N88-16378

Knowledge-Based Monitoring of the Pointing Control System on the Hubble Space Telescope

Larry L. Dunham
Thomas J. Laffey
Simon M. Kao
James L. Schmidt
Jackson Y. Read

Lockheed Missiles and Space Co., Inc
O/62-85 B/579,
P.O. Box 3504
Sunnyvale, CA 94088-3504
(408) 742-3415

Abstract

This paper describes a knowledge-based system for real-time monitoring of telemetry data from the Pointing and Control System (PCS) of the NASA Hubble Space Telescope (HST) that enables retention of design expertise throughout the three decade project lifespan by a means other than personnel or documentation. The system will monitor performance, vehicle status, success or failure of various maneuvers, and in some cases diagnose problems and recommend corrective actions using a knowledge base built using nominal mission scenarios and the over 4,500 telemetry monitors from the HST. The real-time system consists of a data management task, an inferencing task, and an I/O task that run concurrently in multiple CPUs and communicate via a message passing scheme. Real-time graphical displays can be selected by the user on the multi-level block diagram of the HST control system displayed by the I/O task. This paper describes the application of L*STAR to analysis of monitors from the PCS. A detailed description of the multiprocessing architecture will be described in another paper in the conference. L*STAR is undergoing continued development and is being used to monitor test cases produced by the Bass Telemetry System in the Hardware/Software Integration Facility at Lockheed Missile and Space Co. in Sunnyvale, California. LMSC is assembling the vehicle under the direction of NASA/Marshall with a 1989 launch planned.

Introduction

Lockheed Missiles and Space Company (LMSC) is the prime contractor for the Support Systems Module (SSM) and Integration Systems Engineering for NASA's Edwin P. Hubble Space Telescope (HST). The HST is considered one of the greatest scientific experiments in history of mankind. The field of astronomy will be revolutionized by the opportunity to see to the edges of the universe (14 billion light years away), seven times further than we can now observe. Unimpeded by the Earth's atmosphere, scientists will be capable of seeing objects fifty times fainter than those visible today with a stability equivalent to focusing a laser beam in Washington, D.C. on a dime in Boston!

The launch of the HST by the Space Shuttle has been delayed due to the Challenger space shuttle accident. The current launch date is late 1989 and the lifetime of the spacecraft after launch is expected to be a minimum of fifteen years. This gives the total project, from design to end of mission, a lifetime of over a quarter of a century. Capturing knowledge of engineering experts to ensure continued expertise over the project's lifetime is one of the goals of the application described in this paper.

A second factor driving this application is the complexity of the ground operations task. The Space Telescope Operations and Control Center (STOCC) at the NASA/Goddard Space Flight Center in Greenbelt, Maryland, will monitor the vehicle's health and safety 24 hours a day using three shifts of operators. There are almost 5,000 different telemetry monitors in 11 different possible formats. Each format has a subset of monitors available in it, and the rate at which a specific monitor is reported varies from 40 hertz to 0.025 hertz. Execution of on-board stored program commands (SPC) are handled by a 40 hertz processing rate making it impossible for the operator to watch individual command executions.

Lockheed's Artificial Intelligence Center in Menlo Park, California, has been working on developing a real-time monitoring tool called *L*STAR* (Lockheed Satellite Telemetry Analysis in Real-Time). This knowledge-based monitoring system is still under development with initial rule bases being centered on the Pointing and Control System (PCS), the Data Management System (DMS), and the Electrical Power System (EPS). This paper will address the PCS application of the *L*STAR* system. The PCS application, while far from completion, has already led to many valuable insights.

Requirements

The real-time requirements of the PCS include the following: command verification, safemode prevention warnings, configuration validation, performance assessment, and configuration monitoring.

Command verification is the process of checking a new command against the configuration of the spacecraft prior to the sending of that command. This is to ensure that the command will not endanger the vehicle or interrupt the current mission. Verification would include checks to prevent commands that would expose a scientific instrument aperture to a bright object such as the Earth or the Sun, for example.

