoms, we consider it to be a valid hypothesis.

The model used to simulate functional propagation of a fault is component centered. It models the *functional dependencies* among components. The functional dependencies represent potential paths of intended interaction among components in the physical system that are designed into the system; i.e., component B is functionally dependent on component A if the proper operation of B depends on the proper operation of A. This occurs, for example, when the input to B depends on the output of A. Thus if A is malfunctioning, the effect may be that its output is incorrect. The input to B is abnormal, so B's operation is necessarily affected. B may be operating completely correctly given the inputs it has, but since the inputs are wrong, B's outputs will be wrong for the larger context. Thus the notion of abnormal operation of a component is dependent on the context of the overall operation of the physical system.

It is important to note that the functional dependencies represent potential paths of interaction. For example, component B may depend on component A for its proper operation, but only under certain circumstances, such as a switch being turned on. The context under which the interaction may occur is not modeled in this representation of functional relationships. A functional dependency between two components only
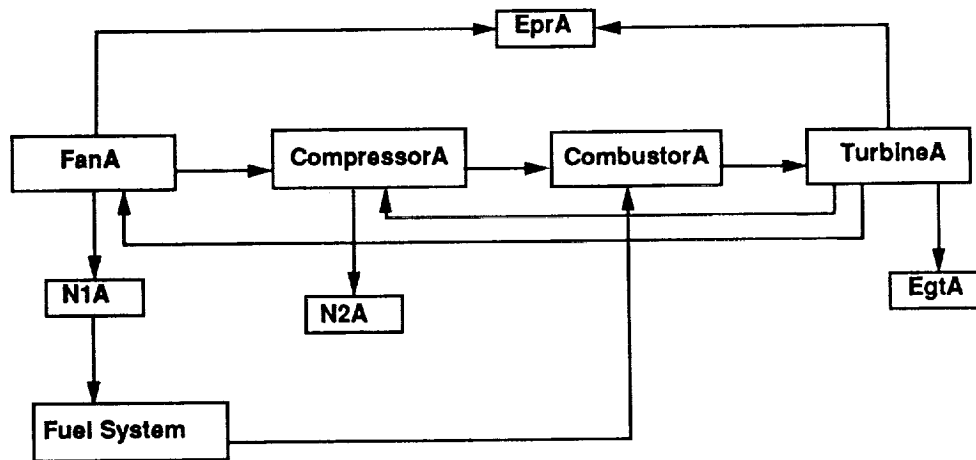
Figure 4.4: Functional Model of the Engine.

suggests that an interaction may occur, not that it must occur.

Functional dependencies are represented by directional links within a directed graph, where the nodes of the graph represent the components. Figure 4.4 illustrates the functional model of the engine. Cycles in the graph represent the feedback in the system.

This is a model of the normal functioning of the physical system, but it is abstract enough to model the system with many faults as well. This is mainly because this functional dependency relationship between components does not say how the components interact, merely that they could. Many faults will cause the components to interact along the same paths of interaction that are normally followed, so we can use this model to simulate the propagation of abnormal effects among the components. In fact, many faults which do not involve physical damage propagate along normal functional paths of interaction. It is this which gives the model the power to allow us to make inferences

about fault propagation.

> *Draphys then explores all remaining functional propagation paths. When all paths are exhausted, the hypothesis is tested for validity.*
> *Draphys then uses the same simulation process from each remaining candidate component to construct the hypothesis corresponding to that component. After all possibilities are explored, two valid hypotheses remain...The first is that the fan is the responsible component, and the second is that the N_1 sensor failed.*

The hypotheses created from this process are shown in Figure 4.5. Out of these, only two are valid. As we mentioned earlier, the propagation paths are subgraphs of the functional model, augmented by including system status information. When a subgraph includes all symptomatic sensors, it represents a valid hypothesis.

4.2.4 Discrimination Based On Dynamic Inputs

> *Extending this example further illustrates the incremental updating of hypotheses. Assume that a short time after the N_1 symptom was first detected and diagnosed, a symptom in N_2 is detected. Draphys then tries to extend the propagation path of all the valid hypotheses to explain the new symptoms. These paths are extended by continuing the qualitative simulation from the end of the propagation path in the old hypotheses.*

The reasoning at the higher abstraction level is designed and implemented to handle new symptoms that arrive after the initial diagnosis is done. To accomplish this, Draphys uses the same simulation process as when the hypothesis was initially created, except that it begins the simulation from the point in the valid hypothesis' propagation path where the propagation ended before.[2] For example, take the valid hypothesis where the fan is the source of the problem. The propagation ended at the fan rather than continuing to the compressor, because N_2 was not symptomatic when this hypothesis was created. However, when Draphys starts propagation again from the fan, simulation of propagation *can* continue to the compressor because its sensor is affected now. Draphys adds the compressor and N_2 sensor to the propagation path and continue propagating. There is a functional dependency path from the compressor to the

---

[2] Note that this assumes that all previously affected components remain affected.

VALID HYPOTHESES

⬤ Responsible Component
◍ Definitely Affected

HYPOTHESIS 1 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ Fan ⟩⟩⟩ ◍ N1 ◍

Propagation Type: Functional
Explained Symptoms: N1

HYPOTHESIS 2 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ N1

Propagation Type: Functional
Explained Symptoms: N1

INVALID HYPOTHESES

HYPOTHESIS 3 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ COMPRESSOR

Propagation Type: Functional
Explained Symptoms: N1

HYPOTHESIS 4 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ N2

Propagation Type: Functional
Explained Symptoms: N1

HYPOTHESIS 5 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ COMBUSTOR

Propagation Type: Functional
Explained Symptoms: N1

HYPOTHESIS 6 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ TURBINES

Propagation Type: Functional
Explained Symptoms: N1

HYPOTHESIS 7 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ EPR

Propagation Type: Functional
Explained Symptoms: N1

HYPOTHESIS 8 OF 8
Fault Type: Single Fault

Propagation Path And Component Status:

⬤ EGT

Propagation Type: Functional
Explained Symptoms: N1

Figure 4.5: All Hypotheses (Valid and Invalid) Resulting From a Symptom in N_1.

combustor, but the combustor has no associated sensor to confirm its operational status. Therefore, Draphys marks it as "possibly affected" and continues propagation. It follows a functional dependency path to the turbines, but all turbine sensors are non-symptomatic. Therefore, Draphys assumes that the turbines are unaffected and stops propagation at the combustor. If the turbines had been affected, Draphys would have marked the combustor as definitely affected as well.

Figure 4.6 shows the resulting hypothesis. Note that this accounts for all current symptoms. Also note that the updated propagation path enlarges the subgraph of the functional model to accommodate the new symptoms. This example illustrates the incremental nature of the hypothesis construction, since it builds on previously created hypotheses to account for new symptoms.

> *Continuing this example even further, suppose we now see symptoms in EPR and EGT. The hypothesis shown in Figure 4.6 is extended by continuing simulation from the point(s) in the propagation path where it stopped, resulting in the hypothesis shown in Figure 4.7. Once propagation to the turbine is confirmed, the status of the combustor is updated to "definitely affected," since the reasoning assumes that the fault would not affect the turbine without affecting the combustor.*

This example illustrates the benefit of the approach for diagnosing systems with feedback. If the engine were being diagnosed in an environment where the sequence of symptoms was not available, but the set of symptoms was, then all major (non-sensor) components would be candidates that could not be discriminated without more detailed information. However, examining the temporal sequence in which the symptoms appeared permits isolation of the source of the fault.

Note that we did not describe any multiple-fault hypotheses to explain the multiple symptoms. This is because we use a heuristic that we do not produce multiple-fault hypotheses when we can produce valid single-fault hypotheses that account for the current symptoms. The next chapter will discuss this heuristic in the more global context of multiple classes of faults.

Current Symptoms:

  N2 Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:

Propagation Type: Functional

Functional Propagation
Physical Propagation
Responsible Component
Definitely Affected
Possibly Affected
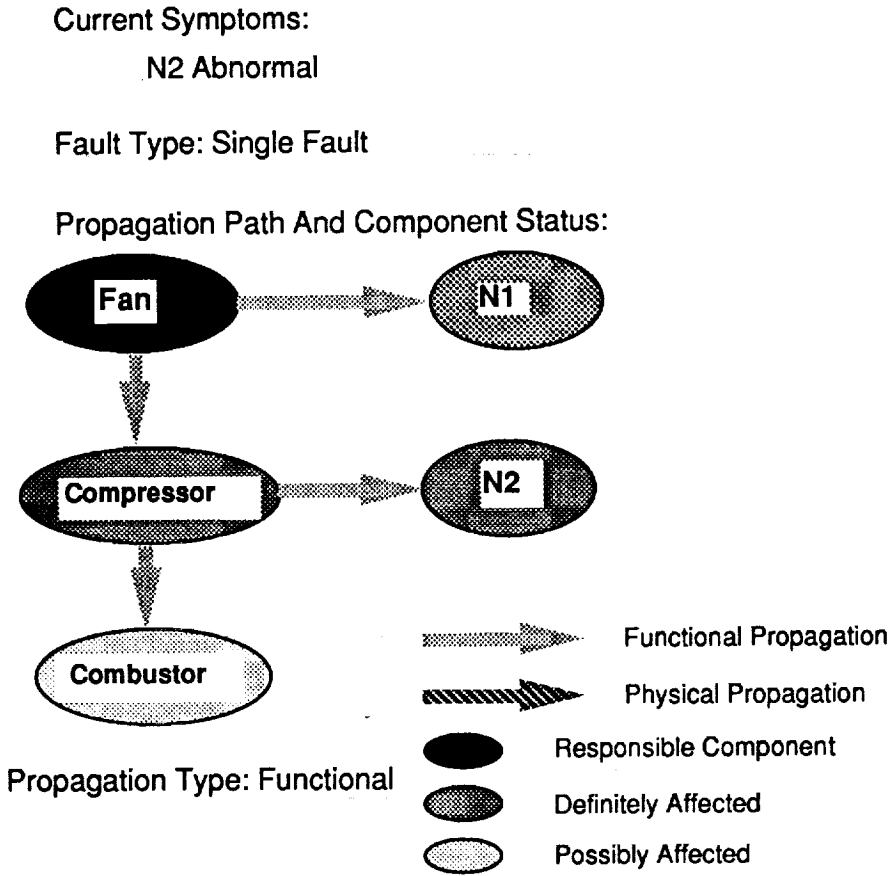
Figure 4.6: Hypothesis Remaining After a Symptom in N_2.

Current Symptoms:

EPR Abnormal
EGT Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:

Propagation Type: Functional

 Functional Propagation

Responsible Component

Definitely Affected

Possibly Affected
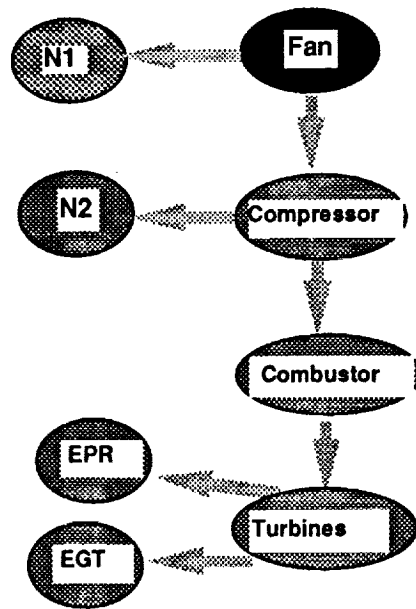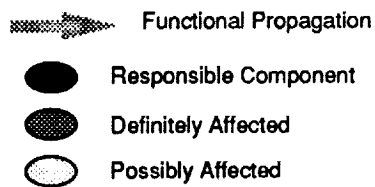
Figure 4.7: Hypothesis After Symptoms in *EPR* and *EGT*.

4.3 General Discussion

The previous section described the approach as implemented in Draphys and illustrated two major ideas in this research, the use of status abstraction for graceful degradation and the incremental, constructive approach to generating hypotheses. In the following sections, we consider both the structural abstraction used in the localization and status abstraction. We then examine several aspects of the abstractions and the reasoning. We then discuss the incremental updating of hypotheses, and highlight its advantages.

4.3.1 Structural Abstraction

The component hierarchy shown in Figure 4.3 and used for localization in Draphys represents an abstraction of the structure of the physical system. The type of abstraction used is an aggregation of lower level elements into higher level, composite components, which we also call subsystems. The resulting relationship between a composite component and one of its aggregated parts is a part-of relationship. Many diagnostic approaches use component hierarchies in their reasoning; we include a discussion here on how the component hierarchy is defined and used in Draphys.

The aggregation of components into higher levels in this component hierarchy was done based on a functional grouping.[3] That is, components that contribute to a particular functionality are grouped together. For example, all components which are considered to be part of the engine are grouped together. Similarly, components which contribute to the workings of the hydraulics are grouped together into a hydraulic subsystem. To support efficient localization, this hierarchy must form a tree, in that a component may only be part of one composite component. If the hierarchy does not form a tree, then the localization process cannot prune entire branches.

The purpose of the grouping is to aid in the localization; that is, to support the efficient exoneration of components. The motivation for using the functional grouping, as described, is the observation that faults tend to propagate (at least in short time

---

[3] The grouping also depends on sensor placement; see chapter 6 for a discussion of this.

periods) to components that are designed to interact with the responsible component. Therefore, by grouping the components according to function, we improve the efficiency of the localization process for many faults.

Components which are functionally grouped may not be physically located together. For example, the hydraulic subsystem has parts which are physically located throughout both airplane wings and the fuselage. One could envision a component hierarchy where the components are aggregated based on physical location rather than functionality. Such a component hierarchy can be useful, and we will discuss the use of such a component hierarchy in the next chapter.

As mentioned previously, the component hierarchy must be defined in a particular way in order to permit pruning of entire branches of the hierarchy. In Chapter 6, we identify some knowledge engineering guidelines for building the component hierarchies.

4.3.2 Status Abstraction

In this section, we discuss the form of abstraction that we have named status abstraction. This form of abstraction is a key factor in our approach to graceful degradation in diagnostic problem solving, because it supports reasoning about the fault and its propagation behavior in less detail as more specific knowledge is not available. We first discuss what is being abstracted, and the corresponding behavioral abstraction. We then consider other ways of categorizing faults than by operational status of components.

In different levels of status abstraction, operational status of a physical component is described at different levels of detail.[4] For example, at the higher level of status abstraction, a component's operational status is described as either normal or abnormal. The lower level describes in more detail how the component is abnormal. For example, at the lower level, an oil filter would be described as clogged. That description of how the component is broken is abstracted to abnormal at the higher level of status abstraction.

---

[4]We only consider abstraction of the operational status of primitive components here, although the same could be done for composite components.
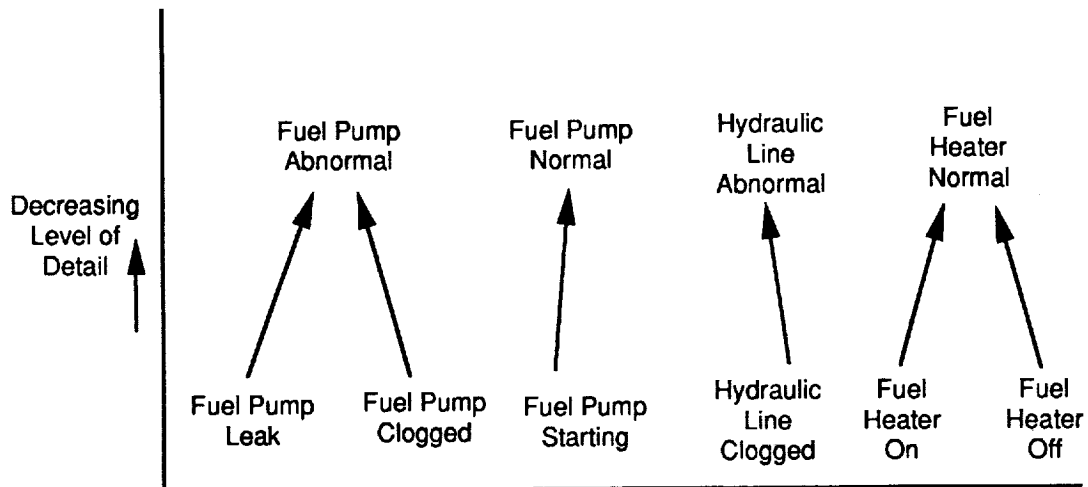
Figure 4.8: Examples of Component Status Abstraction.

Of course, describing how a component is abnormal can be done at differing levels of detail, but we only implemented one level of detail below the binary level. We show several examples of components' status and their abstractions in Figure 4.8. Note that different operational statuses for the fuel line are all grouped together at the higher level into "fuel line abnormal."

To determine what the operational status of a component is at a particular level of abstraction, a diagnostician should use a corresponding level of behavior. At the higher level of status abstraction, we only describe behavior as being normal or abnormal; that is, a parameter value used to describe the behavior of the device is either normal or abnormal for the current operating conditions. At the lower level, we are trying to determine a more detailed description of operational status, so we need to use a correspondingly more detailed description of behavior.

The advantage of using status abstraction to support graceful degradation is because we need less detailed descriptions of behavior at higher abstraction levels. For example, it is easier to determine that a parameter is abnormal than to determine that the same

parameter was high or fluctuating, because no reasoning about temporal intervals is necessary. The difference in determining behavioral descriptions at different levels of detail may seem insignificant when looking at a single parameter at a single point in time. However, the difference becomes clearer and more significant when looking at several parameters' behavior over time and relationship with each other.

There is more to describing behavior than the level of detail of a particular parameter value, of course. For our diagnostic reasoning, it is also necessary to identify propagation behavior. This, too, must be described at the appropriate level of abstraction. The representation of the physical system used to diagnose at a particular abstraction level should be consistent with the necessary level of behavioral detail.

For the higher level of status abstraction, we must identify the paths of interaction followed by the fault as it propagates. The functional model used in Draphys is a model of the normal functional dependencies designed into the physical system, but it is abstract enough that it allows us to model many faulted systems as well. Moreover, it allows us to model faulted systems with dependent failures, if the failures follow the paths of interaction in the normal functional model (which many do). The reason we can model even dependent failures is because our description of behavior is so abstract. As we reason at higher abstraction levels, we achieve greater breadth of fault coverage at the cost of specificity.

