Expert System Support for HST Operations

by

Bryant Cruse
Lockheed Missiles and Space Corp.
Code 400.8, NASA/GSFC
Greenbelt, MD 20771

Charles Wende
Space Telescope Project - Goddard
Code 400.2, NASA/GSFC
Greenbelt, MD 20771

ABSTRACT

An expert system is being developed to support vehicle anomaly diagnosis for the Hubble Space Telescope. Following a study of safemode entry analyses, a prototype system was developed which reads engineering telemetry formats, and, when a safemode event is detected, extracts telemetry from the downlink and writes it into a knowledge base for more detailed analysis. The prototype then summarizes vehicle events (limits exceeded, specific failures, etc.). This prototype, the Telemetry Analysis Logic for Operations Support (TALOS) uses the Lockheed Expert System (LES) shell, and includes over 1600 facts, 230 rules, and 27 goals. Although considered a prototype, it is already an operationally useful system.

The history leading into the TALOS prototype will be discussed, an overview of the present TALOS system will be presented, and the role of the TALOS system in contingency planning will be delineated.

The Hubble Space Telescope (HST) is not, for the most part, an autonomous spacecraft. Its engineering telemetry will be monitored for vehicle health and safety on a nearly continuous basis from the ST Operations Control Center (STOCC) at the NASA/Goddard Space Flight Center in Greenbelt, MD. STOCC personnel must recognize and respond to anomalies by initiating the appropriate contingency procedures. One exception to this dependance on ground personnel is the vehicle safemode system. An on-board computer continually tests critical vehicle subsystems. When one of these tests fails, a predefined sequence of stored commands is exercised to place the vehicle in a safe configuration, or a "safemode". Several safemodes have been defined and are activated depending on the nature and severity of the malfunction. Each safemode is designed to isolate the failed subsystem or component and then to place the vehicle in a stable, powerconserving attitude. The safemode system buys time for the STOCC personnel to respond to a serious on-board situation. It is still incumbent upon the STOCC to recognize that the vehicle has entered safemode, to determine which safemode test or tests have failed, and to diagnose the cause of the problem. These tasks must be acomplished before the vehicle can be recovered and the science schedule resumed. The development of tools that can speed up these analyses, therefore, has a very high payoff for enhancing mission operations.

Analysis of a vehicle safemode event requires analyzing raw telemetry which appears in one of a variety of formats depending upon, among other things, the type of safemode entry (which is to be determined!). Following a safemode recovery study in 1984, it was recognized that, because of the complexity of this task, some sort of ground software assistance would be needed if the HST were to be operated efficiently. Lockheed Missiles and Space Company, the Mission Operations Contractor (MOC) for HST, undertook to write a prototype expert system (the Telemetry Analysis Logic for Operations Support, TALOS, system) to attack this problem in the summer and fall of 1986.

In its current state of development, TALOS operates in either of two modes. In the monitor mode, TALOS scans a telemetry history file (optionally starting from a specified time) and looks for the existence of any safemode event; if an event is found, it automatically changes to the diagnostic mode. Upon entering the diagnostic mode, TALOS

extracts the values of specific telemetry monitors from the history file and writes them to the system knowledge base. The TALOS system then performs the following analysis tasks:

determines whether the HST itself is in a safemode;

and if so,

assesses the sequence of vehicle events,

summarizes what happened and when, and

verifies that the vehicle response was correct.

If desired, the operator can ask for a rationale explaining why any particular conclusion was reached. The TALOS system consists of four major subsystems, of which two were provided by the MOC and two were provided in the Lockheed Expert System (LES) shell:

- a Data Interface (developed by the MOC)
- a Knowledge Base (populated by the MOC),
- an Inference Engine (provided by LES), and
- a Knowledge Interface (also provided by LES, but customized for this application).