Safemode prevention warnings are messages to the operator that indicate a dangerous condition developing and show the autonomous safety system checks done by the HST flight computer that will initiate a response if ground action isn't taken. The response by the flight computer's safemode system can range from shutting down subsystems to shutting the whole flight computer down and passing control of the vehicle over to the PSEA (Pointing/Safing Electronics Assembly). Almost all responses by the safemode system abort the current command list and require recovery by the ground system. The safemode prevention warnings are intended to warn the ground so that either the safemode response can be avoided or anticipated.

Configuration validation checks compare the modes of various subsystems and ensure that they are compatible. Certain states should never occur simultaneously. For example, the attitude of the vehicle is not available for computations when the vehicle is in Drift mode (used in deployment and recovery from safemode) or trying to align with the Sun (Sun Point Control). Computations such as the Momentum Management's Minimum Energy Law which predicts the momentum profile for the next half orbit using the current attitude should not be active during either of those two states.

Performance assessment allows the operator to know in real-time how successful the planned mission is. For example, at each point in the mission, there is an anticipated inaccuracy in the vehicle attitude. After an attitude update by the Fixed Head Star Trackers (FHST), the error should be less than ten arcseconds, and after establishing lock on stars with the Fine Guidance Sensors (FGS), the error should not exceed the radius of the search used to find the stars. The operator would be warned if reported attitude errors did not meet the expectations for any given time.

Configuration monitoring simply reports, to the operator, changes in the mode of any subsystem. This includes unplanned changes such as those resulting from unexpected loss of lock on the stars being used for guidance of the vehicle.

Organization of System

L*STAR uses rules from its knowledge base to intelligently monitor the HST telemetry stream. So that the system does not have to examine the entire ruleset, each rule has certain contexts in which it is valid. The rule will not be examined or triggered if its context does not match the current context of the inferencing system. For instance, a rule to check the performance of a vehicle maneuver does not need to be tested during Drift mode. Rules may be applicable in a single context or multiple contexts. The context of the inference engine is at all times identical to the mode of the HST. The mode of the HST is an enumerated attribute with currently ten legal values: *Drift, Inertial Hold, Science, Sun Pointing Control, Loss of Lock, Attitude Hold, Maneuver, FGS Acquisition, FHST Acquisition, Mechanism Motion*, and *PSEA*.

Each flight computer software subsystem (total of 13) is a different class (i.e., schemata or frame type) with unique attributes. They all have at least two attributes called Status and Mode. Status is typically either **normal** or **abnormal** and Mode has values that are specific to the subsystem.

A natural way to organize the rules that will potentially number in the thousands was to put each subsystem's rules in separate files. In each file the rules are titled with a number assigned (similar to the Dewey Decimal System) as shown in Table 1. In some cases the rules may fit two categories, in which case the lower number is used.

```
1.x - determines subsystem mode
2.x - determines subsystem variable attributes
3.x - context dependent limit checking
4.x - checks for invalid mode switch or illegal
      variable attributes
5.x - outputs messages to operator
```

Table 1 - Rule Number Convention

A sample rule for testing to ensure that speed of reaction wheel 1 is normal for certain contexts would look as follows:

```
RULE       : "[3.1] Reaction Wheel 1 Check";
CONTEXT    : { Inertial_Hold, FGS_Acq, FHST_Acq, Science,
               Mechanism_Motion };
PRIORITY   : 100;
AUTHOR     : "Simon Kao/Larry Dunham";

IF   ([value\monitor\QDVOMEVO] > 12.0) (*radians/sec*)
THEN [status\momentum_management\MOMAN] := abnormal;
     send( IO, ALERT, "QDVOMEVO", "Wheel speed 1 high at
                                   %[time\satellite\HST]");!
```

An identical rule is needed for each of the other three reaction wheels. This need for vector notation is common in satellite telemetry systems. Many monitors are position vectors in the vehicle frame, or are sets of values for identical hardware (6 gyros, 4 wheels, etc.). Facilities for processing and manipulating vectors is generally not available in commercial AI tools. This capability is being added to *L*STAR* and should lead to a significant decrease in the total number of rules.