It is important to note that the abstraction levels are independent of the reasoning techniques applied at those levels. For example, one could use associational reasoning or model-based reasoning at any level of status abstraction. Another important point is that this is not just abstraction of abnormal operational status, but normal operational status as well. However, since we are performing diagnosis, the abnormal status is the motivation for the abstraction scheme chosen.

Categorization of the Faults by Component Operational Status

Draphys abstracts faults according to the operational status of the physical components that make up the device, but it could categorize faults in other ways. In artifact diagnosis, we could categorize the faults by structural abstraction. One of the motivating factors behind performing hierarchical diagnosis in a structural hierarchy is that it does provide some capability for graceful degradation. That is, the diagnosis goes as far down the hierarchy as it can, providing useful diagnosis information even when it cannot diagnose to the lowest level in the hierarchy. Another example in medicine might be disease categories; e.g., cancer is one such category.

The choice of categorization scheme might be based on many motivations. A major organizational motivation is the existence of a corrective action corresponding to a fault category. If there is some action or therapy associated with the general fault category (which may not be the same as actions for specific faults within the category), then the diagnostician can fall back on that action when the specific fault cannot be identified. For example, if a doctor can identify a problem as a bacterial infection, but cannot identify what type, the recommended prescription might be a wide-spectrum antibiotic. If the specific bacteria type can be identified, the doctor would prescribe an antibiotic specific to that bacteria.

In Draphys, we assume that identifying the best (safest, most likely to correct or compensate) response is a major motivating factor, and that specific hypotheses are associated with the most-preferred actions, but the more general hypotheses and associated default actions are better than nothing.

Levels of Status Abstraction

Draphys currently uses two status abstraction levels, but could include more levels. It is possible to have a level of status abstraction between the two current status abstraction levels that identifies the level of severity of the fault's effect on the component. For example, at the most specific level we might describe an eroded compressor blade. At

an intermediate level, the compressor's operational status is abstracted to "operating but at a degraded level." (Another operational status at this level might be "totally nonoperational.") At the highest abstraction level, the operation status is "abnormal." An even more detailed status abstraction level might refine "eroded compressor blade" to a description of how badly eroded it is.

The levels of abstraction chosen might be determined by the information requirements of the human operating the physical system. For instance, they might correspond to the default actions available as less specific information about the fault can be determined. Conversely, overly detailed levels serve no purpose if the additional discriminating power they appear to provide has no practical consequences. Even when actions are not available, the information at a particular abstraction level might still be useful to the human, just as localization can be useful information to the human. This must be determined by examining the domain itself, and what information the human operator can use to make decisions.

4.3.3 Constructive Versus Classification Problem Solving

It is interesting to note that the reasoning at the higher level of status abstraction is constructive. That is, hypotheses are constructed (and tested) via simulation rather than retrieved from a set of pre-enumerated solutions as in classification problem solving [9].

We could create associational rules at the higher level of status abstraction, but this is undesirable for reasons similar to those discussed in the previous chapter, as follows. First, it is difficult to know how much propagation to express in the rules. If the rules only express the initial symptoms, a rule can be satisfied as soon as symptoms appear. This has the disadvantage that we cannot easily use subsequent symptoms to distinguish hypotheses. Moreover, it is difficult to determine dynamic system status using associational rules because of the variability of fault propagation, both over time and extent of effect.

Second, since we base the reasoning on use of a model of normal functional structure, there is no need to store or search through a large set of rules that represent each propagation step. The same model can be used for all faults whose propagation behavior follows the functional dependency paths designed into the physical system. This is because the model of the normally operating system used in Draphys is also a model of the system under many fault conditions as well. Moreover, determining system status during the simulation process is straightforward.

Third, since all symptoms are treated identically at this level of status abstraction, a single model and a simple inference technique (simulation of propagation of a single fault) are sufficient to diagnose faults within this fault class. The only search process that takes place is the generation of candidates. Because of this, the efficiency of this reasoning process depends to a great extent on the pruning ability of the localization process. In turn, the localization process's pruning ability depends on the proper organization of the component hierarchy, as discussed earlier, and on sensor sampling rate. If much time passes between samples, more fault propagation will occur, making it more difficult to isolate the fault to a small set of candidate components. Given a candidate component, the level of abstraction of the physical system's behavior must be general enough that using simulation to construct hypotheses will be efficient.

We could do the diagnostic reasoning within the specific level of abstraction using a constructive approach. To obtain the same detailed fault information as the specific associational knowledge, such an approach would represent simulation of the broken device. However, physical-system modeling technology cannot adequately represent specific system behavior. For example, representing transient behavior is not well understood; existing modeling techniques typically assume steady state behavior of the physical system [44]. But a fault by its very nature is often transient, although subsequent behavior may assume some steady state characteristics. This would not be a problem at the higher abstraction level because the level of detail at which we describe the behavior of the system abstracts away details and just identifies paths of interaction. One advantage of empirical associational rules, though, is that they can express

empirical relationships between symptoms and faults for which no underlying theory or model exists. A diagnostician may know from experience that there is an association between certain symptoms and a particular fault (or corrective action) without knowing why they are associated. We did not encounter any examples of this in our domain, but such a situation may occur in other domains.

The issue of constructive versus classification problem solving is explored further in the next chapter.

4.4 Related Work

We consider related work in both AI and psychology here. In the AI area, we consider the use of abstraction for problem solving, diagnosis of novel faults, and incremental diagnosis. Most of the relevant work in diagnosis involves the use of models of structure and behavior to relieve the brittleness of current knowledge-based systems. We also address approaches which use multiple kinds of abstraction for diagnosis. In psychology, human models of performance were developed based on analysis of verbal protocols of humans doing fault diagnosis.

4.4.1 Using Abstraction for Diagnostic Problem Solving

Abstraction is a powerful technique for problem solving in general [48], [28], [10]. We consider here those approaches to diagnosis which generate abstract fault hypotheses, and whether that abstraction is used implicitly or explicitly.

4.4.2 Diagnosing Novel Faults

In AI, most approaches to diagnosis of novel faults use the same definition we do, that the fault is one for which specific symptom-fault knowledge is not available. These approaches use model-based diagnostic reasoning based on models of structure and behavior. First, we will discuss the model-based diagnostic approaches, then we examine the specificity of the diagnostic output of these approaches. We also discuss their

appropriateness for operative diagnosis.

Current research in model-based diagnosis involves reasoning about models of structure and behavior of the device under consideration, also described as reasoning from first principles, for example [11], [22], [47], [20], [24], [7], [25], [18]; detailed surveys may be found in [26] and [13].

The model-based diagnosis process is generally viewed as performing prediction, candidate generation, and discrimination. Prediction is the process of generating expectations about behavior, and for identifying discrepancies between expectations and actual system behavior. For example, given a device model built as a network of components, each with its own behavior description, behavior prediction can be accomplished by propagating the individual behaviors of each component. The behavior prediction in Draphys generates expected behavior for each component compared with the normal operation of the device as a whole. This computation is discussed in section 1.3.2; we also describe this process as monitoring rather than prediction. The reason that we use this type of monitoring is because we need to provide a comprehensive system status to the human operator. To do that, we must track the progression of the fault, even when diagnosing systems with feedback.

Candidate generation produces one or more explanations for any discrepancies found. There are several ways to perform this task. One approach uses only knowledge of normal system behavior. In this approach, each prediction is associated with the set of components whose correct behavior supports that prediction. Therefore, any discrepancy between that prediction and actual behavior can be explained by the failure of one or more of the components in that set. If there are several discrepancies, the broken components must form a covering set of all discrepancies. This approach is the basis of the candidate generation procedures in many research efforts, which can be described as a dependency tracing process, although the details and formality of the descriptions differ (e.g., [22], [11], [49], [16], [25], [23] and [47]).

Another approach is to use knowledge about how components fail, or fault models

[6], [39], besides using knowledge about how systems work. After finding a component whose failure could explain all symptoms, the effects of known failure types are simulated. Thus the model-based part of the reasoning is used to identify the faulty component, and the fault models are used to identify the specific failure. If the list of known failures is exhaustive, this approach can be used to exonerate components, as done in SOPHIE [6]. Pan [39] uses fault models to represent known dependent failures, and explicitly reasons about failures causing other failures. This is particularly appropriate for diagnosis of complex physical systems, because dependent failures are a real possibility. ABEL [41] uses fault models to check the consistency of candidates as they are generated.

Discrimination is usually done by taking additional measurements or providing different inputs to test the resulting output. For example, [16] proposes an information-theoretic approach to choosing measurements. Shirley [55] presents an approach to generating test inputs by exploiting designed behavior. In operative diagnosis, discrimination is limited because additional information is hard to acquire.

. Many approaches to novel fault diagnosis produce diagnostic information that is more abstract than the specific associational knowledge [11], [22], [29], [16], [20], [47]. That is, they identify the component that is the source of the problem and (usually implicitly) make some assumptions about the fault's propagation behavior being functional. The class of faults diagnosed in the implementation of these approaches contains more specific hypotheses than the fault hypotheses generated by functional propagation in Draphys, because they assume no dependent failures. Figure 4.9 shows the level of detail produced by some of these other diagnostic approaches.

The approaches that integrate model-based reasoning with fault models are often performing at multiple levels of abstraction, even if they are not always explicitly labelled as such. Examples include [20] and [39]. One approach that uses and explicitly identifies multiple levels of abstraction was developed by Abu-Hanna and Gold [2]. This approach uses multiple levels of structural and behavioral abstraction. The behavioral abstractions correspond to different levels in the structural hierarchy. That

Decreasing
Level of
Detail in
Hypotheses

Draphys - higher level of status abstraction

Davis [11]   Fink [20]   GDE [16]

Pan [39]
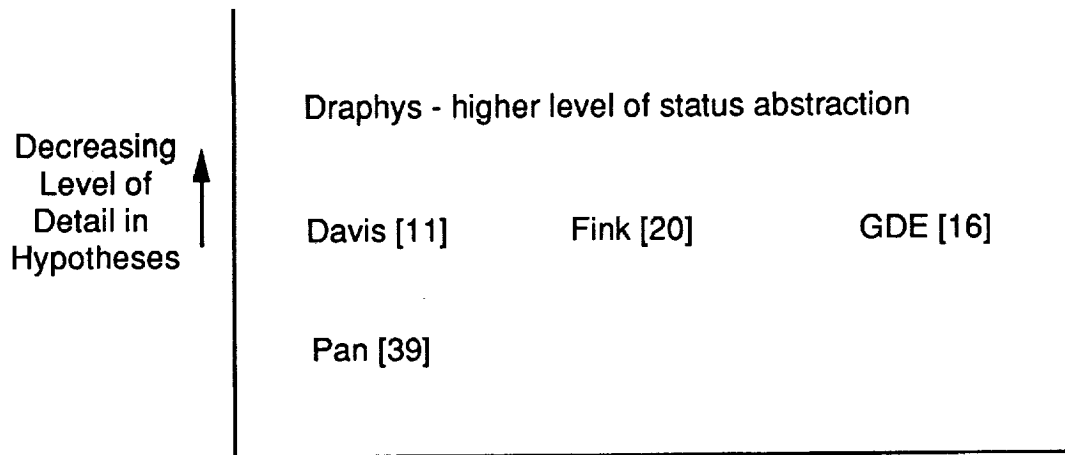
Figure 4.9: Level of Detail of Hypotheses from Various Diagnostic Approaches

is, a qualitative component behavioral state is an abstraction of several subcomponent states.

In general, diagnosis programs that use the model-based approach generate hypotheses that are less specific than the associational knowledge (unless they are augmented with specific fault models). That is, they describe what component is the source of the problem. The faults they diagnose are faults that propagate within a particular model, usually the functional model. Except for [11], each diagnoses one class of fault. In the class of failure diagnosed by most of these approaches, only one component is broken and it does not cause dependent failures. Symptoms may appear on other components because the input they are receiving is incorrect, but the components are operating correctly for the inputs they have. None of the approaches are designed to handle systems with feedback. As a result, they do not completely address the issue of multiple consequences of a single initial fault in a system that does have feedback.

Incremental Diagnosis

One of the contributions of this thesis is the incremental construction of hypotheses. Diagnosis in Draphys is incremental in that it builds on previously created hypotheses when new symptoms arrive. Another approach that performs incremental diagnosis is GDE [16]. GDE is similar in the general idea of incrementally building on previous computations when new information becomes available to discriminate hypotheses. However, the new information that GDE uses is a measurement taken to get more information about the faulty device. GDE assumes that all fault propagation has already occurred when the diagnosis is done. In contrast, Draphys uses the fault propagation behavior of the system as a discriminator. Therefore, although both approaches are incremental, the information they use is quite different and how the information is used to increment the diagnosis is correspondingly different as well.

4.4.3 Human Performance

Rasmussen's work on human performance modeling both in routine task environments and during unfamiliar task conditions [45] is an important piece of work. This model describes human problem-solving performance at three different levels, as shown in Figure 4.10. The lowest level represents skill-based behavior, which is sensory-motor performance. Riding a bicycle is an example of a skill-based behavior.

The next higher level represents what Rasmussen calls rule-based behavior. In this level, familiar situations are typically controlled by a stored rule or procedure which may have been derived empirically during previous occasions, communicated from another person's knowledge, or created during problem solving or planning. Rasmussen states that the boundary between skill-based and rule-based performance is not always distinct, and that much depends on the level of training and attention of the person.

When an unfamiliar situation arises and no rules for response are available from previous experience, the performance must move to a higher conceptual level that Rasmussen calls the knowledge-based level. At this level of reasoning, the internal

KNOWLEDGE-BASED
BEHAVIOR

RULE-BASED
BEHAVIOR

SKILL-BASED
BEHAVIOR

Symbols

Identification

Decision
Of Task

Planning

Signs

Recognition

Association
State/Task

Stored Rules
For Tasks

Feature
Formation

Signs

Automated
Sensory-motor
Patterns

Sensory Input

Signals

Actions
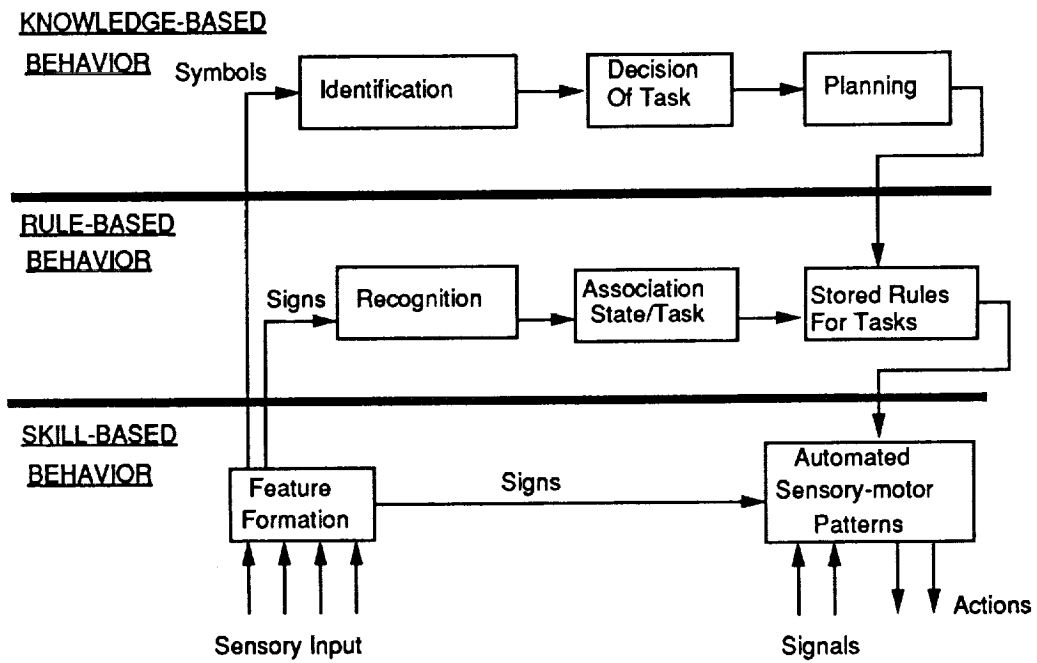
Figure 4.10: Levels of Performance of Skilled Human Operators (from [45]).

structure of the system being reasoned about is explicitly represented by a "mental model" which may take several different forms.

What do these levels have to do with graceful degradation in performance of diagnosis? If one considers knowledge-based behavior to be the top level, then as a human becomes more proficient at some task, the reasoning is compiled into the lower levels. For instance, as one practices driving a car or playing a musical instrument, performance becomes proceduralized eventually to skill-based behavior. Not all behavior becomes skill-based, but most behavior can be compiled to at least the rule-based level of performance. Thus familiar problem solving tasks are performed at the lower levels, but unfamiliar, non-routine tasks must take place at the highest, knowledge-based level. One reason we attribute the ability to degrade gracefully to humans is because they can reason at different levels as necessary.

Because humans have limited attention span and short term memory, they use various techniques to facilitate mental data processing. Rasmussen identifies three such techniques:

- *Aggregation* - Elements of a representation are aggregated into larger units, chunks, within the same category of mental model as familiarity with the context increases.

- *Abstraction* - The representation of properties of a system or environment is generalize or abstracted to a model category at a higher level of abstraction.

- *Analogies and Use of Ready-Made Solutions* - The representation is transferred to a category of model in which solutions are already known or rules are available to generate a solution.

This thesis may be viewed as an exploration of the use of abstraction to provide a structured way of defining Rasmussen's rule-based and knowledge-based levels of problem solving. Our specific associational rules correspond to Rasmussen's rule-based reasoning and our higher level of abstraction corresponds to his knowledge-based level. We did not encounter a need to use any reasoning corresponding to the skill-based level

in the diagnostic task, although the recovery task may utilize such reasoning.