The Data Interface consists of an adaptive telemetry extraction program written in FORTRAN. Presently it reads data only from an HST engineering telemetry history tape; enhancements will allow reading real-time engineering telemetry streams or disk-based data. The extractor selects 170 monitors (data points) out of the 4690 monitors available, and performs quality checks before reformatting and forwarding the data. Eleven telemetry formats are available, with up to 4015 parameters being downlinked in any one format. Each of these parameters are sampled at least once every two minutes, and some are sampled many times in that interval. The telemetry format itself may be changed autonomously by the HST spacecraft when a safemode situation is encountered. Format changes in the telemetry stream are recognized automatically and are handled almost instantaneously by the Data Interface. The set of monitors being extracted can be changed in less than five seconds under the control of either the console operator or the expert system. The telemetry commutation schemes are stored in a database and are subject to change during the mission.

However, a different commutation scheme can be loaded into TALOS in a matter of seconds under either operator or expert system control. Thus, old data can be revisited for testing, training, or comparison purposes without requiring significant software changes or substantial operator intervention.

The Knowledge Base includes 1600 facts, 230 rules, and 27 goals. As an entity, it is already more knowledgeable about safemode entry than the average console operator was during the Ground System Thermal-Vacuum Test. Knowledge is represented in four ways:

backward chaining, goal-driven rules (if ... then...),

forward chaining, data-driven rules (when ... then...),

facts stored as slots in frames, and

goals which can be run concurrently with dynamically changeable priorities.

The following example illustrates a Backward Chaining Rule in the Knowledge Base:

```
HEADING:

   RULE_NAME          'SMEVENT1'
   FROM_WHOM          'BRYANT CRUSE'
   ACT_TIME           '17 NOV 1986'
   AUTHOR_ENGLISH     'If, after a gyro test failure in low'-
                      ' mode, the gyros are found in high'-
                      ' mode, the software response to the'-
                      ' test failure is nominal.'

IF:

   TYPE_ENTRY         'STATE'
   ACTOR              'RESULT[SMTEST(PNAME=SMTEST1)]'
   ACTION_VERB        '='
   OBJECT             'FAILED'

   TYPE_ENTRY         'STATE'
   ACTOR              'VALUE[MONITOR(PNAME=QDFHILO)]'
   ACTION_VERB        '='
   OBJECT             '0'

   TYPE_ENTRY         'ACTION'
   ACTOR              'ROBOT(PNAME=LES)'
   ACTION_VERB        'PRINT'
   OBJECT             'The DF-224 has responded normally to '-
                      'DESCRIPTION[SMEVENT(PNAME=SMEVENT1)].'

THEN:

   TYPE_ENTRY         'STATE'
   ACTOR              'EFFECT[SMEVENT(PNAME=SMEVENT1)]'
   ACTION_VERB        '='
   OBJECT             'VERIFIED_NOMINAL'
   LIKELIHOOD         '100'
```

This rule fires when the condition following the THEN statement is exactly matched by a condition following an IF statement of another rule or by an Hypothesis statement of a goal. When the rule fires the system then tries to find a match for the conditions following the IF statement in in the knowledge base or in the THEN clauses of other rules. Note the ACTION statement (third entry under IF). LES will execute such a statement in an IF clause when the other two statements are matched. In this case, a text block is written to the screen to inform the user of the result.

The following example illustrates a Forward Chaining rule:

```
HEADING:

   RULE_NAME          'SMWHEN-9'
   FROM_WHOM          'BRYANT CRUSE'
   ACT_TIME           ' 9-DEC-1986'
   AUTHOR_ENGLISH     'When the value of DTMFDC (telemetry '-
                      ' format data content monitor) is '-
                      ' determined and it is not equal to '-
                      ' 145 (S format) or 48 (C format) then'-
                      ' the number of safemode events equals'-
                      ' the value safemode fault recorder'-
                      ' pointer divided by 8.'

WHEN:

   TYPE_ENTRY         'STATE_CHANGE'
   ACTOR              'VALUE[MONITOR(PNAME=SSFRPTR)]'
   ACTION_VERB        'IS_DETERMINED'

   TYPE_ENTRY         'STATE'
   ACTOR              'VALUE[MONITOR(PNAME=DTMFDC)]'
   ACTION_VERB        'IS_DETERMINED'

   TYPE_ENTRY         'STATE'
   ACTOR              'VALUE[MONITOR(PNAME=DTMFDC)]'
   ACTION_VERB        '^='
   OBJECT             '145'

   TYPE_ENTRY         'STATE'
   ACTOR              'VALUE[MONITOR(PNAME=DTMFDC)]'
   ACTION_VERB        '^='
   OBJECT             '48'

THEN:

   TYPE_ENTRY         'STATE_CHANGE'
   ACTOR              'NUMBER_OF_EVENTS[EVENT_SEQUENCE(PNAME='-
                      'SMSEQUENCE)]'
   ACTION_VERB        '='
   OBJECT             'VALUE[MONITOR(PNAME=SSFRPTR)] / 8'
```

