Exploring Hypotheses in Attitude Control Fault Diagnosis

Benjamin Bell
GE Astro-Space Division
U7049 P.O. Box 8555 Philadelphia PA 19101

Abstract

Recent activity in spacecraft design has been geared toward providing an assortment of new capabilities in space, in an attempt to satisfy the demanding mission requirements posed by such programs as the Strategic Defense Initiative (SDI) and Space Station. These requirements will include on-board fault detection and fault correction, and current work at GE Astro-Space is addressing this area through the use of knowledge-based systems. This paper describes a system which analyzes telemetry and evaluates hypotheses to explain any anomalies which are observed. Results achieved from a sample set of failure cases are presented, followed by a brief discussion of the benefits derived from this approach.

1. Introduction

The attitude control subsystem (ACS) orients and stabilizes the satellite after launch vehicle seperation, maintains pointing during on-orbit and payload operations, and controls the satellite attitude during orbit-adjust operations. Spacecraft attitude is of critical importance, since a slight error in vehicle alignment not only degrades mission performance, but also may cause changes in momentum rates that may propagate until the spacecraft is spinning out of control. Current architectures provide some level of autonomy in the ACS via closed-loop control, which can generally compensate for attitude errors attributable to normal vehicle dynamics. More serious errors are handled from the ground, during a process which includes a rapid effort to put the satellite into a safe state (thirty minutes to several hours), followed by an analysis phase which may take several weeks. During this time, mission performance may be interrupted, a situation which is not compatible with Government objectives.

To better maintain mission performance and to avoid propagation of attitude errors, anomalous conditions must be detected and isolated as rapidly as possible. This goal becomes harder to achieve as satellites become more complex; but if the satellite itself were capable of performing fault-isolation, then the risk of serious failure would be greatly reduced, along with the need for highly skilled ground personnel.

The **ACS Diagnostic System** demonstrates the potential of knowledge-based systems to offer this capability. As a preliminary step toward achieving satellite autonomy, this diagnostic program performs ground-based detection and isolation of faults within a satellite ACS. Detection of an anomaly triggers the generation of hypotheses. By extracting information from the telemetry useful to its diagnosis, this system independently pursues each possible explanation. The likelihood of each explanation depends on features identified in the telemetry; additionally, explanations are ruled out when contradictory evidence appears in the telemetry.

2. Current Diagnostic Procedures

Present techniques for diagnosing ACS faults rely on attitude error limits for detection. If the satellite attitude error limits are exceeded, the primary objective is to put the satellite into a safe state, which may involve unloading the momentum in the wheels, performing Auto Sun Acquisition, or disabling thrusters. Diagnosis proceeds only after the satellite is in a safe configuration. The principal technique employed in locating the source of the anomaly is to switch to a redundant component and then recheck the anomalous telemetry. In the case of a Reaction Wheel anomaly, the suspect wheel is turned off and the telemetry is monitored with the satellite operating on three wheels.

Effective application of these procedures requires some degree of expertise. For example, the choice of which component to switch with its redundant partner relies in part on an examination of certain informative telemetry behaviors. The diagnostic system, therefore, requires expertise, so that it may apply knowledge of the ACS and of telemetry analysis to explain a currently observed anomalous condition.

3. Expert Knowledge

To trouble-shoot anomalies, ACS analysts generally rely on their knowledge in the areas of telemetry interpretation, failure mode behavior, and diagnostic strategies. Telemetry interpretation knowledge is applied when the analyst extracts useful information from the large volume of telemetry data. Examples of such information are transients and trends.

Failure mode knowledge may be encoded by capturing the expert's mental representation of a failure. Analysts often characterize an ACS failure in terms of the symptoms (anomalies) which may appear during that failure, including qualitative measures of the support a particular symptom lends to a hypothesized failure. The expert might indicate, for example, that during a tachometer failure there is a high probability that the wheel speeds will oscillate.