4.5 Limitations of Using a Single Physical-System Model

Even though the diagnostic reasoning at a higher level of abstraction can be used for diagnosing some novel faults, the reasoning as described assumes that the fault propagates along the normal functional paths of interaction. Not all faults do. Davis [11] uses the bridge fault in circuits as an example of a fault that propagates along paths of interaction in a model of physical rather than functional adjacency. In the aircraft domain, similar concerns arise. As we discuss in the next chapter, faults can occur that cause damage because of physical proximity rather than functional interaction of components. The functional model does not represent such fault propagation behavior. Therefore, the diagnostic reasoning using a functional model, while necessary for some faults, is not sufficient for others.

4.6 Summary of Diagnosis of Novel Faults Via Abstraction

This chapter presented an approach based on reasoning at a higher status abstraction level to diagnose faults which cannot be diagnosed using any of the specific associational knowledge that is available. The type of abstraction used, called status abstraction, supports graceful degradation in the presence of novel faults because it requires less specific knowledge about faults, especially faulted system behavior at the higher abstraction level, but still produces useful diagnostic information. It provides a tradeoff between breadth of fault coverage and specificity of the diagnosis; it is the specificity of the diagnosis that degrades.

The diagnosis at the higher status abstraction level is done using model-based reasoning. We showed that some novel faults can be diagnosed using the functional model at the higher abstraction level, but not all.

C-2

Chapter 5

Diagnosis of Multiple Propagation Types and Fault Classes

As we showed in the previous chapter, using only the functional model of the physical system is not enough, because not all faults propagate locally within the functional model. In this chapter, we introduce the idea of multiple propagation types, and the partitioning of the hypothesis space to correspond to different fault classes. In particular, we consider physical and functional propagation, and multiple as well as single faults.

5.1 The Need for Multiple Models

Another model than the functional model described in Chapter 4 is necessary to simulate certain fault propagation behavior. The simplest example of this is the class of faults where a component physically damages another component, e.g., a component overheats and the heat damages a physically proximate component. Such a path of interaction does not follow any path within the functional model. To use Davis' terminology [11], the two components are not adjacent in the functional model. However, the components are adjacent in the physical model. Here, finding the appropriate model means finding the appropriate adjacency relationship. Note that although we are referring to adjacency as though it were a binary relationship (i.e., two components are either adjacent or not), adjacency is actually more complex than that. For example, in representing physical proximity, one might say that the smaller the distance between two components, the more adjacent they are. However, we could formulate that relationship as a binary one by stating that components closer together than some threshold

d are adjacent, and all others are not. Moreover, as Davis points out, there are many different types of adjacency; e.g., magnetic or electrical.

Use of adjacency for diagnosis is based on the assumption that interaction between two components does not occur at a distance (relative to adjacency, that is), and that the notion of adjacency pertains to a particular model. While this almost always[1] holds for interaction between two components, that only considers behavior local to those two components. If one needs a more comprehensive view of a fault's behavior and associated interactions in all affected components, one needs to take a more global perspective. We showed that there are faults in this domain where more than one model is necessary to explain the behavior of a particular fault occurrence. Local interaction between any two components may occur within a single model, but there are multiple types of interaction when looking at all interactions among all components in a particular fault occurrence. For example, when a fan blade damages a hydraulic line, that unintended interaction is because of their physical adjacency. When the broken fan fails to drive the compressor properly, that interaction is because of functional adjacency.

5.2   Partitioning of the Hypothesis Space

In the previous chapter, we showed how we could perform diagnostic reasoning at a higher abstraction level to support diagnosis of some novel faults. However, abstraction alone is not sufficient for situations where multiple models of the physical system are needed. The need for a different model of the physical system can occur within each abstraction level.

From our view of diagnosis as problem solving in a hypothesis space, the solution is to partition the (single fault) hypothesis space according to the specificity of the hypotheses as well as the fault's propagation behavior; that is, by the model(s) used in

---

[1]Some fault cases involve interaction at a distance, such as a fan blade becoming a projectile and damaging another engine, but these are rare.

the fault's diagnosis. At the higher level of status abstraction, such a partitioning groups the single-fault hypotheses into three such categories: functional propagation only, physical propagation only, and hybrid propagation within both physical and functional models.

This partitioning is not restricted solely to a particular level of abstraction. When choosing a problem solving technique for a particular fault occurrence, both an abstraction level and propagation type must be selected. Therefore, we define *fault classes* that organize hypotheses by abstraction level, physical system model necessary to simulate the fault's propagation behavior, and associated problem solving techniques. Essentially, we define fault classes according to the level of detail of what we know about the fault and its behavior.

At present, Draphys defines four single-fault classes, as shown in Figure 5.1. Associated with each class are the assumptions about the fault's behavior, and the corresponding models, abstraction level, and problem solving techniques. Other single-fault classes might eventually be included, such as intermittent faults and design errors, but our purpose in this research was to explore a framework in which such classes might be included as desired.

We already discussed diagnosis of known, commonly occurring faults in Chapter 3 and novel, functional-propagation faults in Chapter 4. We now explore the diagnostic reasoning within the remaining fault classes.

5.2.1 Physical Propagation Only

As before, the example for this class of fault described in Chapter 2 is presented together with the corresponding diagnostic reasoning.

Suppose that the fault was fan blade separation and that the fan blade broke off and damaged a hydraulic line in the wing to which the engine was attached. Draphys detects symptoms in N_1 and in the hydraulic pressure sensor...by knowing that the fan is physically adjacent to the wing containing the hydraulic line, propagation from the engine to the wing can be identified.
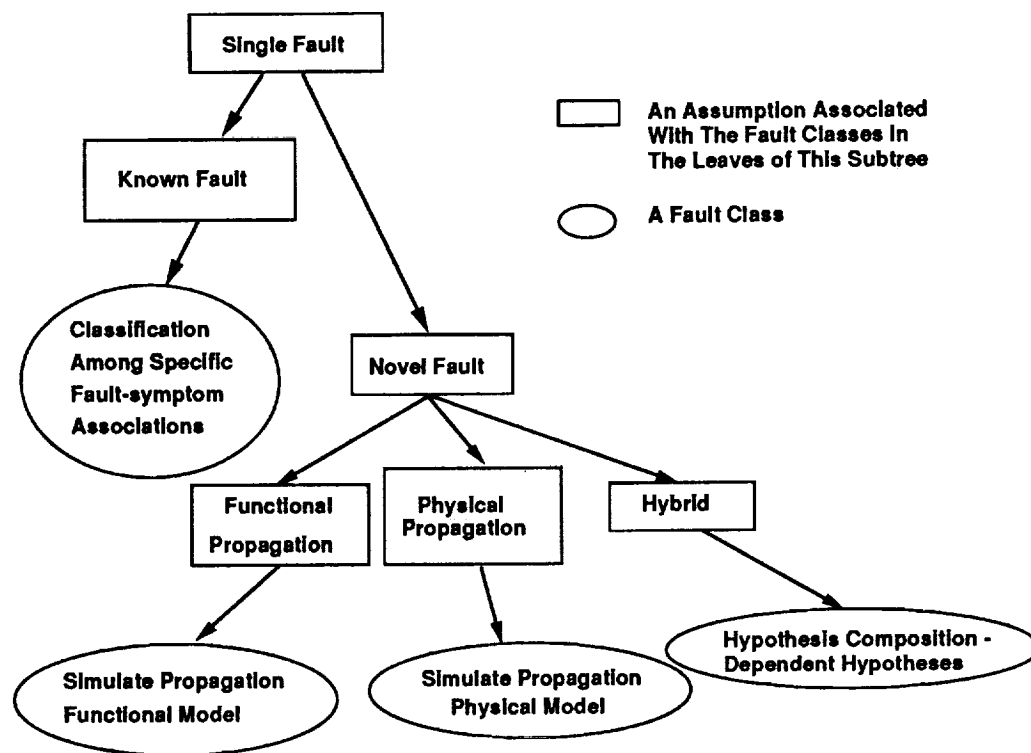
Figure 5.1: Single-Fault Classes.
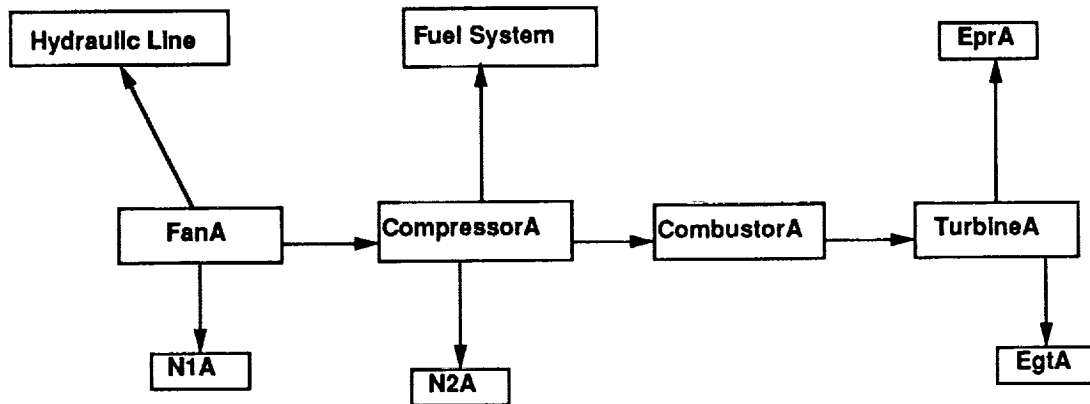
Figure 5.2: Engine Physical Propagation Model.

Draphys simulates the propagation of the fault's abnormal effects in the physical propagation model, just as it simulates fault propagation in the functional model. As in the functional model, a graph represents the components and the interconnections. In the physical model, the links are not functional dependencies but rather are physical dependencies. We call them dependencies, although they may better be described as "potential paths of interaction because of physical proximity." For lack of a more concise term, we name these links in a consistent manner with the functional dependency relationships. These links represent paths of interaction followed by the possible physical damage that can occur between components when a component breaks or malfunctions and are a subset of the physical proximity relationships in the physical system. The physical model of the engine that Draphys uses is shown in Figure 5.2.

The possible physical damage occurs because of physical proximity of the two components. However, it would be inefficient to represent all the non-directional physical

proximity relationships if there is no (or very little) possibility of one component physically damaging another. We used the example earlier of the fan blade breaking off and damaging the hydraulic line. Because it is known that fan blades can break, the physical model should include the (physical) potential path of interactions to components physically proximate to the fan, such as the hydraulic line. However, it is highly unlikely that the hydraulic line would break and damage the fan, so we do not represent an interaction path in the other direction. By only representing physical dependencies based on design knowledge, we substantially reduce the number of paths that the diagnostic reasoning must explore.

Representing physical relationships as just described is based on design knowledge of how one component might break and physically damage another. In that sense, the physical relationships included in the model represent knowledge about how the system breaks. However, it is not a model of how a component breaks and the resultant behavior, but rather it is a fault propagation model, describing the possible fault propagation paths. It is not making a very strong (i.e., detailed or stringent) statement about what fault behavior to expect; merely that these are the paths of physical proximity that one might expect to see a fault follow. As depicted in Figure 5.3, there are varying degrees of knowledge about faults, depending on how strongly the knowledge identifies how the physical system will fail. At one end of the continuum, only information about the normal physical system is included. Model-based troubleshooting as discussed in [13] falls at this end of the continuum. At the other end, only knowledge about how the individual components fail is included. As in our physical dependencies, adding some knowledge about how the system fails can improve efficiency, and still be in the middle of the continuum. However, it does limit the coverage of faults. The choice of how much information to include about how the system fails should be made explicitly when designing the diagnostic approach.

```
How System                                    How System Works
Fails _____  _____
```

| empirical fault-symptom | physical-dependency | Model-based |
| associations | model | Troubleshooting |
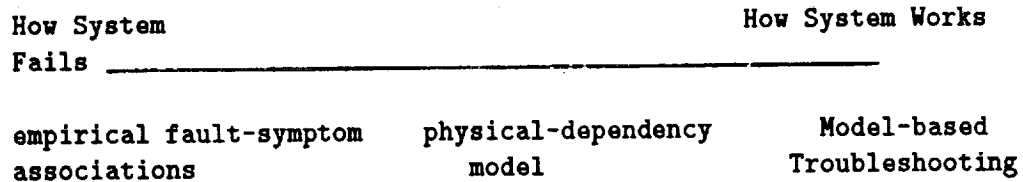
Figure 5.3: Continuum of Knowledge About How the Physical System Fails.

One important characteristic of the physical dependencies is to note that these relationships generally represent *unintended* interactions that result from physical proximity, where the functional dependencies represent *intended* physical interactions. However, when building the model, we must decide which potential interactions to represent. If we were concerned with externally-caused physical damage (e.g., battle damage caused by bullets), Draphys' model of physical dependencies would not be sufficient. We would have to include all physical proximity relationships, because we could not eliminate possible interactions based on design knowledge. Many more paths in the model would have to be represented and explored than Draphys now does.

Another interesting point concerning the physical-propagation hypotheses is that localization must be done in a physical structure hierarchy. In the previous fault class, that of functional-propagation faults, localization was done in a component hierarchy where components were aggregated into higher levels based on functional grouping. However, since we are reasoning about physical propagation, localization in the functional component hierarchy might exonerate a component that is propagating physically. For example, suppose component A breaks and propagates physically to component X. Suppose A and X are functionally part of different subsystems, but physically grouped together. Localization in the functional hierarchy might exonerate A, if no sensors in A's subsystem are symptomatic yet. Therefore, when looking for physical propagation, localization should be done in the physical component hierarchy.

None of the pilots described reasoning about physical damage. We will see in the accident analysis in Chapter 6 that many of the engine faults that resulted in accidents involved physical damage. It is possible that diagnosing this type of fault propagation is difficult for humans, or at least this type of fault is often overlooked.

5.2.2 Hybrid Propagation

For some faults, a single model (physical or functional) cannot adequately model the entire propagation behavior of the fault. In the aircraft domain, this most commonly occurs when one component physically damages another, resulting in functional propagation from the two damaged components. The example from Chapter 2, partially shown below, is one where physical propagation caused functional propagation.

> *Suppose we have a fan blade failure, with resulting symptoms in N_1, N_2, and the hydraulic pressure sensor. The hypothesis Draphys generates for this example is presented in Figure 5.4.*

Given that we know that functional propagation almost always follows physical propagation, we could modify the generator to simulate propagation of the fault's effects in the two models simultaneously. However, this is unnecessary. The hypotheses describing the propagation within a single model were already generated earlier, when the fault classes for a single propagation type were explored. Therefore, Draphys takes advantage of the earlier processing by composing the primitive hypotheses together. Figure 5.4 shows the composition of three such primitive hypotheses. One describes the physical propagation from the fan to the hydraulic line, one describes the functional propagation within the engine, and one describes the functional propagation within the hydraulic system.

This composition is not done randomly, since we can apply knowledge about how faults propagate in different models. In Draphys, a single heuristic is used to guide the composition of primitive hypotheses. This heuristic says that physical propagation can be followed by functional propagation. No other compositions are considered.

HYPOTHESIS 1 OF 1

Current Symptoms:

N1 Abnormal

N2 Abnormal

Hydraulic Pressure Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:

Functional Propagation

Physical Propagation

Responsible Component

Definitely Affected

Possibly Affected

N1   Fan   Hydraulic Line   Hydraulic Pressure

N2   Compressor

Combustor

Control Surfaces
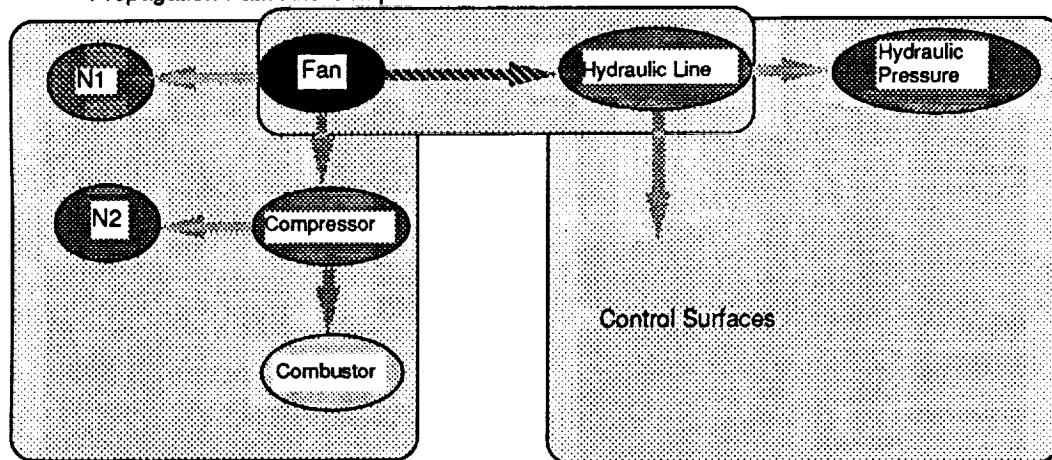
Propagation Type: Hybrid

Figure 5.4: Composed Hypothesis.

Using this heuristic reduces substantially the number of compositions examined. The generator successively examines each physical propagation hypothesis. It assumes that functional propagation could proceed from any component in the propagation path of a particular physical hypothesis, so the generator composes the physical hypothesis with the functional hypotheses from each affected component.

Extending Hybrid Hypotheses

When new symptoms arrive, and we have valid hybrid-propagation hypotheses, Draphys extends them in a similar manner as described for functional-propagation hypotheses. Draphys attempts to continue propagation from the components in the propagation path where propagation halted. In hybrid-propagation hypotheses, the functional-propagation portions of the hypotheses are extended first. If that does not account for the new symptoms, then the physical-propagation portions of the hypotheses are extended.

An example of an extension of the previously-shown hybrid hypothesis is depicted in Figure 5.5, when new symptoms in *EGT* and *EPR* are detected. Draphys attempts to functionally propagate from the combustor. It is able to extend the hypothesis to the turbines and to the *EGT* and *EPR* sensors. Since this accounts for all the new symptoms, Draphys stops extension of the hypothesis.

If extending the hybrid hypothesis in the manner described does not account for all the symptoms, then it would be treated as a multiple fault situation.