This rule will fire only when all four conditions following the WHEN statement are met. Those conditions are checked by the system each time a condition defined by a TYPE_ENTRY of 'STATE_CHANGE' undergoes some change in the knowledge base. When the rule fires, the condition following the THEN statement becomes true.

The following example illustrates a simple category file storing facts in slots in frames. This file defines the different safemodes to the expert system. Any number of attributes can be defined.

```
FILENAME:SAFEMODE_LEVEL.CAT

        SAFEMODE_LEVEL


/*** ATTRIBUTE DEFINITIONS

  ATTRIBUTE_NAME    ACTIVE
  TYPE_ATTRIBUTE    ACTIVE                'TRUE-FALSE'
  ASKABLE           ACTIVE                'FALSE'

  ATTRIBUTE_NAME    GROUND_COMMANDED
  TYPE_ATTRIBUTE    GROUND_COMMANDED      'TRUE-FALSE'
  ASKABLE           GROUND_COMMANDED      'TRUE'

/*** TOKENS

  PNAME             'SMLEVEL0'
  DESCRIPTION       '  No safemode events have occurred.  '

  PNAME             'SMLEVEL1'
  DESCRIPTION       '  The vehicle is in Inertial Hold Mode. '

  PNAME             'SMLEVEL2'
  DESCRIPTION       '  The vehicle is in Software Sunpoint
                    Safemode. '

  PNAME             'SMLEVEL3'
  DESCRIPTION       '   The vehicle is in Hardware Sunpoint
                    Safemode. '

  PNAME             'SMLEVEL4'
  DESCRIPTION       '  The vehicle is in Gravity Gradient
                    Mode. '

  PNAME             'SMLEVEL5'
  DESCRIPTION       '   The vehicle is not in Safemode.'-
                    '  However one or more safemode events
                    have occurred. '
```

The following example illustrates a Goal with a default priority of 95:

```
PNAME                    'SMGOAL4'
DESCRIPTION              'determine the safemode level if
                         any'
GOAL_PRIORITY            '95'
SUBJECT_CATEGORY         'DETERMINE_SAFEMODE_LEVEL'
FIND_ALL_SOLUTIONS       'TRUE'
GOAL_RESULT_PUTOUT       'FALSE'
GOAL_MESSAGE             'I am now determining whether
                         the vehicle'-
                         ' has entered safemode and if so
                         what level.'

HYPOTHESIS

'(ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL0)] = TRUE >< '-
' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL1)] = TRUE >< '-
' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL2)] = TRUE >< '-
' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL3)] = TRUE >< '-
' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL5)] = TRUE >< '-
' ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL6)] = TRUE )'
```

Within LES it is possible to alter the priority of a goal and cause a new line of reasoning to be pursued. This change is implemented using a forward chaining rule of a type generally called "Demons". An example of a demon follows:

HEADING:

```
RULE_NAME          'SMWHEN-01'
FROM_WHOM          'BRYANT CRUSE'
ACT_TIME           '11-AUG-1986'
AUTHOR_ENGLISH     'When no safemode events have occurred'-
                   ' reduce the priority of Goal-6 to 0'
```

WHEN:

```
TYPE_ENTRY         'STATE_CHANGE'
ACTOR              'ACTIVE[SAFEMODE_LEVEL(PNAME=SMLEVEL0)]'
ACTION_VERB        '='
OBJECT             'TRUE'
```