Knowledge about diagnostic strategies defines the expert's own internal protocol for diagnosing faults. It was determined through interviews with ACS analysts that the expert initially reacts to an anomaly by considering all possible explanations, creating a mental model for each hypothesis. The expert then seeks evidence which can distinguish among the possible failures, primarily by comparing the actual telemetry to the predicted observations for that failure. Important also is the expert's ability to rule out a failure on the basis of evidence to the contrary. The expert may explain, for example, that the possibility of a drive failure may be ruled out if the motion in the opposite wheel indicates that the wheels are spinning normally.

4. Encoding Expertise

An appropriate representation must be selected for each of the three types of knowledge discussed above. Telemetry interpretation knowledge consists of categorical descriptions of anomalous behaviors, or *features*. This knowledge is encoded as rules, with each rule capable of identifying whether a specific feature is present in a telemetry point's recent

value history. The one-to-one correspondence between feature types and rules facilitates knowledge engineering and rapid prototyping.

Failure mode knowledge characterizes ACS failures as the collection of symptoms which may appear during that failure. A suitable representation for this type of expertise is a *schema*, an object which is composed of *slots* that specify the attributes of the schema. One schema completely defines a failure in terms of its symptoms, with each symptom described in its own slot. This slot description identifies the name of the symptom, the maximum time it would take for the symptom to appear during the failure, and the likelihood of the symptom appearing during the failure. Likelihoods are expressed qualitatively with the symbols 'H', 'M', and 'L' (High, Medium, and Low), 'A', and 'N'. An 'A' indicates that the symptom must **always** appear, and 'N' that the symptom will **never** appear. Figure 1 shows a sample failure schema. Because each symptom has attributes which

```
(defschema pss+.failure
  (symptom ((jump-to-zero pss+.current.use) A (00 00 35)))
  (symptom ((steady-decrease pss-.current.use) A (00 01 30)))
  (symptom ((steady-decrease yrss+.current) M (00 04 30)))
  (symptom ((steady-decrease yrss-.current) M (00 04 30)))
  (symptom ((gradual-decrease estimated.aom-y) M (00 06 00)))
  (symptom ((transient estimated.att-p) H (00 00 05)))
  (symptom ((transient estimated.att-r) H (00 00 30)))
  (symptom ((transient estimated.att-y) H (00 00 05)))
  (symptom ((unbalanced py+.wheel.speed py-.wheel.speed) M (00 02 30)))
  (symptom ((oscillation pr+.wheel.drive) L (00 00 00)))
  (symptom ((oscillation pr-.wheel.drive) L (00 00 00)))
```

Figure 1: A sample failure definition.

are independent of any one failure, schemas are used for defining symptoms as well. In this case the symptom is linked with failures by identifying each failure as a *possible cause* of the symptom, where appropriate. This schema organization allows rapid access to the predefined symptom and failure characteristics, and its modularity permits rapid prototyping as well.

5. Encoding Diagnostic Strategy

The third type of expertise, diagnostic strategy, differs in that it is comprised of *procedural* knowledge, whereas the two previous categories of expertise encompass *declarative* knowledge. This leads to a different implementation: rather than employing rules or schemas, this knowledge is represented as **meta-rules** which govern the way in which the declarative knowledge is applied to the problem-solving task.

The objective of this meta-structure is to implement a reasoning strategy derived from the techniques employed by human experts. The diagnostic knowledge discussed earlier is suitably represented using the Set Covering Model [2]. Applying this approach, generating hypotheses to explain a symptom is simply a matter of locating the symptom's schema

and reading the possible causes for that symptom. Each such possible cause then becomes a hypothesis. Hypothesis evaluation is accomplished by observing how well a hypothesis *covers* the symptom-set, i.e. how many of its symptoms have been detected. Symptoms vary in how strongly their presence supports a hypothesis, so this factor is incorporated into the evaluation procedure. In addition, some symptoms are required to support a hypothesis, and so that symptom's absence will allow the system to rule out the hypothesis. Conversely, some symptoms provide contradictory evidence, so that a hypothesis may be ruled out on the basis of that symptom's presence. This strategy is an appropriate representation for the diagnostic expertise discussed above, but requires a mechanism for handling the multiplicity of hypotheses. Fortunately, such a mechanism is available in this system.