Hypothesis Composition

We claim that hypothesis composition is a rational, incremental, and reasonably efficient approach to explaining complex fault behavior for faults that propagate in multiple models. The composition is only done when a single model cannot account for all symptoms, but it uses hypotheses created when those single-model hypotheses were explored. Moreover, the composition is not done randomly, but rather uses a heuristic

HYPOTHESIS 1 OF 1

Current Symptoms:

> N1 Abnormal
>
> N2 Abnormal
>
> Hydraulic Pressure Abnormal
>
> EPR Abnormal
>
> EGT Abnormal

Fault Type: Single Fault

Propagation Path And Component Status:
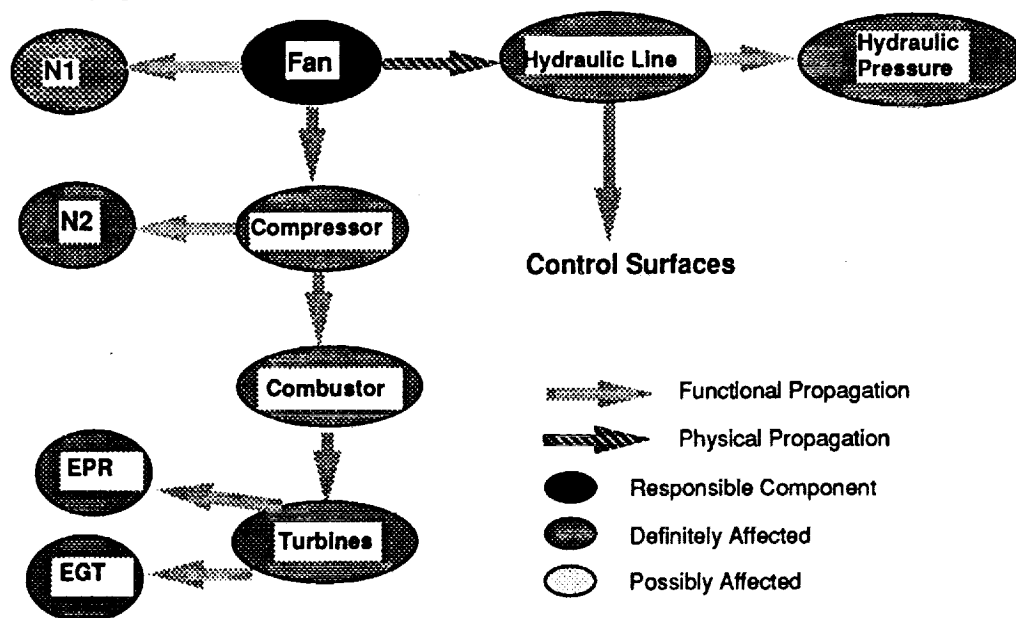
Propagation Type: Hybrid

Figure 5.5: Extended Hybrid-Propagation Hypothesis.

to limit the search for compositions. As we show in the next chapter in the evaluation of Draphys on actual accident cases, this heuristic applies in several cases.

Other Aspects of Fault Propagation

Other combinations of fault propagation behavior can occur, of course. Sometimes functional propagation results in physical propagation, which might happen when an oil leak results in bearings overheating, causing physical damage to adjacent components. Draphys currently does not implement such a heuristic.

If we want to provide more specific information about the physical propagation behavior, we could extend the current implementation by including knowledge about what is propagating and the associated behavioral differences. For example, a propagating physical object might be something rigid that is not likely to break up (like a fan blade) or something that disintegrates (like a bird ingested into the engine). Similarly, the type of "stuff" propagating might provide information about the distance (within a model) of propagation. For example, heat would propagate differently than a physical object or an electrical charge.

Providing such information might be beneficial by improving predictive capability and determination of fault severity levels. However, there are also costs associated with it. The physical structure model would have to include more information about the types of physical dependencies between components and thus the reasoning would become more complex. An improved corrective response may be generated, but this would increase the computational and representational complexity. The tradeoff between costs and benefits would have to be evaluated.

5.3   General Discussion

The approach described here provides a structure for relaxing assumptions associated with fault classes. The contribution to defining fault classes is the addition of specificity as a criterion. These fault classes provide a means of organizing the hypothesis space

into subspaces. We argue that this approach supports robust diagnostic problem solving, by allowing the diagnostic process to move to a new subspace when previous spaces fail to provide hypotheses. In the following sections, we discuss some general issues associated with the overall approach to designing fault classes in this manner. These issues include defining fault classes, the order in which fault classes are processed, and control of the diagnostic process. We also discuss constructive problem solving, because the hypothesis composition process is constructive in nature.

5.3.1 Defining Fault Classes

When we define a fault class in Draphys, we are identifying the set of hypotheses at a particular level of detail with a specified type of fault behavior. That is, we define fault classes according to what we know about the fault and its behavior. If one considers the diagnosis process as a search through a space where the elements of the space are fault hypotheses, these fault classes represent a partitioning of that hypothesis space.

The fault classes in Draphys are not totally independent. For example, at the specific level of status abstraction, the associational knowledge represents the set of known, functionally propagating faults. We say they are known faults because we know the specific qualitative symptoms associated with that fault. This fault class contains hypotheses which are more specific instances of the functional-propagation faults that represent a fault class at the higher status abstraction level. Interestingly, all the faults for which we had specific knowledge were functional-propagation faults, so there were no fault classes containing specific instances of the other single-fault classes. In Figure 5.6, we show the relationships among the fault classes diagnosed in the current implementation of Draphys.

5.3.2 Order of Fault Class Processing

When diagnosing a fault, Draphys checks for membership in each class by traversing the diagnosis taxonomy (shown in Figure 5.7) in a depth-first fashion. The ordering
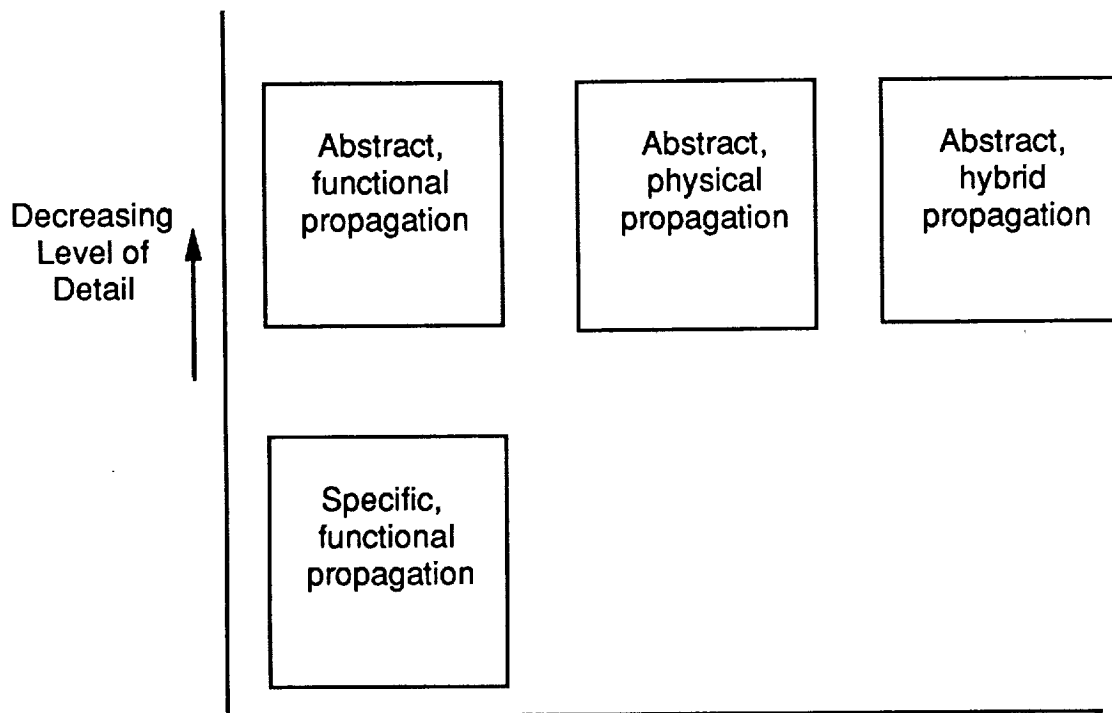
Figure 5.6: Relationships Among Single-Fault Classes in Draphys.

of fault classes is based primarily on fault likelihood, but it also simplifies the reasoning by allowing later fault classes (in particular, the hybrid-propagation fault class) to build on reasoning done in previously-explored fault classes. Functionally-propagating faults, whether described specifically or at the higher abstraction level, occur more often than faults involving physical damage. This seems reasonable, especially since functional propagation paths represent intended interactions, and physical propagation paths represent unintended interactions. Draphys checks for physical propagation (only) before checking for hybrid propagation in multiple models, because it is more likely that a fault is detected before it propagates in more than one model. The reasoning about hybrid-propagation hypotheses uses the primitive hypotheses created in the previously-explored fault classes. Lastly, multiple faults are least likely of all. In the current approach at least, if valid hypotheses can be identified in a fault class, we assume that there is no need to consider less likely fault classes. The less common fault classes can be examined later, if new information eliminates the current hypotheses. This assumes that all hypotheses in one class have the same likelihood compared to all hypotheses in another class, and that we only want the most likely hypotheses. This is a domain-specific choice, which may differ for another domain than the one under consideration.

We are not restricted to this order of processing, however. We can consider two aspects of the order of fault class processing, moving between levels of abstraction and moving among fault classes within a level of abstraction. In the first aspect, we start at the specific level and move up a level of abstraction when necessary. We could also first reason about the fault classes at the abstract level of reasoning and refine the resulting hypotheses according to our specific knowledge.

This abstract-to-specific approach has several advantages. It is a potentially efficient way of grouping the specific knowledge, so that the search process is more efficient. This is particularly true if there is a large amount of specific knowledge. If, however, the amount of specific knowledge is not large, or it can be processed very efficiently, the additional reasoning at the abstract level may not be worth the pruning advantage that
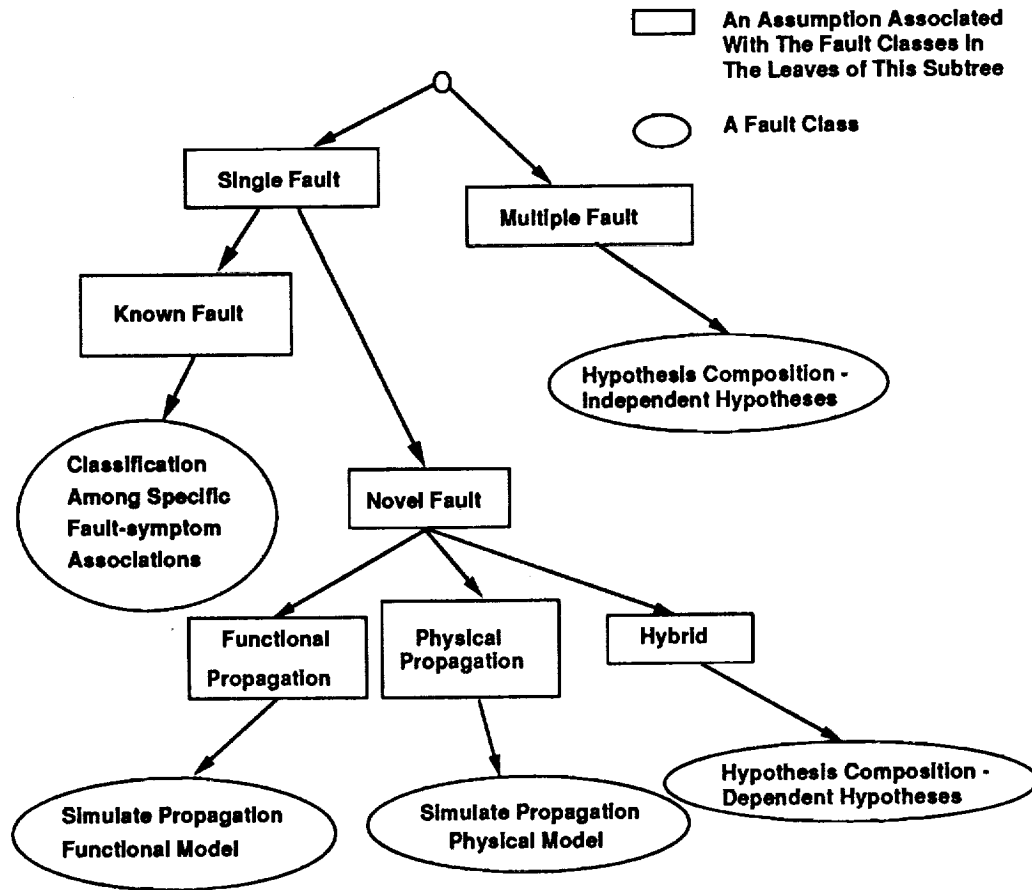
Figure 5.7: Diagnosis Taxonomy and Fault Classes in Draphys.

it gives.

Another potential advantage of an abstract-to-specific approach might occur when the time for diagnostic reasoning is limited. Since the reasoning at the abstract level is less detailed, it may take less time than reasoning at the specific level, especially if there is a large amount of specific fault knowledge. Therefore, it could generate useful information in less time than it takes to reason with specific knowledge. Indeed, if the time available for diagnosis ends before the specific level has been processed, useful diagnosis information has still been generated.

Moving between fault classes within an abstraction level is now ordered according to likelihood. This assumes that all faults in a particular class are more likely than any faults in a class considered later. Other means of ordering the fault classes are possible, such as ordering them by criticality. In Section 5.3.4, we briefly discuss one means of choosing a fault class to consider based on the localization process.

The order in which fault classes are processed can have an effect on the reasoning necessary to generate or identify hypotheses in that fault class. For example, the fault class at the higher level of status abstraction which looks for hybrid propagation performs composition of hypotheses that were constructed when prior fault classes were explored. That is, it composes hypotheses created in the functional-propagation and physical-propagation classes. By doing so, it takes advantage of computation done previously. If the order of fault class processing were changed, then the reasoning within that class may have to change, also.

5.3.3 Reasoning in the Component Hierarchies

Levels in the component hierarchies (both functional and physical) used for localization are levels of structural abstraction, where the primitive components are aggregated into subsystems at progressively higher levels. Many diagnostic approaches use such levels of structural abstraction to perform hierarchical diagnosis, generating hypotheses at each level in the component hierarchy and progressively refining them.

We chose not to do this in Draphys, because we cannot use this approach with hybrid propagation. To understand this point, consider the functional and physical component hierarchies. The rules for organizing subsystems are different for the two hierarchies, since one is based on functional relationships among components and one is based on physical location. Therefore, the subsystems in the hierarchies are quite different from each other, except at the lowest level, since the primitive components are the same in both hierarchies. Since the subsystems are not compatible between hierarchies, composition of hypotheses cannot be done above the primitive component level.

5.3.4 Control of the Diagnosis Process

As implemented in Draphys, the order of fault class processing is fixed. One possible future enhancement might be to make the control more opportunistic. We might modify the control so that fault classes are explored according to the current fault behavior, rather than in order of likelihood. The localization process may be useful as a means of indicating the fault behavior and, thus, the appropriate fault class. For example, if we can localize a fault to a single subsystem in the physical component hierarchy but not the functional component hierarchy, we can bypass the class of functionally propagating faults. To do this, the notion of what it means to localize in one hierarchy but not another would have to be clarified. Moreover, we cannot identify the fault class involving multiple models this way, since a single component hierarchy will not suffice for localization.

Updating hypotheses is another control issue. At present, Draphys updates hypotheses when new symptom information arrives. This new information is used to determine whether the old valid hypotheses are consistent with the new symptoms. Therefore, the diagnostic process is only triggered when new symptoms arrive. However, lack of symptoms can be useful in pruning hypotheses, when expected behavior does not occur. If this capability were added, the diagnostic process would have to be triggered more

often, rather than just when symptoms occur. At a minimum, the diagnostic process would be triggered at the end of an interval when an expected symptom should have occurred.

5.3.5 Constructive Problem Solving

As discussed in previous chapters, constructive problem solving is the assembly of a solution, in contrast to classification problem solving, which is the choice of a solution from a set of pre-enumerated solutions. This assembly might construct the solution as a subset of existing knowledge, as in diagnosis of functional propagation faults, where the generator creates subgraphs of the graph describing the functional model. The assembly might synthesize pieces of knowledge from various sources, such as the hypothesis composition in Draphys, or in constraint propagation.

Draphys uses simulation to focus the hypothesis assembly process, but there are certainly other ways to do it. The simulation-based constructive approach makes sense for operative diagnosis of physical systems such as aircraft subsystems, because we have the knowledge of expected propagation behavior.

When is it appropriate to use constructive versus classification problem solving? The constructive approach is better when the problem solving must be done often, but the solutions are not that different. This occurs in cases such as operative diagnosis, because the time-varying nature of fault propagation means that the hypotheses for two consecutive time snapshots often will only differ by the amount of additional abnormal behavior caused by the fault.

Formulating a problem as a constructive problem solving task is more appropriate when the pre-enumerated solutions would have much redundant information in them, and a single theory exists from which these solutions could be constructed (e.g., a model). This is analogous to the distinction made by Davis and King in [14], where they discuss how a production system is not very appropriate for the class of problems where unifying principles emphasize the similarities in seemingly different states. Similarly,

classification problem solving is not as appropriate because we have the model as the unifying theory for creating the hypotheses.

A constructive approach is better when a desired hypotheses comes from pieces of multiple sources of knowledge. An example of this is the hypothesis composition in Draphys. Neither model of the device is sufficient to provide an explanation of all the faulted system behavior, but the use of two models and heuristics to describe the type of interaction between them is enough to explain many fault situations.

5.3.6 Diagnosing Multiple Independent Faults

We described the classes of single fault. Another fault class to consider is that of multiple independent faults. Although this has not yet been completely implemented, a logical extension to Draphys' approach for diagnosing multiple independent faults could be done by composing hypotheses. This also need not be done randomly, although it would certainly be done differently from the single-fault, hybrid-propagation hypotheses. First, we need not check compositions of hypotheses that were checked previously as hybrid propagation. Second, we can look for two independent faults, then three, then four, and so on. Third, we can look for combinations based on the most likely single faults. For example, look for combinations of known faults before looking for combinations of novel faults. Another combination might be to look for the same type of failure in similar devices, such as the same kind of failure in multiple engines.[2]

5.4 Related Work

We discuss two categories of related research. The first is research using multiple models and in multiple classes of faults. The second is in diagnosis of multiple faults.