Temporal Reasoning

Commercial AI tools have few, if any, capabilities for reasoning about past, present, and future events. For satellite telemetry monitoring it is a necessity to have such capabilities. *L*STAR* has implemented them as built-in functions of the inference engine. Temporal reasoning in the simplest form is the ability to use trends and statistics in rules via functions such as rate-of-change and average value over a time period, for example. This makes it straightforward to write rules based on the vehicle error decreasing or the commanded body rate being steady.

The more complex part of temporal reasoning involves the difficulties of handling data from telemetry with various time tags associated with them. The monitors are reported at various rates and at various times. For example, if telemetry flags A and B cannot be true at the same time, the fact that the last reported values are both true should not fire a rule for illegal configuration. A or B may have just been reported true and the other value has not been reported since that time, and the system needs to wait for the next sample of the second monitor and discover if it is reported as still being true. If it is, then the assumption (which in some cases is not valid) is that if one of the monitors was reported true, both before and after the time that the other one was reported true, then at some point in time they must have both been true simultaneously. However, for fast changing monitors with slow reporting rates, even this logic is faulty.

An example rule is that the Minimum Energy Law should be off when the vehicle mode is in Sun Point control. The vehicle mode flag changes very slowly, so that the user is assured that two identical samples ensure a constant mode over that time period. If the vehicle mode were reported twice as frequently as the Minimum Energy Law flag, then the Table 2 shows the valid and invalid telemetry patterns.

		valid sequences					invalid sequences		
t = 1.0	ME sample	on	on	on	off	off	on	on	off
t = 1.01	SP sample	off	off	on	on	on	off	off	on
t = 1.06	SP sample	off	on	on	on	off	off	on	on
t = 1.1	ME sample	off	off	off	on	on	on	on	on
t = 1.11	SP sample	on	on	on	off	off	on	on	on

Table 2 - Valid/Invalid Sequences for Temporal Reasoning

A rule prohibiting ((ME on) AND (SP on)) would have prohibited three of the valid states. The *L*STAR* Inference Engine has an AND function being added to it to handle rules of this type properly. This need is common in all telemetry sampled systems. Tools which process rules based only on change data (e.g., an OPS5 based on the Rete network) cannot handle these types of Relationships

*L*STAR* User Interface

The I/O Process, which can run on its own processor, provides sophisticated displays that help both the console operators and the analysts. The operator is provided with messages that are either information, alerts, or warnings. These messages are stored and recallable using the mouse.

For the PCS, a set of hierarchical displays of the flight computer software is done in great detail. The diagram shows the relationships of the various monitors. Each monitor can be plotted on the screen in real-time by simply mousing the monitor name on the diagram. The analyst can then track problems backward using the diagram to determine the initial monitor that indicates abnormal behavior.

Conclusions

The *L*STAR* system has been proven to be able to handle real data from HST tests and perform the monitoring in real-time. The multi-processor design allows for multiple inference processes to be distributed to additional processors if the rule-base becomes unmanageable in real-time for one processor.

The insights into the problems of satellite telemetry systems with regard to easy vector notation and temporal reasoning have shown commercial tools severely lacking. *L*STAR* is an attempt to fill that niche. Both insights and answers have been gained by having a team consisting of personnel from the LMSC AI Center, the HST flight software development group, and the ground system operations group.

This system is currently under development and is being used to monitor test cases produced by the Bass Telemetry System in the Hardware/Software Integration Facility at Lockheed Missile and Space Co. in Sunnyvale, California.

Acknowledgements

The authors wish to acknowledge the encouragement and support received from Wally Whittier, LMSC HST S/W Program Manager in Sunnyvale, California.