Hypothetical Reasoning

The use of hypothetical reasoning is of prime importance to the diagnostic capability of this module. Using this approach, one hypothetical situation ('viewpoint') is created for each possible failure. The viewpoints are distinct from eachother, so each operates under its own set of assumptions (including of course the assumption about which failure occurred). When evidence rules out a hypothesis, the associated viewpoint is eliminated. Thus the use of hypothetical reasoning allows the system to pursue multiple hypotheses simultaneously. The reasoning within each viewpoint is geared towards supporting the hypothesis which generated that viewpoint, by setting goals to look for symptoms of the corresponding failure, and so is called 'goal-directed reasoning'.

Goal-Directed Reasoning

Hypotheses are generated in a forward-reasoning fashion. This means that the detection of a symptom triggers the hypothesizing mechanism, because the system has been 'told' that **if** a symptom occurs **then** it should hypothesize all failures which could cause that symptom. Once the hypotheses have been generated, however, the reasoning occurs in reverse. In backward reasoning, the system is told, for example, that **if** Symptom-A occurs **then** the likelihood of Failure-1 is increased. This generates a goal of detecting Symptom-A. Suppose the symptom knows that **if** the reaction wheels are unbalanced **then** conclude that Symptom-A has occurred. Then a *subgoal* is generated to detect an unbalanced wheel pair.

Within each viewpoint, then, unique subgoals are generated. This is a very important feature because of the computing expense involved in detecting symptoms. The savings is realized by only looking for symptoms which will help to support or rule out hypotheses. Savings also is achieved in the case when a hypothesis is ruled out, because all goals generated to support that hypothesis are eliminated.

6. Implementation

Combining this reasoning strategy, the failure and symptom schemas, and the feature-identification rules results in a system which emulates the diagnostic performance of a human expert. The process operates first in a fault detection mode, and then if triggerred, in a fault isolation mode.

Fault Detection

The current practice of fault detection by limit-checking has an inherent limitation, in that an anomaly may go undetected because no limits are exceeded. Moreover, even when a telemetry value exceeds its limits, it may not do so immediately upon failure, but instead may remain within limits for several minutes after the failure occurs. Further, simulations of ACS failures indicate that transients are reliable fault indicators. The fault detection mechanism used in this system, therefore, achieves the earliest possible fault detection, by using any ACS transient to trigger activation of the diagnostic procedures. This detection is the responsibility of the feature-identifying rule for transients, which monitors the histories of ACS telemetry points. The detection of a transient triggers the creation of **viewpoints**, each of which contains the assumption that one particular failure has occurred.

Fault Isolation

The meta-rules responsible for viewpoint creation not only hypothesize failures, but also scan each failure's list of symptoms, and set as *goals* the determination of these symptoms' presence or absence. These goals are satisfied by the feature-identifying rules, each of which is specific to a particular symptom type. A goal to find an oscillation in a wheel drive, for example, would activate a rule which 'knows' how to identify oscillations. In this way, only those symptoms relevant to the diagnosis are investigated.

The status of a symptom is initially `UNKNOWN`. Once it becomes a goal, it is assigned an expiration time, by adding that symptom's maximum appearance time to the current time. If it is observed prior to its expiration, it is assigned a status of `KNOWN-PRESENT`, otherwise its status is `KNOWN-ABSENT`. This information is processed by probability determination rules, which maintain a current probability for each hypothesized failure. The contribution a symptom's presence makes to a failure's probability depends on the likelihood, expressed qualitatively with the symbols 'H', 'M', and 'L', of that symptom occuring during the failure. These values are stored as part of the symptom slots in the failure schema, and are assigned numerical equivalents for computational purposes. If a symptom is known to be absent, it contributes a corresponding negative likelihood. Unknown symptoms are not included in the calculation. Probabilities are kept current by rules which react to a change in a symptom's status, by adjusting the probability of any failure related to that symptom (i.e. any failure identified as a **possible cause** in the symptom's schema).