---

[2]Such a case arose in an L-1011, which had missing O-rings in all three engines. This case had a single cause, that of a maintenance error, but determining that single cause *in situ* is highly unlikely. However, problems having a single cause such as this might be similar in the same kind of component.

5.4.1   On Using Multiple Models

This need for multiple models when accounting for a particular fault's behavior makes it necessary to use more than one problem solving approach. Therefore, the question arises, how do we accommodate the multiple types of reasoning necessary? Simon [57] proposed a conceptual framework to account for the creative, problem finding aspect of the ill-structured problem-solving process. Within this framework, methods for the evocation and synthesis of problem structure alternate with problem solvers of familiar kinds in the progressive definition, refinement, and eventual solution of problems. He uses design as an example of such a process, stating: (op cit, p. 190)

> The whole design, then, begins to acquire structure by being decomposed into various problems of component design, and by evoking, as the design progresses, all kinds of requirements to be applied in testing the design of its components. During any short period of time, the architect will find himself working on a problem which, perhaps beginning in an ill-structured state, soon converts itself through evocation from memory into a well-structured problem.

Figure 5.8 illustrates the conceptual framework, showing the alternation between a problem solver working on a well-structured problem, and a recognition system continually modifying the problem space. Thus two major aspects of the reasoning process are the problem solver and the noticing-and-evoking mechanism. These two aspects permit the ill-structured problem to be viewed as a succession of well-structured problems.

We view diagnosis in a similar way. The notice-and-evoking mechanism would identify when a current fault situation is not among the set of specific diagnostic alternatives and move to a different partition of the problem space accordingly. This thesis explores moving among hypothesis subspaces of the diagnosis problem space to cope with faults that require different problem solving techniques and models. In short, we cope with ill-structured diagnosis problems by viewing diagnosis as conducting a search through a space of spaces.
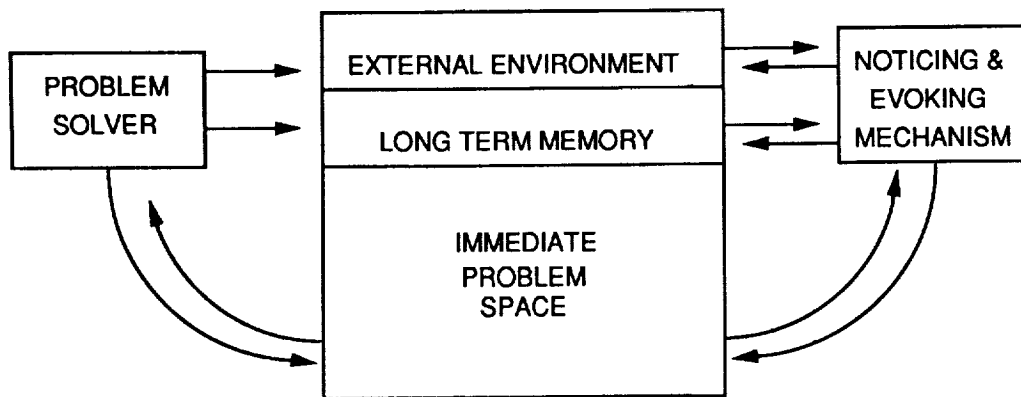
Figure 5.8: Schematic Diagram of a System for Ill-Structured Problems (from [57]).

5.4.2 Using Multiple Models for Diagnosis

The most related research on using multiple models, especially for diagnosis, is Davis' work [11]. He describes both functional and physical structure of a device. He also describes the advantages to having an ordered set of fault categories, which are similar to our fault classes. He discusses the notion of simplifying assumptions that correspond to each fault category, and that exploring the categories is a matter of making and retracting assumptions. He identified the adjacency principle as a mechanism for determining which model of the physical system to use; the correct model is one in which the paths of causal interaction are adjacent. Moreover, he assumes that there is no action at a distance, and finding the paths of interaction is a matter of finding the right type of adjacency.

Obviously, this thesis research applies the principles that Davis described. His fault categories are based on assumptions about adjacency. Draphys' fault classes are similar in that they are based on assumptions which are successively relaxed when they fail to provide a satisfactory hypothesis, and the order in which this occurs is determined by likelihood. However, they differ in the assumptions on which the classes are based. The

fault classes in Draphys are based on abstraction level and fault propagation behavior. The notion of fault propagation behavior is analogous to the adjacency assumption, because the kind of adjacency is the reason a fault propagates the way it does. Moreover, not all fault classes in Draphys are mutually exclusive. The specific, associational knowledge represents the class of specific, functionally propagating faults (it just so happened that all the known faults were functional propagation faults). The members of this class are specific instances of hypotheses in the higher level, functional-propagation fault class.

Concerning multiple models, Davis uses the example of a bridge fault to illustrate the use of both functional and physical knowledge. His program, HT, uses functional knowledge to generate candidates for the source of the problem. When HT determines that no single fault could account for the symptoms in the functional model, it looks for pairs of physically adjacent components among the candidates that might have a bridge between them. That basic idea, of looking at interaction via paths in a different model, is quite powerful. Draphys also implements that idea, but in a different way. Draphys systematically composes hypotheses based on knowledge about how faults propagate in different models. The composition process uses previously created hypotheses to incrementally create the composite hypotheses. The composite hypotheses can also be incrementally updated when new symptoms appear as the fault continues to propagate.

5.4.3   On Diagnosing Multiple Faults

Although not yet implemented, some obvious extensions to this research for diagnosis of multiple faults exist. We could look at combinations of single-fault hypotheses, using the approach implemented in GDE and described in [16], although that and other approaches for multiple-fault diagnosis do not at present reason about different classes of faults, and do not use different models. For example, they treat faults which propagate physically as multiple faults, because they do not have the model that would allow them to diagnose it as a single fault. However, GDE could be expanded to

accommodate such reasoning.

5.5 Limitations

Although this research has expanded the capability of existing diagnostic systems, there are still fault classes we cannot diagnose. These include intermittent faults, design faults, and so on. Moreover, we cannot rank the hypotheses that we have within a fault class. We also do not include any notion of time duration in our temporal reasoning; we have abstracted it away because we do not yet know how to represent it.

Chapter 6

Evaluation of the Fault Diagnosis Approach

This chapter includes two types of evaluation of the diagnostic approach. The first is an experimental evaluation of the implemented system on actual accident cases. The second is an analytical evaluation that includes a credit assignment analysis and a knowledge degradation analysis. In the analysis of credit assignment, we discuss the characteristics of the approach that are responsible for its success. We describe the constraints on the design of the diagnostic approach and guidelines for the design of the models. In the analysis of knowledge degradation, we describe the consequences of incompleteness or of elimination of types of knowledge in Draphys.

6.1 Experimental Evaluation

This section describes the experimental evaluation of the diagnostic approach on actual aircraft accident cases involving single engine faults. First, we discuss the experiment design and the reconstruction of the accident cases. The resulting diagnosis for each accident is then presented and discussed.

6.1.1 Approach

Eight official National Transportation Safety Board (NTSB) reports on accident cases were found involving single faults in turbine engines in commercial transport aircraft. These cases were collected at the beginning of this research. This set of cases was divided into two sets. One set of four cases was put aside for the experimental evaluation and was not examined for the design of Draphys. The other four, together with reports on engine-related problems that did not result in accidents, were used in Draphys' design.

When the prototype of Draphys was complete, a colleague who was unfamiliar with the design of the diagnosis system reconstructed all eight accident cases and generated the symptoms that occurred in the accidents.[1] Although this colleague was unfamiliar with the diagnostic process, he was familiar with the operation of the fault monitor which provides inputs to Draphys. He reconstructed the symptoms as though they were produced by the fault monitor. These reconstructed symptoms were presented to Draphys as input, and the resulting hypotheses are presented below.

Before discussing the results, some of the limitations of the reconstruction must be explicated. Numerical sensor data from the engine parameters was not available, so the symptoms were reconstructed based on the descriptions in the NTSB analysis of each accident. The symptoms in the report were usually those described by the flight crew, who normally will not detect deviations in sensor readings as soon as the fault monitor does. Therefore, the sequence of symptoms could not always be determined completely, because the flight crew only noticed a problem when several sensors were symptomatic. In several fault cases it appears that the sensors became symptomatic simultaneously, which is not very likely. However, the symptoms as described by the crew were used.

Another limitation on the reconstruction was because not all engine faults occurred in the same type of turbine engine. The design of Draphys is based on a Pratt and Whitney JT8D-7 engine, but many of the failures occurred in other models of turbine engines. The reconstruction process generated the symptoms as though the engine were the JT8D-7. The engines were similar enough *qualitatively* that this did not create a problem. For diagnosis, the main differences between engines were the sensors available. Several turbine engines have vibration sensors. However, the vibration sensors are notoriously inaccurate, and many airlines disable them. The different engines did not greatly affect the diagnosis because the diagnostic process depends on *qualitative* models and symptoms. If we were evaluating the fault monitor, which uses *quantitative* models, we would need numerical sensor data and we would need to examine fault cases for the

---

[1] I am indebted to Paul Schutte for reconstructing the accident cases and doing the initial evaluation as described in [54].

JT8D-7 only.

6.1.2 Results

In this section, we present the results of the experimental evaluation. Table 6.1 presents a summary of the hypotheses, without system status or propagation path, produced for the test cases used as test input. We discuss each test case individually, and its results; we then discuss the major points brought out by the analysis of the results.

Before we begin describing the results, we first must define a *successful* diagnosis. In this evaluation, we define a successful diagnosis to be one in which the correct hypothesis (the actual cause as identified by the NTSB) is among the set of valid hypotheses produced, and that the remaining hypotheses were such that they were a reasonable explanation of the situation, given the data available.

While this may not appear to be a particularly stringent definition, it is consistent with the goal of this thesis research. That is, we are concerned with producing useful diagnostic information, even in the presence of novel faults. Therefore, it was important to the goal of this research to be able to generate a correct hypothesis, even for a novel fault. If we generated a correct hypothesis, we considered that a success. Moreover, since we cannot rank hypotheses, we could not test Draphys' ability to choose *the* correct hypothesis. Also because of our goal, we made the design decision of preferring false positive hypotheses (hypotheses which said that a component had failed when it actually had not) to false negatives (not producing a hypothesis for a component which actually is at fault). In all the cases, Draphys never produced more than six valid hypotheses.

Note that both status abstraction levels were invoked in all cases, in order to test the diagnostic reasoning at each level. In each case, the lower abstraction level was invoked first. For evaluation purposes, even when this level produced hypotheses, the higher level also was invoked.

Table 6.1: Summary of Hypotheses (Without System Status) Produced by Accident Case Analysis

Case Description	Stage 1 Hypotheses	Stage 2 Hypotheses
1. Turbine Blade Separation	1. Turbine Blade Separation 2. Flameout	1. Fan 2. Compressor 3. Combustor 4. Turbine
2. Fan Failure	1. Turbine Blade Separation	1. Fan
3. Fan Failure	1. Turbine Blade Separation	1. Fan
4. Foreign Object Ingestion	none	1. Fan 2. Compressor 3. Combustor 4. Turbine
5. Massive Rain	1. Flameout 2. Turbine Blade Separation	1. Combustor 2. Turbine
6. Engine Separation	1. Fuel System Failure 2. Flameout	1. Engine - Fan
7. Turbine Disk Separation	1. Turbine Blade Separation	1. Combustor 2. Turbine
8. Bearing failure	1. Flameout 2. Turbine Blade Separation	1. Compressor

Case 1 - Turbine Blade Separation

On July 19, 1970, a United Airlines Boeing 737-222 (Flight 611) crashed shortly after taking off from the Philadelphia International Airport [30]. During takeoff, the number 1 engine failed. The captain thought that both engines were failing. Therefore, he decided to reject the takeoff and land the aircraft on the existing runway. The aircraft came to a stop past the end of the runway. The NTSB determined that a first stage turbine blade had failed in the number 1 engine which caused the engine to cease rotation. The number 2 engine was operable throughout the flight. The probable cause of the accident was determined to be the captain's inappropriate decision to reject the takeoff and land the aircraft based on his lack of understanding of the true state of the aircraft.

The first stage [2] of Draphys's diagnosis process produced two hypotheses, namely "Turbine Blade Separation" and "Flame-out" for engine number 1. The first hypothesis was correct. The second hypothesis was also correct even though flame out was caused by the turbine blade separation. The second stage produced four hypotheses, one for each major component of the engine, namely the fan, the compressor, the combustor and the turbine. Since all the engine sensors became symptomatic simultaneously, stage 2 could not distinguish one as the source of the problem. The stage 2 diagnosis contained one correct hypothesis and three false-positive hypotheses. It is improbable that all the engine sensors would have become symptomatic at once; however, this was the only information available from the NTSB report. If a more realistic simulation of this accident could have been produced, the second stage might have pruned some of the false-positive hypotheses.

A decision aid such as Draphys may have helped to avoid this accident. The accident occurred because the pilot was confused about which engine had failed. The interface design which has already been implemented in Draphys could have reduced

---

[2]Note that the first diagnosis stage refers to reasoning at the lower abstraction level, and the second diagnosis stage is the reasoning at the higher abstraction level.

the ambiguity between the two engines. The correct diagnosis of the turbine blade separation might not have been useful until the aircraft landed; however, the correct localization of the failure was crucial to the safety of the flight.

Case 2 - Fan Failure

On November 3, 1973, a National Airlines DC-10-10 (Flight 27) suffered an engine failure and made an emergency landing at Albuquerque International Airport [31]. The engine fan assembly of the number 3 engine disintegrated and its fragments penetrated the fuselage, the number 1 and number 2 engine nacelles, and the right wing area. The resultant damage caused the loss of certain electrical and hydraulic subsystems. The NTSB determined that the probable cause was the disintegration of the number 3 engine fan assembly as a result of an interaction between the fan blade tips and the fan case.

Stage 1 of Draphys produced a false-positive hypothesis of "Turbine Blade Separation." There is no fault in the set of fault-symptom associations which corresponds to a fan blade failure. "Turbine Blade Separation" was triggered because of certain similarities in the symptoms for "Turbine Blade Separation" and "Fan Failure." This problem could be corrected by extending the fault-symptom association set to handle fan failures. Before the stage 1 diagnosis, stage 2 of Draphys correctly hypothesized a problem in the engine fan. Stage 2 tracked the fault propagation into the hydraulic line. Were Draphys implemented for all aircraft subsystems, more of the propagation may have been evident.

This case raises an interesting point regarding physical propagation paths in the stage 2 model of the physical structure. All the physical propagation paths on the aircraft are stated explicitly within the stage 2 physical system model. As mentioned above, fragments of the number 3 engine (a wing mounted engine) penetrated the number 1 engine nacelle (mounted on the opposite wing). This physical propagation is

highly unlikely and probably would not have been explicitly modeled in a fully implemented system. Draphys would have treated any symptoms in engine 1 as resulting from a separate fault, totally unrelated to fault in engine 3. If every possible physical propagation path were stated in the model, the generation of hypotheses would be very extensive and therefore, very slow. Also, the pruning of hypotheses would be very difficult. Therefore, the practical option is to only state those propagation paths which are likely, and treat unlikely propagations as multiple faults.

Case 3 - Fan Failure

On January 31, 1981, a Northwest Airlines DC-10-40 (Flight 79) suffered an engine failure after departing from Dulles International Airport, Chantilly, Virginia [35]. The NTSB determined that the probable cause of the incident was the failure of a fan blade in the number 3 engine. The failure of the fan blade led to the inflight separation of the nose cowl assembly and the fan containment case.

Stage 1 of Draphys again produced a false-positive hypothesis of "Turbine Blade Separation." The symptom set needed to trigger "Turbine Blade Separation" is very small and therefore, this association is very sensitive. Stage 2 correctly named the fan as the responsible component, and showed propagation throughout the other engine components.

This case is similar to Case 2 above in that a fan failure triggered the "Turbine Blade Separation" hypothesis in stage 1. These two cases demonstrate the need to add a "Fan Failure" fault to the stage 1 fault symptom associations.

Case 4 - Foreign Object Ingestion

On November 12, 1975, an Overseas National Airways DC-10-30 (Flight 32) crashed while attempting to take off from John F. Kennedy International Airport, Jamaica, New York [32]. During the takeoff roll a large number of sea gulls rose from the runway and were ingested into the engine. The number 3 engine disintegrated. The takeoff

was rejected and the aircraft crashed off the end of the runway. The NTSB determined that the probable cause of the accident was the disintegration and subsequent fire in the number 3 engine when it ingested a large number of sea gulls.

There was no diagnosis produced from the first stage of Draphys even though one of the faults in the fault-symptom association is "Foreign Object Ingestion." The reason that this fault was not triggered was that the symptoms in the association correspond to a light ingestion in which the engine continues to operate during ingestion. In this test case however, the massive ingestion almost immediately destroyed the engine. Therefore, the symptoms were different, qualitatively and quantitatively, from a light ingestion. Stage 2 produced four separate hypotheses, naming each of the following as the responsible components: the fan, the compressor, the combustion section, or the turbine. The reason that all four of the major engine components were hypothesized as the responsible component is because symptoms occurred on all major engine sensors simultaneously. Stage 2 did not have enough information to prune the hypotheses.

There are several noteworthy points concerning this test case. One is that the same fault may manifest itself in different ways depending on the severity of the fault and the ambient conditions. Another point concerns the second stage of the diagnosis process and the fault monitoring stage. If the monitor could discern which sensor became symptomatic first, instead of reporting that they all became symptomatic at once, the second stage could reduce the number of hypotheses in its diagnosis. Finally, this test case demonstrates that there will always be failures which the system cannot completely diagnose; however, it does provide as much information to the flight crew as possible. In other words, the system provides a graceful degradation of information instead of simply providing no information, when it cannot completely determine the cause of the failure.