THEN:

```
TYPE_ENTRY         'STATE_CHANGE'
ACTOR              'GOAL_PRIORITY[GOAL(PNAME=SMGOAL6)]'
ACTION_VERB        '='
OBJECT             '0'
```

The Inference Engine provides the standard expert system functions. While in the monitor mode, TALOS reasons in a data-driven manner and awaits the detection of a safemode event before proceeding with the analysis. 20 of the 27 defined goals are initially set to a priority of zero. Upon entering the diagnostic mode, TALOS begins processing goal-driven, backward chaining rules. Then, the priority of a goal may be raised or lowered by data-driven forward chaining rules, depending upon how the analysis proceeds. (Did this fail? Are these monitors available in this format?, etc.) New data can cause further refinements of the priorities. Thus LES is capable of abandoning one line of reasoning and switching to another course of analysis depending upon what it discovers about the state of the HST spacecraft.

The Knowledge Interface (user interface) uses windows to keep the operator appraised of what it has found. At any time, one window is maintaining summary statistics on safemode events, while another window is giving details of the ongoing analysis. At the conclusion of its analysis, TALOS presents its findings and the operator may ask for a printout, or may ask for a detailed rationale behind the findings. By design, TALOS serves to advise the operator and cannot of and by itself issue any corrective commands to the HST spacecraft.

The TALOS has demonstrated its ability to scan a telemetry history tape, to identify an initial safemode event, and to analyse a complex sequence of events correctly. A particularly complex but logically consistant series of safemode events were placed on a telemetry history tape using the Hardware/Software Laboratory at Lockheed in Sunnyvale, CA. Analysis of the telemetry to decipher this sequence would present a real challenge, even to the most expert analyst, would typically require an hour. This sequence of failures proceeds as follows:

> First, the current in the vehicle's magnetic torquer bars exceeds safe limits. This anomaly causes a safemode test to fail, and an on-board computer commands the vehicle to the first level of safemode: Software Sunpoint.
>
> As a result, the solar panels are commanded to rotate. But, since there are no solar panels in the laboratory, another safemode test fails.

Next the battery depth-of-discharge fails through two successive limits (logically, since the solar arrays are mis-aligned). This last failure would normally result in entry into the next level of safemode: Hardware Sunpoint.

In this last level of safemode, a backup computer shuts down the primary on-board computer. However, the lab doesn't have a backup computer either, so the vehicle response is again anomalous.

The printout produced at the end of the analysis clearly shows this sequence of events. On an unloaded system, this entire analysis takes only a few minutes.

TALOS should be understood as applying current technology to the contingency analysis problem. Contingency planning includes:

anomaly recognition,

immediate action definition,

diagnostic techniques, and

recovery plans.

Present TALOS capabilities include fault identification with rationale. Contingency planning maps directly into present and potential TALOS functions; the further development of TALOS will build on our contingency planning. Conversely, TALOS will provide a framework for codifying such planning. By merging the two, it is expected that the TALOS development will force higher degrees of organization, consistency and completeness upon the contingency planning process. The cost will be in training operations personnel to care and feed the TALOS knowledge base, and in the time it takes for these people to insert their contingency plans into the knowledge base itself. However, by testing TALOS against HST spacecraft or simulator data, the contingency analyses can be validated directly, a more thorough testing of TALOS is provided, and a training tool is provided for personnel. Further, the self-documenting nature of the TALOS knowledge base provides paper procedures when needed, while the explanation feature of TALOS provides a teaching tool for new personnel and develops rationales for some unexpected cases.

Development costs thus far have been on the order of a few months of effort and liberated time on a shared VAX/8600 (TALOS also operates quite well solo on a MicroVAX II/GPX). The concept has been demonstrated, and its capabilities will be expanded. The near-term development of additional TALOS capabilities will proceed cautiously, as the a cost of augmenting contingency planning with an expert system will have to be ascertained. TALOS will not be immediately expanded to cover all possible contingencies, but instead will be directed at a small number of high return situations. Three diagnostic modules will be added to service the pointing control system (PCS), the electrical power system (EPS), and the data management system (DMS), and these modules will be limited to handling contingencies related to vehicle safemodes.

The results to date have been very promising.