The probability analysis helps the operator to distinguish among failures by providing a basis for comparing the alternative hypotheses. But more powerful than that is the system's capability to rule out failures on the basis of evidence in the telemetry. A failure may be ruled out by the program under two conditions (the operator may also rule out a failure). The first occurs when a symptom required to support the hypothesis is absent. This is detected when a symptom identified in a failure schema as an 'A' symptom has a status of `KNOWN-ABSENT`. The second condition occurs when a symptom which would not appear during this failure is observed. This is indicated when a symptom identified in a failure schema as an 'N' symptom has a status of `KNOWN-PRESENT`. When a failure is ruled

out, all goals set to support that hypothesis are removed, and the reason for ruling out the failure is recorded in its schema.

7. Test Environment

The diagnostic capability of this system was examined within a test environment which provided simulated spacecraft telemetry and an interactive operator station. The telemetry was derived from actual satellite telemetry (for 'normal' data) and from an ACS simulator (for failure data). Telemetry processing software generates telemetry frames and sends one frame to the diagnostic system every two seconds, to create a real-time environment. The operator station keeps the operator fully informed about detected anomalies and about the status of each failure being investigated. Three displays are available: a table of all detected anomalies, a symptom display detailing a selected symptom, and a failure display providing data about a selected failure.

8. Example

When the ACS Diagnostic System first detects a transient in ACS telemetry, the hypothetical reasoning mechanism creates four possible explanations: a tachometer failure, a sun sensor failure, a wheel drive failure, and an Attitude Control Electronics (ACE) failure. The goal-directed reasoning then sets as goals the detection of symptoms appearing in each of these failures. As a result, the feature-identifiers are activated and telemetry analysis begins. Figure 2 illustrates the result of the operator selecting a ruled-out failure for display. A few moments later, the operator selects an observed anomaly, which brings up the symptom display shown in Figure 3. After about forty seconds, a single failure remains, but the system continues its analysis until the operator decides to accept the failure. Figure 4 shows the main display after two and a half minutes, identifying the relevant features found in the telemetry since the failure was first detected.

9. Conclusion

This prototype demonstrates the promise of the approach used because effective reasoning was achieved using a straightforward representation and relatively simple heuristics. Further, because of the way the expertise is segmented, the system can not only be made to perform better in this domain, but can also be applied to different problem areas. Installing operational versions of an expert system such as this one thus becomes more cost-effective, as new diagnostic systems can be generated from existing ones by replacing a specific knowledge base and leaving the reasoning strategies intact.

Certainly, then, AI-based diagnosis will help to increase mission reliability and reduce the need for highly-skilled personnel. As this technology evolves, advances in the intelligence of these systems will provide even greater benefit. One way to improve knowledge-based systems is through the use of learning. Expert systems can learn by modifying their strategies when such strategies fail to provide acceptable solutions. This approach has been tested in a satellite diagnosis system [1], and future work will provide large reductions in the time and cost involved in knowledge engineering.