Case 5 - Massive Water Ingestion

On April 4, 1977 a Southern Airways DC-9 (Flight 242) crashed in New Hope, Georgia [33]. The aircraft had flown through heavy thunderstorms and had lost both engines. The crew attempted an emergency landing on a highway and crashed. The NTSB determined that massive water ingestion into the engines accompanied by thrust lever movement induced severe stalling in and major damage to the engine compressors. The NTSB determined that the aircraft might have been able to survive the weather had the flight crew not made significant movements in the thrust lever.

The first stage of Draphys hypothesized that the failure was either "Turbine Blade Separation" or "Flame-out" for both engines. This diagnosis was produced after both engines had failed. "Flame-out" was correct; "Turbine Blade Separation" was a false-positive hypothesis. Again, there was no association in the set of fault-symptom association which corresponded to "Massive Water Ingestion." Therefore, stage 1 could not correctly diagnose the original cause of this failure, although it diagnosed "flame-out" as one of the consequences. Stage 2, however, produced an earlier diagnosis of problems in either the combustor, turbine or the EPR sensor. This incorrect diagnosis was based on an early change in EPR which was believed to have occurred before a symptom in the N2 sensor.

This case identifies an excellent potential benefit of a fully implemented first stage for both the diagnosis system and recovery planner. The first stage of the diagnosis system could recognize the symptoms of "Massive Water Ingestion." The recovery planner could caution the flight crew not to make any significant movements in the throttle lever.

Case 6 - Engine Separation

On May 25, 1979 an American Airlines DC-10-10 (Flight 191) crashed into an open field northwest of Chicago-O'Hare International Airport [34]. During takeoff rotation, the left engine and pylon assembly, and about 3 feet of the leading edge of the left wing

separated from the aircraft. The aircraft began to roll to the left until the wings were past the vertical position. During the roll, the aircraft's nose pitched down below the horizon and crashed. The NTSB determined that the probable cause of this accident was the asymmetrical stall and the ensuing roll of the aircraft at a critical point during takeoff. This was caused by the uncommanded retraction of the left wing outboard leading edge slats and the loss of the stall warning and slat disagreement indication systems resulting from separation of the number 1 engine and pylon assembly. The NTSB determined that the accident would have been survivable had the flight crew known that the stall warning and the slat disagreement indication systems were inoperative.

This case was actually a multiple fault, in that multiple primitive components failed simultaneously. Since Draphys does not yet have multiple-fault diagnosis capability, it does not truly evaluate this particular case. This case is analyzed below, while taking this factor into consideration.

Both stages of Draphys produced diagnoses for this failure. The first stage produced two hypotheses, namely "Fuel System Failure" and "Flame-out." The hypothesis of "Flame-out" is most appropriate since the fault-symptom association dictionary does not include a fault of "Engine Loss." The second stage hypothesized a physical propagation from an engine failure (specifically the engine fan) to the hydraulic subsystem. This too was the correct diagnosis since the critical information needed by the crew was the loss of thrust and the propagation to the hydraulic system. The diagnosis hypothesized that the responsible component was the fan because the only component in the stage 2 model of our aircraft with a physical adjacency to the hydraulic line was the fan.

Again, the significance of this case is that the second stage recognized that the abnormal sensor readings from the hydraulic system sensors were not the result of a separate failure in the hydraulic system but a physical propagation from an engine failure. This demonstrates the concept of recognizing fault propagation between functionally unrelated aircraft systems. Were Draphys implemented for all the aircraft sensors, it probably could have determined that the stall warning system and the slat

disagreement indicator were possibly inoperative because of fault propagation from the engine separation. As indicated above, this information might have greatly increased the crew's ability to compensate for the failure and survive the failure.

This case highlights an important consideration. A fault which may be a single fault at a given level in the component hierarchy (as in the subsystems level here, where our diagnosis was successful) can be a multiple fault at the next level down in the component hierarchy. Therefore, identifying a fault as single or multiple fault may depend greatly on the definition of the primitive components.

Case 7 - Turbine Disk Separation

On September 22, 1981, an Air Florida Airlines DC-10-30CF (Flight 2198) suffered a failure in the number 3 engine during takeoff at Miami International Airport, Miami, Florida [36]. The takeoff was rejected. The engine disintegrated and the resultant debris damaged the right wing outboard leading edge slat. Components of the number 1 and number 3 hydraulic systems were also damaged by engine debris. The NTSB determined that the probable cause of the accident was presence of foreign material in the low pressure turbine cavity. The foreign material damaged connecting bolts in the engine, and when these bolts failed the low pressure turbine disk separated from the its rotor assembly, oversped, and burst.

Stage 1 of Draphys hypothesized this fault as "Turbine Blade Separation." This diagnosis was correct in the sense that all the blades and the disk itself separated. There is no fault for "Turbine Disk Separation" in the stage 1 fault-symptom association dictionary. The second stage diagnosis hypothesized either the turbine, combustor, or the EPR sensor as being the responsible component. The reason that the second stage could not solely identify the turbine as the responsible component is that there is no specific sensor monitoring the turbine; therefore, it must base its hypotheses on EPR and EGT information.

Case 8 - Bearing Failure

On September 22, 1981 an Eastern Airlines L-1011 (Flight 935) experienced a failure in the number 2 engine which forced an emergency landing at John F. Kennedy International Airport, Jamaica, New York [37]. Before the failure, abnormal sensor readings occurred in the N2 sensor of the number 2 engine. The pilot reduced the throttle on the number 2 engine and it returned to normal. Later, the engine disintegrated. The NTSB determined that the probable cause was the thermally induced degradation and consequent failure of the number 2 engine low pressure bearing because of inadequate lubrication. After the number 2 engine was lost, the A, B, and D hydraulic systems failed.

Stage 1 of Draphys hypothesized the problem as being either "Flame-out" or "Turbine Blade Separation." "Turbine Blade Separation" was incorrect. "Flame-out" was accurate after the failure. There is no fault for inadequate lubrication or engine overheat in the stage 1 set of fault-association. Stage 2 of Draphys' diagnosis system detected a problem with the number 2 engine before the engine failed. The hypothesis was that the compressor was malfunctioning. As time progressed, the second stage tracked the propagation of the failure into the combustor, the turbine, and the fan as well as the N1, N2, EGT, and EPR sensors. From the fan, stage two tracked a physical propagation to the hydraulic line which is near the fan. In reality, the hydraulic line was cut by a compressor blade. This hypothesis was produced because there is no physical link in the model between the compressor and the hydraulic line. Again this is a deficiency in the model which the second stage uses and not in the reasoning concept. In section 6.2.2, we discuss the general implications of missing or incomplete knowledge in the models. If there were a physical link in the model between the compressor and the hydraulic line, stage 2 would have hypothesized the correct physical propagation of the fault.

While the propagation path in the hypothesis from stage 2 may not be exactly correct, it does contain important information (i.e., the failure has propagated to the

hydraulic system). Moreover, the remainder of the hypothesis, identifying the responsible component and the system status, is correct.

6.1.3 Discussion

The implications of the results for both the first stage (the specific fault-symptom associations) and the second stage of diagnosis (the reasoning at the higher level of status abstraction) are discussed below. For each stage, we consider the false-positive hypotheses produced and the research areas that still need to be explored.

Concerning false-positive hypotheses, two types are possible. In the first type of false-positive hypothesis, the diagnosis process identifies a component as the source of the problem, and a corresponding system status, that cannot be ruled out based on the current symptoms. The second type of false positive is one that could be ruled out if the system were smarter. Draphys produced false-positive hypotheses of the first type, where each hypothesis was a reasonable possible explanation of the current symptoms, and none of the second type. That is, all the false-positive hypotheses were such that they were a reasonable explanation of the situation, given the data available, even though post-crash analysis ruled it out based on closer examination of the physical components.

The results from the test cases identified three areas of research and development for the first stage, containing the specific fault-symptom associations. First, the fault-symptom association dictionary needs to be expanded, and, as discussed in Chapter 3, a new representation considered. The second area involves a refinement of the first stage hypothesis validation process. As mentioned above, there were 4 incorrect hypotheses. Each of these hypotheses was a false-positive hypothesis of "Turbine Blade Separation." This particular fault-symptom association can be triggered by many different sequences of symptoms. At present, stage 1 validates a hypothesis if all the symptoms which are characteristic of that fault are present. However, the fault hypothesized does not have to account for all the symptoms present as in stage 2. If this second criteria were placed

on stage 1 hypotheses, "Turbine Blade Separation" (as it is currently characterized in the fault-symptom association) would not have been triggered as often as it was in these test cases. The third area involves refining the fault-symptom associations to be more clearly and carefully defined. The repeated hypotheses of "Turbine Blade Separation" indicate that the fault-symptom association may not accurately describe this fault.

Stage 2 of the diagnosis process, the reasoning at the higher level of status abstraction, was considered to be very successful, even though all aspects of the hypotheses were not always correct as determined by the NTSB. There were two factors which limited the diagnosis success. The first was that the model of the physical and functional structure had some limitation which prevented an accurate diagnosis. The second factor involved the reconstruction of the test cases for simulation. As mentioned earlier in discussing the limitations of the reconstruction, some information was absent from the NTSB reports. This information should normally be available from the Draphys monitor. This information could improve the hypotheses generated by the second stage of Draphys.

Concerning execution time, the computer code was not optimized in any way. It was executed on a Symbolics 3650, written in Common LISP and Flavors, and was interpreted, not compiled. The physical system model represented approximately 40 components and 100 interconnections (both the functional and physical dependencies). The execution time was not measured precisely, but seemed reasonably efficient. In all cases the diagnosis was done in less than ten seconds, and often took less than that.

This evaluation demonstrates that the Draphys concept shows promise for performing diagnosis of physical systems in operation. However, further development would be useful to enhance the present implementation. The implementation should be expanded to include other subsystems in the aircraft, since most failures are rarely confined to the engine and hydraulic subsystems. Also, the subsystems now included should be described in more detail to handle a greater complexity of failures. This increased depth could allow improved flight crew procedures for handling inflight failures. Another desirable enhancement is the capability to handle multiple failures. The test cases above

were all single failures; however, many accidents are the result of multiple failures.

6.2 Analytical Evaluation

This section first discusses why this diagnostic approach works, and what characteristics give it its power, particularly at the higher abstraction level. The cost of violating the constraints on these characteristics, and the cost of eliminating types of knowledge, are also discussed.

6.2.1 Credit Assignment

Why does Draphys work? The answer lies mainly in three aspects of the approach: the monitoring that identifies the symptoms, the simulation of fault propagation, and the system models used. The following section discusses why the success of the approach depends on these three aspects, and the consequences of not having them.

The Symptoms

The entire diagnostic reasoning approach depends on the symptoms identifying when the parameters become abnormal. By identifying when parameters values differ from their expected values, the propagation of the fault's effects can be identified. This differs substantially from the monitoring systems currently operational for aircraft systems (or many other systems as well). Current operational systems identify when a parameter exceeds its total operating range, which can be a broad range. In contrast, the symptoms produced by Draphys' fault monitor are identified as soon as the parameter differs from its expected value (with allowances for sensor noise, etc.), even if the parameter is still within its normal operating range. This detects abnormalities sooner than a monitor that waits for the signal to exceed its proper range.

Not having the ability to identify abnormal parameters as soon as they become abnormal may restrict the applicability of this diagnostic approach. The simulation of fault propagation depends on knowing when signals become abnormal. If the order in

which the signals exceed their operating range is the same order in which they become abnormal (which will not necessarily be the case), the diagnostic approach will still produce correct information, but it will not be as timely and the effects of the fault probably will have propagated extensively.

There are disadvantages in requiring the symptoms to satisfy this requirement. Determining when parameters differ from their expected value requires a model to calculate what those expected values are. It may be difficult to obtain a numerical simulation model for some systems. It may also be that in some systems, such as digital circuits, using the expected binary inputs and outputs will not provide the abnormality information necessary to track the propagation of the fault. For example, if one input to an AND gate is zero and the other input is potentially faulty, it is impossible to tell whether the effect of the fault has reached that far, because the output will be zero. Moreover, sensor noise and mismodeling are also concerns.

Simulation

This diagnostic approach within the higher abstraction level is considered to be a simulation-based approach to generating hypotheses because the fault propagation behavior is simulated to construct hypotheses. Other approaches could be taken to construct hypotheses, such as, tracing backwards along the dependency links to find a single "parent" in the graph that describes the physical system. However, this graph is complex, since the physical system it describes has feedback, among other behavioral complexities.

The simulation-based approach is parsimonious, in that it focuses attention quickly on the candidate elements needed to construct a hypothesis. By doing this, the search for constructed hypotheses is pruned in a very knowledge-intensive manner. Moreover, the simulation enables incremental updating of hypotheses. This aspect allows very efficient use of new information when it becomes available, and facilitates the pruning or confirmation of hypotheses.

The simulation-based approach has several disadvantages. One of these is that a model is necessary to do the simulation. As we will discuss in the next section, the models must be carefully defined. Not having a model defined appropriately for the class of faults under consideration may very well mean that the class cannot be diagnosed properly.

Another concern is that the lack of sensor information leads to ambiguity in the simulation, since the extent of the fault cannot always be accurately tracked. The result is ambiguity in both the system status and in number of hypotheses generated. This problem will arise with any approach, though, because of limited sensor information. However, the implication for the simulation-based approach is that the reasoning must take into account the relationships among the sensors and non-sensor components.

Another, more major concern is that the definition of a model used for simulation is inherently based on assumptions. Assumptions must be made to build a model at all. For example, the functional structure model used in Draphys assumes the directionality of functional interaction is as described. Therefore, if this assumption is violated, the model can no longer be used to accurately simulate fault propagation behavior for a class of faults that violate directionality. Davis points out and discusses the issue of assumptions and their relationship to choice of model in [11].

The Models

Because the diagnostic approach is simulation based, the models of the physical system used in the simulation process are critical to the success of the diagnosis. In the following discussion, the important characteristics of the models are described, together with the consequences of not possessing these characteristics.

The functional and physical structure models of the physical system being diagnosed are defined hierarchically. We first discuss the necessary characteristics of the model within a single level in the component hierarchy; in particular, the primitive component level. We then discuss the definition of the component hierarchies themselves, or the

manner in which lower level components must be organized into subsystems.

Choice of Components

The first characteristic we discuss here is the choice of components at the primitive component level. This addresses two issues: what a component is, and how detailed our model should be. First, what is a component? A component is a physical part or set of parts of the device under consideration. A device has any number of components, depending on how detailed a model is desired. We could conceivably model the device down to the molecular level, but that is not necessarily desirable or beneficial.

Occam's razor is appropriate for choosing a level of detail for the model. It is undesirable to model the device at a greater level of detail than that needed to diagnose the problem. How the device is divided into components is determined by (1) what components need to be identified as faulted when they break, (2) whether the components can be disambiguated with the sensor placement available, and (3) whether the components are necessary in the propagation path to determine the propagation of abnormal status.

We now expand on each of these determining factors. If it is important to identify when a particular component breaks because, say, it determines or differentiates the necessary corrective action, then this component should be included in the model. Hamscher [25] also discusses this, and identifies it as a principle for building models. If it is not important to identify when a particular part breaks, and none of the other factors apply, then the component either should not be included in the model or the model designer should consider aggregating it to the next higher level of detail. For example, consider the individual fan blades in the fan. It makes no difference when a particular fan blade breaks, it is only important to know that one of the blades broke. Therefore, Draphys does not model the individual fan blades.

Another motivation for not modeling the fan blades is that we do not have enough sensor information to identify individual blades. The cost of modeling at a greater

level of detail than we have sensor information to discriminate with is ambiguity in hypotheses. We will have additional hypotheses because we cannot tell them apart.

The third factor is whether the part of the physical system contributes towards tracking the propagation path. If a component does not satisfy any of the prior criteria, the model designer should determine whether it can be a factor in the branching of the propagation path. If a particular component is a branching point in the propagation path that enables identification of the propagation to other components, it must be included in the physical system model.

If we leave a part out of the physical system model because of the above factors, and it breaks and damages another part that is in the physical system, then it is quite possible that we will suspect the component that is modeled as being the responsible component. If the action associated with it is the appropriate one, whether that component was truly the source of the problem or merely the first one affected, then our goal for diagnosis is served. Our diagnostic system design choice of preferring false positive hypotheses over false negative hypotheses may lead to this as a consequence. If the diagnosis were not for operative systems, this design choice may not be appropriate.

Organization of Subsystems

The localization process as described depends on the subsystems being defined in a particular way because this grouping into subsystems is important to the pruning ability of the localization process. The purpose of the localization process is to exonerate sets of components by pruning whole branches in the component hierarchy. The subsystems must be defined so that available sensor information can exonerate an entire subsystem at once. For instance, it is desirable to be able to say that if no engine sensor readings are symptomatic, then that engine (and by extension, its subcomponents) cannot be the source of the fault.

To avoid excessive exoneration, the subsystems must be defined so that sensors "bound" the subsystems by measuring inputs and outputs. It is not usually possible
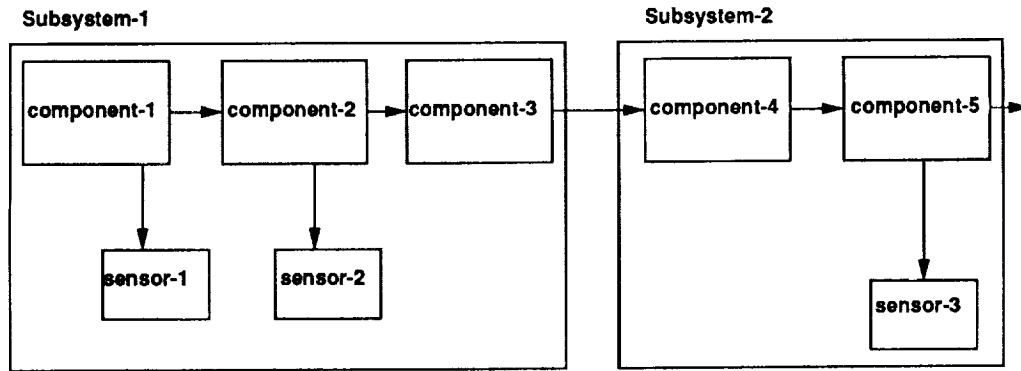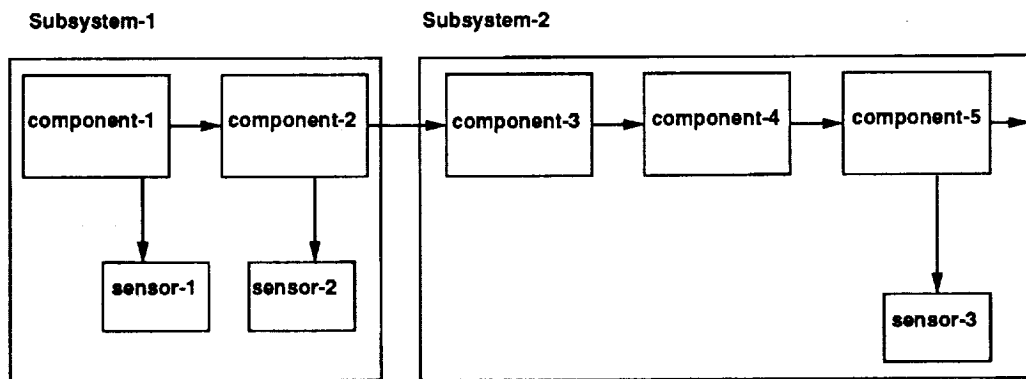
Figure 6.1: Inappropriate Subsystem Definition.