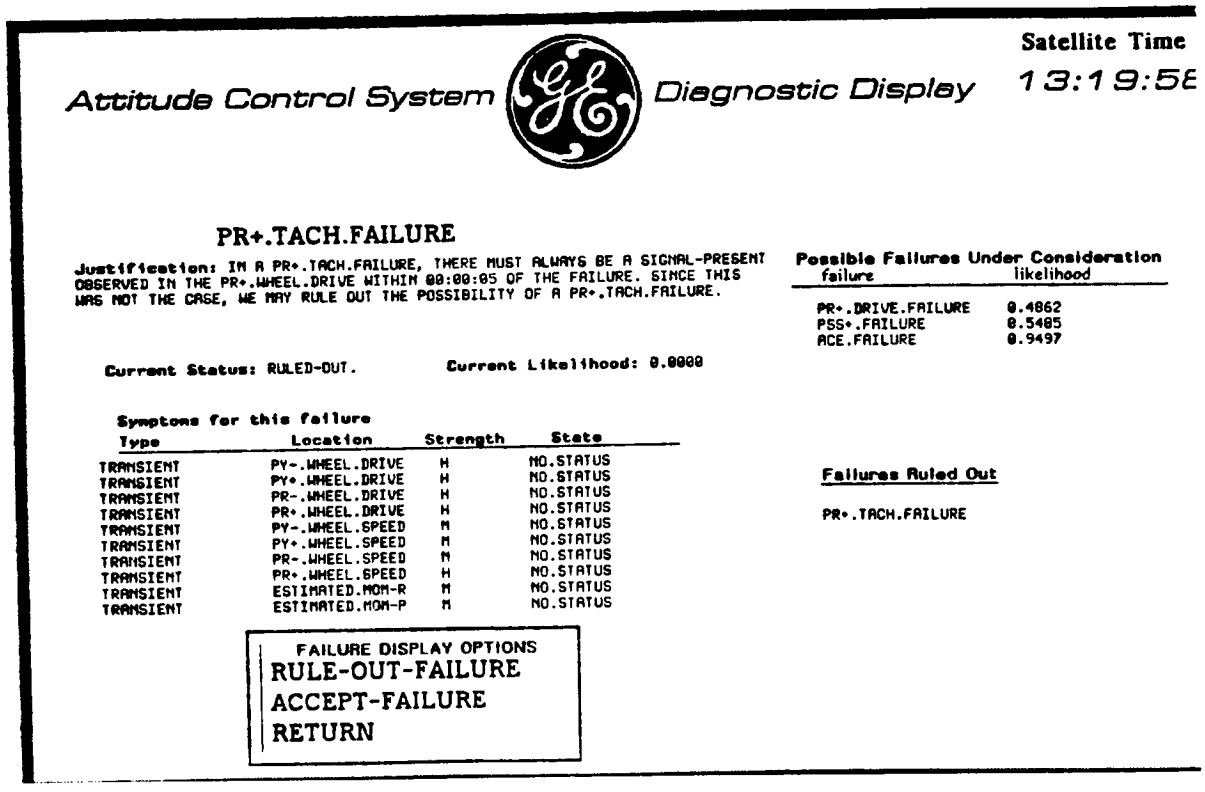

Figure 2: Failure Display Screen example.

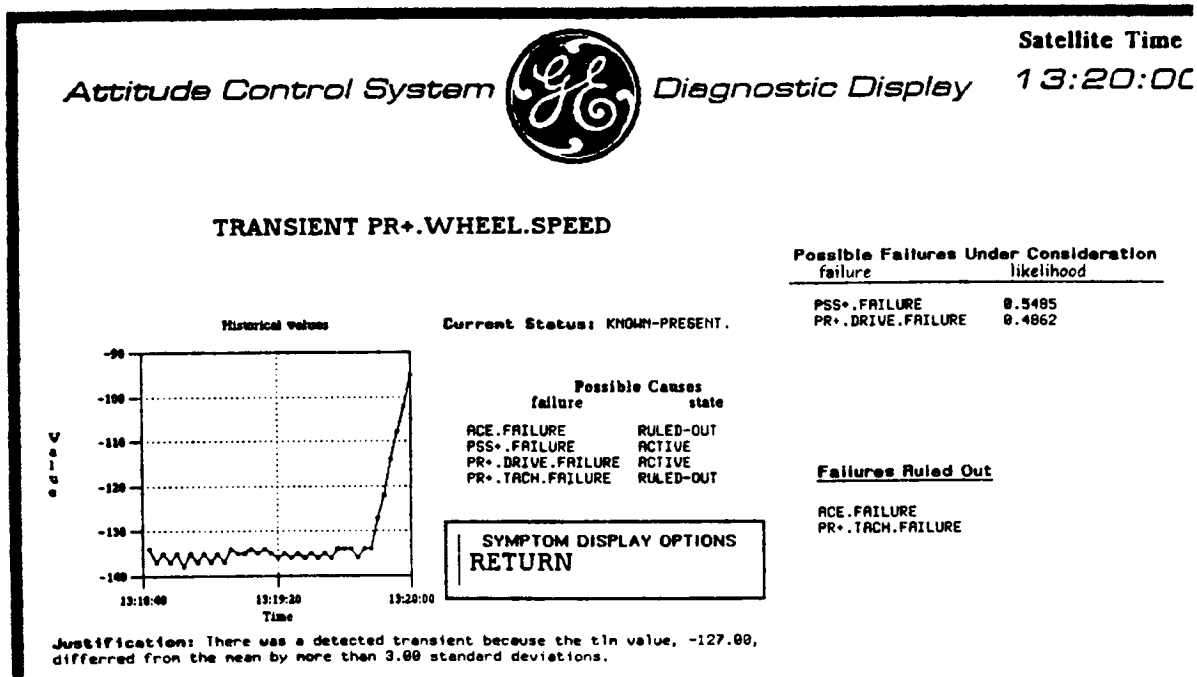

Figure 3: Symptom Display Screen example.