Figure 6.2: Appropriate Subsystem Definition.

to have sensor readings on all components, so the readings available must be used to judiciously organize the subsystems. For example, Figure 6.1 illustrates a subsystem definition that would not permit exoneration of subsystem-1, even when none of its sensors are symptomatic. If a symptom on sensor-3 came from the fault monitor, it may have been caused by a fault in component-3. Thus this subsystem organization will not support the localization process. Given the same sensors and components organized as shown in Figure 6.2, localization could be done.

Another consideration is that the boundary sensors must measure a parameter of the component or subsystem that can exonerate upstream components. To illustrate this point, consider the fuel temperature sensor in the fuel control subsystem. This sensor measures the temperature of the fuel in the fuel line. However, the function of the fuel line is to transmit fuel to the combustion section of the engine. Since this sensor could be normal when the fuel line has an abnormal operational status, it is not a good sensor to use for bounding a subsystem.

This last consideration highlights the point that sensors are not always placed in a manner to support diagnosis. However, one result of this thesis research is that we can identify some principles, as stated above, for placement of sensors to support the diagnostic process.

Representing Dependencies in the Models

Simulation of fault propagation follows the dependency links in the particular model being used; i.e., either physical or functional. Since the successful propagation (or lack thereof) is critical to the construction of hypotheses, these dependency relationships must be carefully and accurately defined.

The functional dependency links represent the normal, designed interaction among components in the physical system. When a fault occurs, the effect of the fault is expected to propagate along the normal paths of interaction in the functional model. The effect of the fault propagates because the output of the broken component is

abnormal and thus the input to dependent components is abnormal. However, for the input given, these dependent components operate properly, since they are not damaged. This is why the model of normal functional interaction can be used to represent faulted system behavior as well.

Functional dependency links between two components represents a *potential* interaction based on the normally operating system, but does not mean that there *will* be an interaction along this path. Whether a normal interaction occurs or not along a particular path may depend on specific parameter values that are unavailable at the level of detail of this model. Therefore, by representing all the potential paths of normal interaction, we can represent even those fault cases where the interaction is not intended under the current circumstances but which happens anyway. Moreover, because we simulate propagation of abnormal status, and do not simulate the specific qualitative values (such as high or low) of the system parameters, we can simulate fault behavior other than a localized failure of function. While the fault propagates along the normal functional paths of interaction, we can simulate it in this functional model.

6.2.2 Knowledge Degradation Analysis

The purpose of this analysis is to determine the cost of degradation in knowledge. To examine knowledge degradation in Draphys, we examine the consequences of not having a particular type of knowledge or of that knowledge being incomplete. This is straightforward in Draphys, because the types of knowledge are partitioned as the fault hypotheses are partitioned. Because types of knowledge are associated with fault classes, we examine each fault class individually.

In general, not having the type of knowledge associated with a fault class means that membership of faults in that class cannot be determined. We will examine, for each fault class, what the resulting diagnoses would be for faults in that class if the knowledge were not available.

The first class is that of specific known faults. The knowledge associated with this

class is the knowledge of fault-symptom associations for known, commonly occurring faults. If this knowledge were not available, then the specific fault hypotheses could not be generated. If all other knowledge is still in Draphys, the removal of this knowledge would result in hypotheses being generated at only the abstract level.

The second fault class is that of novel faults which propagate functionally. The knowledge associated with this class is the functional model of the physical system. Without this model, the propagation behavior of many faults could not be simulated; therefore, the fault hypotheses could not be constructed or pruned. If we had a physical structure model but not the functional structure model, then physical propagation would be identified but not the functional.

Similarly, when the physical model is unavailable, the physical propagation behavior cannot be identified. Without the model necessary to identify the physical propagation, this entire class of faults cannot be identified because the type of behavior cannot be recognized.

When a hybrid of physical and functional propagation occurs, both models are needed. In the fault class that includes all hybrid propagation, missing either form of knowledge (physical or functional) will result in the fault appearing to be a different fault class, that of multiple faults of the single propagation type. For instance, consider the fault shown earlier where physical propagation occurred from the fan to the hydraulic line and subsequent functional propagation occurred in each. Without a physical model, this fault appears to be two independent functional-propagation faults, one originating in the fan and one originating in the hydraulic line. However, other combinations may appear to be possible, such as a fan fault and a hydraulic pressure sensor fault.

Lastly, if we did not know how to diagnose multiple faults, then any fault whose behavior did not belong to any of the single fault classes would cause the diagnosis to fail. Providing multiple fault information is a useful capability, even if the fault was a single one. Even diagnosing multiple faults when only one occurred may still generate

appropriate responses.

Chapter 7

Conclusions

In this thesis, we have presented an approach to robust operative diagnosis of physical systems. In this chapter, we briefly summarize the contributions of the research and remaining open research questions.

7.1 Contributions

This thesis described and demonstrated a view of robust problem solving, especially graceful degradation, as reasoning at higher levels of abstraction (less detail) whenever the more detailed levels prove to be incomplete or inadequate. A form of abstraction was defined that applies this view to the problem of diagnosis. This form of abstraction, named status abstraction, represents the abstraction of the operational status of components in the physical system. We defined two levels of status abstraction. At the higher level, we presented a graph representation that describes the real-world physical system. We demonstrated an incremental, constructive approach to manipulating this graph representation that supports certain characteristics of operative diagnosis. That is, we showed the suitability of the constructive approach for diagnosing fault propagation behavior over time, and for sometimes diagnosing systems with feedback. We also showed a way to represent different semantics in the same type of graph representation to characterize different types of fault propagation behavior (physical and functional, and their combination). We demonstrated an approach that treats these different behaviors as different fault classes, and our approach moves to other classes when previous classes fail to generate suitable hypotheses.

Each of these contributions is described below, beginning with a description of the

problem of operative diagnosis.

7.1.1 Operative Diagnosis

Operative diagnosis, or diagnosis of physical systems in operation, differs from diagnosis of non-operating systems (such as maintenance diagnostics) in the following ways: the information a hypothesis must contain, dynamic fault propagation behavior, and limited testing for additional information. In operative diagnosis, the diagnosis is done to facilitate continued operation of the system under consideration. In maintenance diagnosis, however, the purpose is to determine which part to fix or replace. This distinction changes the information that a hypothesis must contain. In operative diagnosis, the symptoms or effects of the fault often must be treated in addition to the initial failure, so knowing the source of the fault alone is not always enough. To facilitate continued, safe operation of the device, the diagnosis must identify the cause and the effects of the fault. These effects, or system status, must include dependent failures and must identify components which are working properly but are receiving incorrect inputs.

In an operating physical system, the effect of the fault propagates and the set of symptoms may change as time (and the fault) progresses. This reflects the characteristic of non-zero-time propagation, where not all effects of a fault happen immediately. In operative diagnosis, the propagation often will still be occurring while the diagnosis is performed. In maintenance, the diagnosis is usually done after all propagation has taken place.

7.1.2 Robustness

Robustness in problem solving process is a capability that humans exhibit, and a highly desirable one for knowledge-based systems to exhibit. This thesis described an approach to improving robustness, particularly graceful degradation, in knowledge-based diagnosis through status abstraction.

This research presented and demonstrated the idea that graceful degradation can

be achieved in a structured way by using abstraction. The basic idea is that graceful degradation is not achieved by simply exploring whether something is known about faults, but at what level of detail is it known. Knowledge at different levels of specificity can provide different fault coverage. Increasing fault coverage is achieved at the cost of degrading specificity.

For diagnosis, status abstraction of hypotheses is particularly useful when corrective actions are associated with the general fault categories represented by the abstract hypotheses. However, even when these corrective actions are absent, the localization of the fault is still useful. Thus, the approach of using different status abstraction levels for diagnosing novel faults is appropriate when specific hypotheses are most desirable, but abstract hypotheses are better than nothing. Moreover, some known faults (such as physical damage) are more appropriately represented at the higher level of abstraction. This is the case when more specific hypotheses do not improve ability to take remedial action or the increase in number of specific hypotheses would inhibit their timely retrieval. The key to success is using a type of abstraction where the higher level information has some utility.

7.1.3 Incremental Hypothesis Construction

At the higher level of status abstraction, we presented a graph-based representation that describes the real-world physical system. The representation is particularly important because it supports reasoning about very complex physical system in a fairly simple way. The representation captures important characteristics about the physical system (i.e., fault propagation paths), that enable the diagnostic reasoning to be useful. To use this representation, we presented and demonstrated an incremental, constructive approach to producing fault hypotheses by simulating fault propagation behavior. This fault propagation behavior can be used to discriminate hypotheses, particularly when symptoms change over time. However, this means that it is desirable for the detection process to identify when sensor readings become abnormal, not just when they exceed

the normal operating range. This incremental construction process is important for efficiently reasoning about new symptoms that arrive as the fault continues to propagate, since it is desirable to build on computations done in previous time steps. We demonstrated this approach for operative fault diagnosis.

7.1.4 Using Multiple Models

Efficient reasoning with multiple models to explain complex fault behavior is a contribution of this research. The mechanism for using functional and physical models that are represented in the same graph representation is hypothesis composition, which is also a constructive process. Individual hypotheses that explain a portion of the fault propagation within a single model are composed in an efficient manner to explain total behavior. It is efficient because it builds on previously-created hypotheses and the composition process uses heuristics about physically realizable combinations to reduce search.

The constructive nature of the diagnostic reasoning is particularly useful for operative diagnosis in the chosen domain. In general, the constructive approach is appropriate when the problem solving must be done often, but the solutions are not that different. This occurs in cases such as operative diagnosis, because the time-varying nature of fault propagation means that the hypotheses for two consecutive time snapshots often will only differ by the amount of additional abnormal behavior caused by the fault.

Formulating a problem as a constructive problem solving task is more appropriate when the pre-enumerated solutions have much redundant information in them, and a single theory exists from which these solutions could be constructed (e.g., a model). Also, a constructive approach is better when a desired hypothesis comes from multiple sources of knowledge. An example of this is the hypothesis composition in Draphys. Neither model of the device is sufficient to provide an explanation of all the faulted system behavior, but the use of two models and heuristics to describe the type of interaction between them is enough to explain many fault situations.

We identified some knowledge engineering guidelines for these models. These guidelines have implications for sensor placement, and are intended to be useful in the design of new physical systems.

7.1.5 Fault Classes

We showed that fault classes can be defined by level of abstraction and by knowledge about fault propagation behavior. We defined four single-fault classes: (1) specific, functionally propagating faults (this class is the one in which Draphys groups known faults in fault-symptom associations); (2) abstract, functional-propagation faults; (3) abstract, physical-propagation faults; (4) abstract, hybrid-propagation faults (these hypotheses include both physical and functional propagation). Each of these classes is explored in turn when the previous ones fail to diagnose the current symptoms. Davis defined the notion of adjacency as a means of defining fault classes; we extend that adding abstraction level to the definition of a fault class. These fault classes partition the hypothesis space into subspaces, and supports a structured way of retreating from subspace to subspace when assumptions associated with prior spaces are shown to be wrong. This approach supports efficient and rational management of the hypothesis space. In short, we treat diagnosis as searching a space of spaces.

We consider this partitioning of the hypothesis space to be a taxonomy for diagnostic problem solving, since it associates assumptions about the fault behavior with the appropriate diagnostic problem solving techniques and physical system models. The taxonomy that is implemented in Draphys is not exhaustive, but does provide a framework for integrating new problem solving techniques into the overall diagnostic process.

The partitioning process is applicable to more than diagnosis. Dividing the search space into classes that require different problem solving techniques and models is a foundation for an intelligent problem solver.

7.2 Open Problems

Several open research issues remain. Some were previously existing research issues; some arose as a result of this research. We present them in the following order: limitations within the approach itself; fault classes not included here; and other remaining issues.

7.2.1 Limitations Within the Approach

Within the diagnostic approach there are open research issues associated with the two levels of abstractions in Draphys. Associated with the specific abstraction level, we have the issues of temporal grain, causal reasoning, extent of propagation, and identifying the level of fault severity. Associated with both abstraction levels, the issues include reasoning with uncertainty, temporal duration, testing for additional information, control of the diagnostic process, and fault masking.

Temporal Grain

The issue of temporal grain is somewhat subtle. As it currently exists, the associational reasoning in Draphys is based on the diagnostic reasoning described by domain experts, who were predominantly pilots. The pilots monitor the aircraft systems based on sensor information presented on the gauges and dials available in the cockpit, in a manner very similar to operators of other complex process control systems. Therefore, the monitoring of the sensors is dependent on the resolution of those gauges and dials. Because the resolution of these devices is not great, the pilots cannot see small distinctions on the guage, it takes them too much time, or it is too hard to do. Because of these human perceptual limitations, they do not depend on instantaneous detection of aberrant measurements, or transient measurements. Instead, they reason about behavior over time periods that may be short but are not instantaneous. They seem to ignore transients and only reason about steady state behavior over a period of time, albeit a short period. However, the symptoms generated by the fault monitor for input to Draphys are instantaneous. Thus Draphys has more detailed information about the

fault's symptoms than the expert pilots can provide. It is desirable to take advantage of these more detailed symptoms, and an open issue is how to acquire knowledge to do so.

Causal Reasoning at the Specific Status Abstraction Level

Since the specific level of abstraction reasons about specific malfunctions of particular components, there is a need to reason about how one of the malfunctions may cause another. Although one initial malfunction occurred, the propagation of its effects may cause another malfunction. To explain the fault's behavior at the specific level, knowledge of the causal relationships among malfunctions is desirable. Draphys does not currently contain such knowledge.

Levels of Severity

One type of information that Draphys does not provide is an indication of the level of severity of the fault's effect on the component. For example, we might describe an eroded compressor blade. It is desirable to describe the compressor's operational status as, for example, "operating but at a degraded level." (Another operational status might be "totally nonoperational.") Whether the fault severity levels are provided to the human operating the physical system would be determined by the human's information requirements in performing the diagnostic task.

Reasoning With Uncertainty

The current implementation does not include any explicit reasoning with uncertainty (other than the ordering of the fault classes). Therefore, individual hypotheses cannot be ordered or ranked according to their likelihood compared with other hypotheses within a fault class. It may be challenging to include reasoning with uncertainty because of the dynamic behavior and because the likelihood of a particular hypothesis is context dependent. An example of the former is, when new symptoms are detected that are

consistent with a current hypotheses, how do we update the certainty of that hypothesis to reflect this? The latter concern can be illustrated by the example of bird ingestion. This hypothesis is quite likely if the aircraft is taking off, but is highly unlikely if the aircraft is cruising at 30,000 feet. Current technology for reasoning with uncertainty has not yet been applied in Draphys.

Temporal Duration

At present, Draphys does not represent the temporal duration of symptoms in its knowledge. Given two faults with the same initial symptoms, the length of time that the symptoms last may allow elimination of one of the hypotheses. However, inclusion of this information in either the fault-symptom associations or the models would be difficult, for several reasons. First, the knowledge of the symptom duration would have to describe a range of time, because the duration of a particular symptom will probably vary, depending on factors such as the severity of the fault. Moreover, this type of duration information may be very difficult to obtain. Currently the duration of symptoms is ignored in both the abstraction levels, and only the temporal relationships among entire symptom intervals are used in the diagnostic reasoning. Merely the existence of new symptoms is used to prune hypotheses, although the lack of symptoms could potentially be used as well. To perform the latter, however, the diagnostic reasoning would need to know how long to wait before pruning a hypothesis whose expected future symptoms did not occur.

Testing for Additional Information

We mentioned earlier that one of the constraints on operative diagnosis is limited testing. However, we can still do some testing. We can use the dynamic response of the device to the operator's control inputs as additional information for diagnosis. For example, the pilot could move the throttle to distinguish between a fan fault and an N_1 sensor failure. Similarly, if we suspect a pump failure in the fuel system which has two

pumps in line, we could turn one off to isolate the faulty pump.

This is a desirable capability for the diagnosis system to have, and it would represent a significant enhancement. Research on incorporating actions into qualitative reasoning is being explored [21].

Control of the Diagnostic Process

Control of the diagnostic process is done implicitly in Draphys. A more opportunistic approach may be beneficial. If we were to incorporate an opportunistic control mechanism, we would have to consider several control issues. These include initiation of the diagnostic process, moving between levels and types of abstraction, and changing the fault class under consideration.

The diagnostic process is initiated whenever the fault monitor detects symptoms, or the symptoms change. This assumes that the symptom appearance triggers the need for a diagnosis. However, if we have several competing hypotheses, the *lack* of symptoms may be as important to the hypothesis pruning process as their *presence*.

At present, Draphys moves from the specific status abstraction level to the higher level when the specific level cannot produce a hypothesis. This reflects the diagnostic reasoning described by the human experts. However, reversing the order in which abstraction levels are explored may have advantages. Determining when to change levels of abstraction and which direction to move are open research questions. Similarly, moving between abstractions is another issue. The component hierarchies represent a type of structural abstraction based on aggregating components according to functional or physical groupings. The status abstraction used for hypothesis construction is another. Coordination between these two types of abstraction and moving among levels are major open control research issues. Moving between fault classes, and making and retracting the assumptions associated with each, are also control issues that are still open questions.

Fault Masking

One assumption made in the design of Draphys was that we could detect when sensor readings are abnormal. If this assumption is violated, then faults are masked, because Draphys would not identify a component as abnormal unless some abnormal sensor reading gave evidence to that belief.

7.2.2 Other Fault Classes

The fault classes as defined in this thesis are not exhaustive, but the diagnosis taxonomy in which they are organized does provide a framework for integrating new fault classes. For example, we assumed that the there were no intermittent faults, which may define a subclass under one of the existing classes (e.g., abstract functional propagation faults).

Intermittent Faults

There are several types of intermittency. This term is used to describe faults that occur only under certain circumstances, in which it is often necessary to determine what those circumstances are in order to diagnose the fault. The term is also used to describe faults that only appear occasionally, even under the same set of circumstances; e.g., a loose connection. "Intermittent fault" might also be used to describe a malfunction that occurs, then the device returns to normal operation after some action or occurrence takes place, e.g., some corrective action by the operator.

Each one of these types of intermittency must be diagnosed differently. Draphys does not currently handle any type of intermittent fault; indeed, the incremental updating of hypotheses assumes that previously abnormal components remain abnormal. The reasoning process would have to be modified to accommodate such fault behavior.

Violated Directionality

Draphys currently assumes that the directionality modeled in the physical system models are correct. If this directionality is violated, the models cannot be used to simulate

the fault propagation behavior. Adding a fault class that looks for such occurrences is possible, although the search process would be extensive. This is a particularly difficult issue for that reason. In the domain of aircraft subsystems, we did not encounter any accident or incident cases where this directionality assumption was violated. In other domains, it may be a more critical issue.

Design Errors and Design Knowledge

Design errors are another fault class that Draphys is not designed to diagnose. We assume that the design of the device is correct, in that it satisfies the requirements it is supposed to satisfy. If the design is not correct, but we have design models that accurately reflect that design, we would never detect that an error had occurred in the case where the system did not satisfy its design requirements. Similarly, we would not be able to detect an operator error, because the monitor only detects abnormalities compared with the operator inputs and the current conditions.

We also assume that our design knowledge, in the physical system models, is correct. There is no mechanism in Draphys to overcome modeling inaccuracies, although the fault monitor does have some tolerance for slight numerical inaccuracies.

7.2.3 Other Research Issues

Operator Actions

Draphys does not explicitly reason about actions taken by the human operator to recover from the fault. Since the fault monitor identifies sensor readings as symptomatic compared with their expected value based on operator input, the only circumstances that would invalidate the diagnostic reasoning at the abstract level would be if the operator action corrected the problem, and the abnormal sensor readings returned to normal. This would affect the diagnostic reasoning because it violates the assumption that abnormal components remain abnormal. Therefore, relaxing the assumption to provide reasoning about such circumstances would be a useful extension.

At the specific level, however, the operator actions have more impact. The specific associational rules assume no operator actions. Thus, reasoning about operator actions would be even more useful at this level to accommodate situations that we can expect to occur. Moreover, if we recommend corrective actions and the operator performs them, we need to monitor their effectiveness.

Determining Corrective Actions - Recovery Planning

Before this diagnostic approach can be useful in an operational environment, it should be coupled with a process that takes the diagnostic information and determines the appropriate corrective actions. These must include both short-term actions, and long-term actions. In the aircraft domain, the short-term actions correspond to the immediate control inputs made to stabilize or minimize the fault's effects. The long-term action might be to modify the flight plan to accommodate the reduced functionality of the airplane, possibly including a new destination.

Short-term actions will be generally be procedures that the operator should follow. Some of the issues that will arise include: combining procedures necessary for separate malfunctions, or for different subsystems; constructing procedures for novel faults, and monitoring the execution of these procedures for effectiveness.

User Interfaces

Because there is a human operator involved, the presentation of diagnostic information in an effective, comprehensible form must be done. The information should include the hypotheses, system status, and potentially an explanation of the diagnostic process. However, presentation of this information raises many human factors issues than cannot be ignored if we are to achieve an effective total human/machine system. The presentation media (e.g., voice output, CRT display) and the presentation format (e.g., pictures versus text) are critical issues.

Another aspect of the user interface is to solicit information from the operator.

Humans have sensors that the diagnostic system cannot use. The human can hear and see things, thereby obtaining data unavailable otherwise. How does the diagnostic reasoning solicit and use this information? The interaction between the two involves several human factors issues. Does the operator enter the information via voice input? Other possibilities include keyboard and touch panel. How the diagnostic system queries the operator involves similar questions.

7.3 Human/Machine Performance

We come to one final issue. Why do we want to automate the type of reasoning explored in this thesis? One of our long-term goals is to maximize performance of the human/machine system as a total system. This brings up some conflicting issues in building an intelligent problem solver (IPS) on a computer. On the one hand, we would like our IPS to be consistent with human cognitive processes, to facilitate the communication between them. However, we do not want to restrict ourselves to human reasoning. We should intentionally design our system to help overcome the limitations of the human. If this is true, then modeling an expert might not be appropriate. In the diagnosis process, diagnosing physical propagation is an example of such a limitation. Apparently humans have difficulty identifying the physical damage that a fault can cause. Yet, we want to have the diagnosis system communicate the reasoning in a manner that is comprehensible to the human. An automated diagnostic system such as Draphys has the potential to augment the human and improve human/machine performance.

7.4 Concluding Remarks

In conclusion, we have explored two ideas for increasing robust diagnosis of physical systems in operation: the use of status abstraction and definition of fault classes based on abstraction level and fault propagation behavior. This definition of the fault classes partitions the hypothesis space into subspaces, and provides a structured way of falling

back from one subspace to another when the fault cannot be diagnosed. We order the processing of subspaces based on likelihood, and less likely fault classes are not explored if more likely fault classes can account for the fault. We explored two additional ideas for efficiently managing the hypothesis space. The first is incremental hypothesis construction, which takes advantage of previously computed hypotheses when new symptoms appear. The second is the use of hypothesis composition when multiple physical-system models are needed to explain complex fault behavior. These ideas were implemented and demonstrated for operative diagnosis of aircraft subsystems. We discussed their advantages and limitations, and open research issues.

References

[1] K. Abbott, P. Schutte, M. Palmer, and W. Ricks. Faultfinder: a diagnostic expert system with graceful degradation for onboard aircraft applications. In *14th International Symposium on Aircraft Integrated Monitoring Systems*, Friedrichshafen, West Germany, September 1987.

[2] A. Abu-Hanna and Y. Gold. An integrated, deep-shallow expert system for multi-level diagnosis of dynamic systems. In *International Conference on Artificial Intelligence in Engineering*, 1988.

[3] D. Allen. *Investigation of Display Issues Relevant to the Presentation of Aircraft Fault Information*. PhD thesis, Old Dominion University, April 1988.

[4] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23, 1984.

[5] *The Aircraft Gas Turbine Engine and Its Operation*. United Aircraft Corporation, East Hartford, Conn., November 1960. PWA Oper. Instr. 200.

[6] J. S. Brown, R. Burton, and J. de Kleer. Pedagogical, natural language, and knowledge engineering issues in SOPHIE I, II, and III. In D. Sleeman and J. S. Brown, editors, *Intelligent Tutoring Systems*, Academic Press, 1982.

[7] B. Chandrasekaren and S. Mittal. Deep versus compiled knowledge approaches in diagnostic problem-solving. *International Journal on Man-Machine Studies*, 19:425–436, 1983.

[8] H. Chang, E. Manning, and G. Metze. *Fault Diagnosis of Digital Systems*. John Wiley and Sons, 1970.

[9] W. Clancey. Classification problem solving. In *National Conference on Artificial Intelligence*, 1984.

[10] L. Darden. Viewing history of science as compiled hindsight. In *National Conference on Artificial Intelligence*, 1986.

[11] Randall Davis. Diagnostic reasoning based on structure and behavior. In Daniel G. Bobrow, editor, *Qualitative Reasoning About Physical Systems*, The MIT Press, 1985.

[12] Randall Davis. Robustness and transparency in intelligent systems. In *Human Factors in Automated and Space Systems*, pages 211–233, National Research Council, Washington, D.C., 1987.

[13] Randall Davis and Walter Hamscher. Model-based reasoning: troubleshooting. In Howard Shrobe and the American Association for Artificial Intelligence, editors, *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., 1988.

[14] Randall Davis and Jonathon King. The origin of rule-based systems in AI. In Bruce Buchanan and Edward Shortliffe, editors, *Rule-Based Expert Systems*, Addison-WesleyPublishing Company, 1985.

[15] J. de Kleer and J. S. Brown. Assumptions and ambiguities in mechanistic mental models. In D. Gentner and A. Stevens, editors, *Mental Models*, Lawrence Erlbaum Associates, New Jersey, 1983.

[16] J. de Kleer and B. Williams. Diagnosis of multiple faults. *Artificial Intelligence*, 32, 1987.

[17] R. Doyle, S. Sellers, and D. Atkinson. A focused, context-sensitive approach to monitoring. In *Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[18] D. Dvorak and B. Kuipers. Model-based monitoring of dynamic systems. In *Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[19] Daniel L. Dvorak. *Expert Systems for Monitoring and Control*. Technical Report AI87-55, University of Texas at Austin Artificial Intelligence Laboratory, 1987.

[20] P. K. Fink and J. C. Lusth. Expert systems and diagnostic expertise in the mechanical and electrical domains. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(3), 1987.

[21] Kenneth Forbus. Introducing actions into qualitative simulation. In *Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[22] Michael R. Genesereth. The use of design descriptions in automated diagnosis. In Daniel G. Bobrow, editor, *Qualitative Reasoning About Physical Systems*, The MIT Press, 1985.

[23] Matthew Ginsberg. Counterfactuals. *Artificial Intelligence*, 30, 1986.

[24] T. Hamilton, D. Simmons, and R. Carlson. HELIX: an engine monitoring system. In *41st Mechanical Failure Preventions Group Symposium*, Oct 1986.

[25] Walter C. Hamscher. *Model-Based Troubleshooting of Digital Systems*. PhD thesis, Massachusetts Institute of Technology, 1988.

[26] Walter C. Hamscher and Randall Davis. *Issues in Model-Based Troubleshooting*. Technical Report Memo 893, MIT Artificial Intelligence Laboratory, 1987.

[27] E. Hudlicka, K. Corker, N. Cramer, D. Young, and S. Baron. Intelligent situation assessment and response aiding in flight emergencies. In *AIAA Computers in Aerospace Conference*, 1989.

[28] D. Lenat and E. Feigenbaum. On the thresholds of knowledge. In *Tenth International Joint Conference on Artificial Intelligence*, 1987.

[29] R. Milne. Fault diagnosis through responsibility. In *Ninth International Joint Conference on Artificial Intelligence*, 1985.

[30] *Aircraft Accident Report: United Airlines, Inc., Boeing 737-222, N9005U, Philadelphia International Airport, Philadelphia, Pennsylvania, July, 19, 1970*. National Transportation Safety Board. NTSB-AAR-72-9.

[31] *Aircraft Accident Report: National Airlines, Inc., DC-10-10, N60NA, Near Albuquerque, New Mexico, November 3, 1973*. National Transportation Safety Board. NTSB-AAR-75-2.

[32] *Aircraft Accident Report: Overseas National Airways, Inc., Douglas DC-10-30, N1032F, John F. Kennedy International Airport, Jamaica, New York, November 12, 1975*. National Transportation Safety Board. NTSB-AAR-76-19.

[33] *Aircraft Accident Report: Southern Airways Inc., DC-9-31, N1335U, New Hope, Georgia, April 4, 1977*. National Transportation Safety Board. NTSB-AAR-78-3.

[34] *Aircraft Accident Report: American Airlines Inc., DC-10-10, N110AA, Chicago-O'Hare International Airport, Chicago, Illinois, May 25, 1979*. National Transportation Safety Board. NTSB-AAR-79-17.

[35] *Aircraft Incident Report: Northwest Airlines 79, McDonnell Douglas DC-10-40, N143US, Leesburg, Virginia, January 31, 1981*. National Transportation Safety Board. NTSB-AAR-81-10.

[36] *Aircraft Accident Report: Air Florida Airlines, Inc., McDonnell-Douglas, Inc. DC-10-30CF, N101TV, Miami, Florida, September 22, 1981*. National Transportation Safety Board. NTSB-AAR-82-3.

[37] *Aircraft Accident Report: Eastern Airlines Flight 935, Lockheed L-1011-384, N309EA, Near Colts Neck, New Jersey, September 22, 1981*. National Transportation Safety Board. NTSB-AAR-82-5.

[38] M. Palmer, K. Abbott, P. Schutte, and W. Ricks. Implementation of a research prototype onboard fault monitoring and diagnosis system. In *AIAA Computers in Aerospace Conference*, 1987.

[39] Y-C. Pan. *Qualitative Reasonings With Deep-Level Mechanism Models for Diagnosis of Dependent Failures*. PhD thesis, University of Illinois, 1983.

[40] R. Parasuraman. Human-computer monitoring. *The Journal of the Human Factors Society*, 29(6), 1987.

[41] R. Patil. A case study on evolution of system building expertise: medical diagnosis. In Grimson and Patil, editors, *AI in the 1980s and Beyond*, MIT Press, 1987.

[42] Ramesh Patil. Artificial intelligence techniques for diagnostic reasoning in medicine. In Howard Shrobe and the American Association for Artificial Intelligence, editors, *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., 1988.

[43] H. Pople, Jr. Heuristic methods for imposing structure on ill-structured problems: the structuring of medical diagnosis. In P. Szolovitz, editor, *Artificial Intelligence in Medicine*, Westview Press, 1982.

[44] R. Rajagopalan and S. Siddiqi. Aircraft turbojet engine modeling using artificial intelligence techniques. In *AIAA 23rd Aerospace Sciences Meeting*, 1985.

[45] Jens Rasmussen. Skills, rules, and knowledge: signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3), 1983.

[46] James Reggia, Dana Nau, and P. Wang. Diagnostic expert systems based on a set covering model. *International Journal on Man-Machine Studies*, 19:437–460, 1983.

[47] R. Reiter. A theory of diagnosis based on first principles. *Artificial Intelligence*, 32(1), 1987.

[48] E. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[49] Ethan Scarl, J. Jamieson, and C. Delaune. A fault detection and isolation method applied to liquid oxygen loading for the space shuttle. In *International Joint Conference on Artificial Intelligence*, 1985.

[50] D. Scharnhorst. *A Computer Model of Operative Reasoning*. Master's thesis, University of Tennessee Space Institute, 1987.

[51] R. Schudy. A conceptual framework for intelligent real time information processing. In *1987 Conference on Space Operations Automation and Robotics*, 1987.

[52] P. Schutte. Real time fault monitoring for aircraft applications using quantitative simulation and expert systems. In *AIAA Computers in Aerospace Conference*, October 1989.

[53] P. Schutte and K. Abbott. An artificial intelligence approach to onboard fault monitoring and diagnosis for aircraft applications. In *AIAA Guidance, Navigation, and Control Conference*, 1986.

[54] P. Schutte, K. Abbott, M. Palmer, and W. Ricks. An evaluation of a real time fault diagnosis expert system for aircraft applications. In *26th IEEE Conference on Decision and Control*, December 1987.

[55] Mark Shirley. Generating tests by exploiting designed behavior. In *National Conference on Artificial Intelligence*, 1986.

[56] Y. Shoham and N. Goyal. Temporal reasoning in artificial intelligence. In Howard Shrobe and the American Association for Artificial Intelligence, editors, *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, Morgan Kaufmann Publishers, Inc., 1988.

[57] Herbert A. Simon. The structure of ill structured problems. *Artificial Intelligence*, 4:181–201, 1973.

[58] S. M. Weiss, C. Kulikowski, S. Amarel, and A. Safir. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11:145–172, 1978.

NASA
National Aeronautics and
Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102767		

4. Title and Subtitle		5. Report Date
Robust Fault Diagnosis of Physical Systems in Operation		January 1991
		6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Kathy H. Abbott	
	10. Work Unit No.
	505-64-13

9. Performing Organization Name and Address	11. Contract or Grant No.
NASA Langley Research Center Hampton, VA 23665-5225	
	13. Type of Report and Period Covered

12. Sponsoring Agency Name and Address	
National Aeronautics and Space Administration Washington, DC 20546	Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

The information presented in this report was offered as a thesis in partial fulfillment of the requirements for Ph.D. in Computer Science, Rutgers, The State University, New Brunswick, New Jersey, May 1990.

16. Abstract

This thesis presents and demonstrates ideas for improved robustness in diagnostic problem solving of complex physical systems in operation, or operative diagnosis. The first idea is that graceful degradation can be viewed as reasoning at higher levels of abstraction (less detail) whenever the more detailed levels proved to be incomplete or inadequate. A form of abstraction is defined that applies this view to the problem of diagnosis. In this form of abstraction, named status abstraction, we define two levels. The lower level of abstraction corresponds to the level of detail at which most current knowledge-based diagnosis systems reason. At the higher level, this thesis presents a graph representation that describes the real-world physical system. The thesis demonstrates an incremental, constructive approach to manipulating this graph representation that supports certain characteristics of operative diagnosis. We show the suitability of this constructive approach for diagnosing fault propagation behavior over time, and for sometimes diagnosing systems with feedback. We also show a way to represent different semantics in the same type of graph representation to characterize different types of fault propagation behavior. We demonstrate an approach that threats these different behaviors as different fault classes, and the approach moves to other class when previous classes fail to generates suitable hypotheses.

These ideas are implemented in a computer program named Draphys (Diagnostic Reasoning About Physical Systems) and demonstrated for the domain of inflight aircraft subsystems, specifically a propulsion system (containing two turbofan engines and a fuel system) and hydraulic subsystem.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Artificial Intelligence Model-Based Reasoning Graceful Degradation Operative Diagnosis Decision Aid Airplane Turbofan Engine	Unclassified-Unlimited Subject Category: 03

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	166	A08