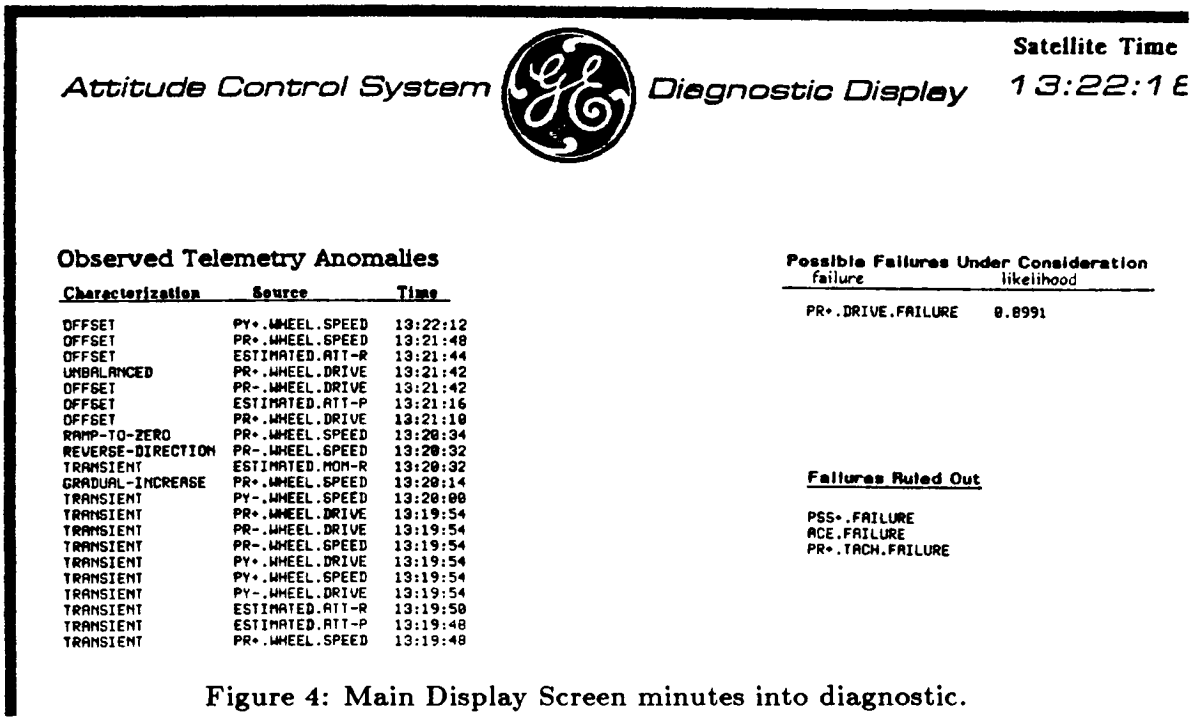

Figure 4: Main Display Screen minutes into diagnostic.

Another area under investigation is the use of model-based reasoning. An expert system could predict behavior using models of the satellite and its subsystems, and so could identify faults by differences observed between actual behavior and behavior predicted by the model. Isolation of faults is also facilitated by the causal links implicit in these models. Applying models to diagnostic expert systems is currently under study at GE Astro-Space. Results from this and other research promise to put more intelligence into AI, so that the increasing complexity of space systems is paralleled by our improved ability to control and maintain them, and eventually, by their ability to maintain themselves.

References

Pazzani, Michael J. (1986). Refining the Knowledge Base of a Diagnostic Expert System: An Application of Failure-Driven Learning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1029–1035, AAAI, Philadelphia, PA.

Reggia, James A., D. S. Nau, and P. Y. Wang (1984). Diagnostic Expert Systems Based on a Set Covering Model. In M. J. Coombs (Ed.), *Developments in Expert Systems*, 35–58. London: Academic Press.